MODEL DISCOVERY AGENT: LLM-ASSISTED BAYESIAN EXPERIMENT DESIGN FOR DATA-EFFICIENT DISCOVERY OF MECHANISTIC WORLD MODELS

Kevin Murphy
Dept. Computer Science
Univ. British Columbia, Canada.

ABSTRACT

A primary goal of science is to learn mechanistic or causal world models from data. These models can be used to explain some phenomenon of interest. They also provide the ability to answer interventional "what if" questions (i.e., to predict the outcome of an action never taken). Identifying such models usually requires experiments, because passive data leaves the mechanisms unidentified. Since experiments are expensive, we need to develop learning algorithms that are data efficient. We therefore introduce the Model Discovery Agent (MDA), which combines three ingredients: a novel SMC^3 algorithm, which uses 3 levels of nested sequential Monte Carlo (over models, parameters, and latents); a large language model (LLM), which is used as a way to propose new models when the current hypothesis space is detected to be insufficient (c.f., M-open Bayesian inference); and an experiment designer based on maximizing the Value of Information. On three existing benchmarks — FORCEBENCH (Wiemann et al., 2026), CHEMBENCH (Kabra et al., 2026) and BOXINGGYM (Gandhi et al., 2025) — we show that MDA sets a new SOTA in terms of performance. Finally, we introduce NEURONBENCH, a new stochastic single-neuron electrophysiology benchmark, which is significantly harder than current benchmarks, but on which MDA performs well due to its noise-robust Bayesian foundations.

1 INTRODUCTION

In this paper, we develop a new algorithm for learning a mechanistic world model (aka causal world model) from a small amount of data.[1] Such models are the foundation of true *scientific understanding* (Salmon, 1984; Shmueli, 2010; Krenn et al., 2022; Messeri & Crockett, 2024; Bajorath, 2025; Serre & Pavlick, 2025; Kramer et al., 2026; Posner et al., 2026). In addition, they can be used to answer *interventional questions* (Pearl, 2009; Richens & Everitt, 2024): not "what will happen?" but "what *would* happen if I did a?" (e.g., predicting the effect of administering a drug to a patient that it has never received).

Unfortunately, a model with latent variables/ mechanisms is typically *unidentifiable* from observation alone: the passive data underdetermines it, and only intervening — perturbing the system and watching how it responds — breaks the degeneracy. But experiments are expensive (a lab assay, a clinical trial), which makes the operative problem *data efficiency*: identify the mechanism, well enough to answer the queries, in as few experiments as possible. This is the classical remit of

[1]The distinction between causal models and mechanistic models has been discussed at length in the philosophy of science literature (see e.g., (Machamer et al., 2000; Craver, 2006; Woodward, 2004; Weber, 2008; Glennan, 2017; Batterman & Rice, 2014)). We use the term "mechanistic world model" following (Posner et al., 2026). These are models composed of modular, scientifically meaningful building blocks or mechanisms, as in a structural causal model. Note also that we restrict ourselves to "level 2" causality (Pearl, 2009; Bareinboim et al., 2022); this can be handled with standard decision-theoretic machinery (Dawid, 2015; Mlodozeniec et al., 2025), and does not need the more complex machinery required for "level 3" counterfactual reasoning (Dawid, 2000).

Bayesian experimental design — choose the intervention whose outcome is most informative (Lindley, 1956; Chaloner & Verdinelli, 1995; Rainforth et al., 2024) — but it has rarely been combined with the *open-ended hypothesis creation* that scientific discovery demands.

To tackle these problems, we present the **Model Discovery Agent (MDA)**, which combines three ingredients: a novel SMC^3 algorithm, which uses 3 levels of nested sequential Monte Carlo (over models, parameters, and latents); a large language model (LLM), which is used as a way to propose new models when the current hypothesis space is detected to be insufficient (this is needed to tackle the $\mathcal{M}$-*open* regime, where the true model may lie *outside* the current hypothesis class (Bernardo & Smith, 1994; MacKinlay, 2016; Kelter, 2020)); and an experiment designer based on maximizing the Value of Information. We find that discovery and design *reinforce* each other: the experiment identifies the novel model the proposal introduced, and the identified model improves the agent's forecasts, enabling the detection of ever more subtle predictive errors (c.f., (Buehler, 2026)).

On three existing benchmarks — FORCEBENCH (Wiemann et al., 2026), CHEMBENCH (Kabra et al., 2026) and BOXINGGYM (Gandhi et al., 2025) — we show that MDA sets a new SOTA in terms of data-efficient model learning and reliable out-of-distribution prediction ability. Finally, we introduce NEURONBENCH, a new single-neuron electrophysiology benchmark[2], which is significantly harder than current benchmarks, due to its complex stochastic dynamics and high degree of partial observability. Nevertheless, we show that MDA works well in this regime.

2 PROBLEM STATEMENT

Agent-environment interface. We consider an agent interacting with an unknown "blackbox" dynamical system, that maps an optional sequence of inputs or control signals $x_{1:T}$, for $x_t \in \mathcal{X} \subset \mathbb{R}^{d_x}$, to a sequence of noisy observations, $y_{1:T}$, for $y_t \in \mathcal{Y} \subset \mathbb{R}^{d_y}$. (A static input-output system is a special case with $T = 1$.) The agent not only controls the input sequence, but can also optionally apply a perturbation or intervention δ that modifies the system parameters. It can also optionally specify the initial condition of the system state ι (which is just a special case of the input x_0). We define an experiment as the tuple $a = (\iota, \delta, x_{1:T})$. (Note that this is an open-loop control setting, and for some benchmarks, we only specify ι and/or δ, and omit the inputs $x_{1:T}$.)

Experimental protocol. The agent is presented with some background context C, and an optional initial dataset $\mathcal{D}_0 = \{(a_0^i, y_{0,1:T}^i) : i = 1 : N_0\}$, where each sample is drawn from the system using $y_{0,1:T}^i \sim p^*(\cdot|\mathrm{do}(a_0^i))$. We assume the initial designs a_0^i are from the default (unperturbed or "wild-type") system, but they may use different input sequences x^i. The agent is then given a budget of B turns to interact with the system. At each step, it designs an experiment $a_r = (\iota_r, \delta_r, x_{r,1:T})$, and then collects data $y_{r,1:T}$ from the environment, to create $\mathcal{D}_r = (a_r, y_{r,1:T})$, and writes $\mathcal{D}_{0:r} = \mathcal{D}_0 \cup \{\mathcal{D}_\rho\}_{\rho=1}^r$ for the accumulated dataset. It can use this knowledge to update its beliefs about the underlying model, $p_r = p(m|C, \mathcal{D}_{0:r})$, and this belief can be used to design the next experiment, and to make predictions about the future. See Algorithm 1 for the full pseudocode.

Evaluation. After spending the budget of B experiments, each agent has the training set $\mathcal{D}_{\text{tr}} = \mathcal{D}_{0:B}$, and belief state, p_B. When working with synthetically generated data from a known function, we can compare the agent's estimated model directly with the true model using a novel metric we develop based on SymPy (see Section A.2); we call the fraction of successful matches the *recovery rate*. However, since most models are not simple functions, and since the "true model" is not even available for real-world domains, our main evaluation metric is the *out-of-sample interventional prediction accuracy*. To evaluate this, we draw held-out test experiments $a \sim \mathcal{Q}$ from a *query distribution* $\mathcal{Q}$ (disjoint from the experiments the agent ran) and ask the agent to predict a *target functional* $F(y)$ of the outcome — the quantity the task actually cares about. (In the simplest case, $F = \mathrm{id}$, i.e. the agent must predict $y_{1:T}$ itself, but in NEURONBENCH, we use a domain-specific target $F(y_{1:T})$ that summarizes the entire trajectory into a set of meaningful summary statistics, such as the number of neuron spikes.) Our primary loss metric is the normalized Mean Squared

[2]Available at https://github.com/murphyk/neuronbench

Error

$$\mathcal{L}_{\text{nMSE}}(F^\star, \hat{F}) = \frac{\mathbb{E}_{a\sim\mathcal{Q}}\big[(\hat{F}(a) - F^\star(a))^2\big]}{\text{Var}_{a\sim\mathcal{Q}}\big[F^\star(a)\big]}, \tag{1}$$

where $F^\star(a) = \mathbb{E}_{y\sim p^*(\cdot|a)}[F(y)]$ is the expected target under the true distribution $p^\star$, and $\hat{F}(a)$ is the agent's prediction (see Eq. (5) for details). (For a vector-valued target, Eq. (1) is applied *per component* (each summary standardized by its own query-distribution variance) and averaged over components.) Crucially, the query distribution $\mathcal{Q}$ perturbs the system in various ways, so we are testing prediction performance under distribution shift. In (Richens & Everitt, 2024) they prove that doing well on this metric implies the agent must have functionally learned a causal world model, even if we do not directly inspect the learned model.

3 METHODS

Overview. The MDA method is visualized in Fig. 5; see Algorithm 2 for detailed pseudocode. At each step, the agent updates its belief state $p_r = p(m|\mathcal{D}_{0:r})$, which is a posterior distribution over models or hypotheses m. Then it chooses the next experiment by maximizing the expected value of information, $a_{r+1} = \arg\max_{a\in\mathcal{A}} \text{VoI}(a)$. It runs the experiment and updates its dataset by appending $\mathcal{D}_{r+1}$. After B rounds, the agent is asked to forecast the outcomes to some novel experimental conditions. We give the details below.

Sequential Bayesian inference. The belief state $p_r = p(m|\mathcal{D}_{0:r})$ is a posterior over models m, represented as a set of N_m particles, which are updated using Sequential Monte Carlo or SMC (see e.g., (Naesseth et al., 2019; Chopin & Papaspiliopoulos, 2020)). At each step r, we propose a set of new particles using an LLM-based proposal distribution $p(m_r|\{m^n_{r-1}\}, \mathcal{C}, \mathcal{D}_{0:r})$. Crucially this conditions on the whole set of previous particles, as in SMC-S (Piriyakulkij et al., 2024), rather than a single ancestor as in ModelSMC (Wahl et al., 2026). The LLM gets to see the previous hypotheses and the errors that they made, and can suggest new hypotheses from the current hypothesis class.[3]

Computing the marginal likelihood (evidence). After proposing a new model (particle), we evaluate its evidence (marginal likelihood), $Z_m = p(\mathcal{D}_{0:r}|m) = \int p(\mathcal{D}_{0:r}|m,\theta)\, p(\theta|m)\, d\theta$, using the adaptive-tempered SMC method shown in Algorithm 5, which uses random walk Metropolis rejuvenation moves. Crucially, the integration over model parameters provides an automatic Occam penalty factor for complex models with many parameters (MacKay, 1991). Thus, over the course of inference, we will get a set of hypotheses that tradeoff complexity with model fit, as shown in Fig. 3b.

Dealing with intractable likelihoods. For latent variable models, computing the likelihood, $p(\mathcal{D}_{0:r}|m,\theta) = \prod_{i\in\mathcal{D}_{0:r}} p(y^i_{1:T}|a^i, m, \theta)$, may be intractable. For this we use a third layer of SMC, namely the bootstrap particle filter method Algorithm 4 to approximate $p(y_{1:T} \mid z_0, \delta, x_{1:T}, m, \theta) = \int \prod_{t=1}^T p(y_t \mid z_t, m, \theta)\, p(z_t \mid z_{t-1}, x_t, m; \text{do}(\delta, \theta))\, dz_{1:T}$. We call the resulting method the SMC^3 algorithm, since it is an extension of the SMC^2 algorithm of (Chopin et al., 2013) which did not perform inference over models (just parameters and latents).

Expanding and shrinking the hypothesis space. SMC can move probability mass between different models in its current hypothesis class. However, in the $\mathcal{M}$-open case (Bernardo & Smith, 1994; Kelter, 2020), we assume our initial hypothesis class is insufficient to capture the truth. Thus we need a way to expand the hypothesis space if the current space is inadequate. To do this, we

[3] We use Opus-4.7 for all the LLM components in this paper, both for MDA and the baselines. In some preliminary results (not shown here) we observed that MDA also works fairly well with Deepseek-v4-pro. However, if the LLM is too weak, the proposal will never be able to sample the right model, so MDA could fail to converge on the truth. In principle we can augment the LLM proposal with random sampling, which would give it full support over the hypothesis space and hence make the method consistent in the limit of infinite compute; but in practice random sampling will not be able to generate the truth either (assuming a sufficiently rich hypothesis space). Fortunately, we do not require the proposal to generate the truth in one shot; instead, by conditioning on previous hypotheses, $\{m^n_{r-1}\}$, it can suggest local mutations (as is standard in LLM-powered code optimization/ generation), and thus can gradually converge on the truth.

compute a *predictive check*, i.e., we evaluate the performance of the current best model on a novel (input,output) pair (c.f., (Kelter, 2020)), and/or check the fit on all the currently collected data. If the error is too large, MDA *expands* the hypothesis space by prompting the LLM to suggest a novel unnamed mechanism. (This is analogous to the Breaker–Builder method of (Buehler, 2026).) Conversely, if the posterior has confidently identified a model that fits well, we reduce the number of hypotheses, to prevent a proliferation of near duplicates, which diminishes performance. See Section A.3 for more details on MDA's meta-controller.

Experiment design. We choose the experiment whose outcome is most informative about which hypothesis is true: $a^\star = \arg\max_{a \in \mathcal{A}} I(M; Y_a \mid \mathcal{D})$ (Lindley, 1956; Box & Hill, 1967; Chaloner & Verdinelli, 1995; Rainforth et al., 2024). This is called the Expected Information Gain (EIG). We also consider a more general task-driven Value of Information (VoI) objective, following (Rainforth et al., 2024; Smith et al., 2023). We derive a simple, closed-form Gaussian approximation for these objectives, shown in Eq. (34) and Eq. (43). The resulting objective prefers the design which maximally distinguishes the models in terms of their predictive performance. We can optimize this objective using a black-box optimization algorithm, such as CMA-ES (Hansen, 2016) for continuous design spaces, or LLM-based optimizers such as Funsearch (Romera-Paredes et al., 2024) for discrete design spaces.[4]

Prediction. After each step, we convert the belief state, which is a distribution over models, $p_r = p(m|\mathcal{D}_{0:r})$, to a predictive distribution in observation space, $\rho_r(a) = p(Y|a, \mathcal{D}_{0:r})$, which is computed by Bayes model averaging (c.f., (Self & Cheeseman, 1987)). From this, we derive the posterior predicted mean of the target, which is optimal under the ℓ_2 loss we use in Eq. (1):

$$\hat{F}_r(a) = \mathbb{E}[F(Y)|a, \mathcal{D}_{0:r}] = \sum_m \int \Big[\int F(y)\, p(y|m, \theta, a)\, dy \Big]\, p(m, \theta|\mathcal{D}_{0:r})\, d\theta \tag{2}$$

We then plot a learning curve of hold-out loss vs number of experiments, and compare the result to the relevant SOTA method for each of the benchmarks we compare to. As an additional control, we pass the *same data* (collected by MDA) to an in-context learner (ICL), based on the same LLM as used by MDA, and ask it to predict the outcome without using an explicit model. (This is known as transduction, and on some problem domains, beats the inductive (model-based) approach that we use (Li et al., 2024b), thus serving as a reasonable baseline.)

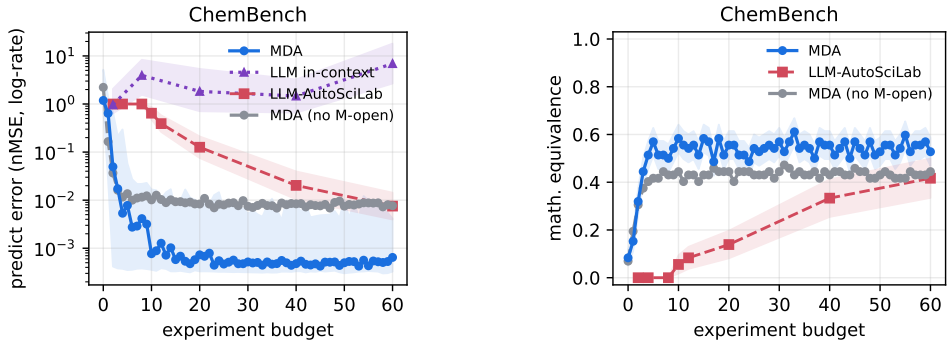
4 Experimental results

In this section, we summarize some of our experimental results on benchmarks from physics (ForceBench), chemistry (ChemBench), biology (NeuronBench), and the BoxingGym suite. We show that the MDA method reduces held-out interventional predictive error, and recovers the true mechanism, much faster (in terms of number of experiments) than the per-benchmark baseline. We give more details in the appendices.

4.1 ChemBench: discovering enzyme-kinetic rate laws

Benchmark. In this section, we briefly describe ChemBench, which is our wrapper on top of ActiveSciBench-Chem from (Kabra et al., 2026). (We don't change the underlying benchmark, just the interface, to make it compatible with our other benchmarks.) The problem is to learn a function mapping seven controllable inputs (substrate, inhibitor, second substrate and product concentrations, enzyme loading, temperature and pH) to a reaction rate r: $r = f\big(C_A, C_I, C_B, C_P, \mathrm{Enz}, T, \mathrm{pH};\, \theta\big)$. See Table 1 for some examples. The experimenter gets to set the 7 input variables, and observes the scalar response (which is the initial reaction rate, not a full trajectory). Every run is seeded with a fixed passive set $\mathcal{D}_0$ of 3 law-agnostic conditions (a substrate-concentration sweep $C_A \in \{0.03, 0.3, 3\}$ with the other inputs at defaults, the shape axis on which the mechanisms most disagree), after which the agent *designs* its experiments over the continuous 7-D input box. Following the paper, we measure performance on a query set $\mathcal{Q}$ of fresh random input conditions drawn from that box (the benchmark's N=1000 novel held-out points), disjoint from the

[4] In this paper, the discrete design spaces are sufficiently small that we can optimize the objective exactly by enumeration.



(a) Data efficiency (held-out log-rate nMSE).

(b) Mathematical equivalence.

Figure 1: **ChemBench (enzyme-kinetic rate laws).** MDA (blue) vs. the LLM-AutoSciLab baseline, a model-free in-context (ICL) forecaster, and an ablation of MDA *without* $\mathcal{M}$-open exploration (grey, "no M-open"). (a): held-out interventional prediction error (nMSE in $\log(1+\text{rate})$ space, matching the benchmark's RMSLE scale) vs. experiment budget. (b): mathematical equivalence (fraction of rate laws recovered exactly, via SymPy) vs. budget. Lines are means over the 36-task subset (12 domains $\times$ 3 tiers); shaded bands are ± 1 standard error *across those tasks* (over 36 tasks $\times$ 2 seeds for MDA, 36 tasks for LLM-AutoSciLab). Opus-4.7.

experiments run, using the held-out root-mean-squared log-error (RMSLE) defined in Eq. (49); this is a monotonic transform of the nMSE metric we use on the other benchmarks. We also compute the fraction of runs that yield a function that is mathematically equivalent to the truth, as estimated using sympy (see Eq. (52)).

Baseline. The baseline agent from the paper (Kabra et al., 2026), which they call LLM-AutoSciLab, is an LLM-driven agent that also maintains multiple hypotheses, and selects experiments to disambiguate between them, fitting each candidate's functional form by symbolic regression (PySR). However, it is a heuristic approach, and does not use any formal Bayesian machinery.

MDA. The MDA agent assumes the unknown function f can be represented as an algebraic equation, and asks the LLM to propose various candidates from a *generic* prompt that names no mechanism families or mechanism-specific heuristics — in fact *less* scaffolding than the LLM-AutoSciLab baseline, whose prompt lists mechanism example forms (Michaelis–Menten, Hill, Arrhenius) as templates (see Section H for both, side by side). It assumes a Gaussian likelihood with multiplicative noise, $p(y|a, f, \theta) = \mathcal{N}\big(y \mid f(a, \theta), \sigma_{\text{rel}}\big)$, where $\sigma_{\text{rel}} = \sigma f(a, \theta)$ is multiplicative noise. Given this model, the agent does posterior inference over f using SMC, and experiment design using the method discussed below. MDA uses EIG to design experiments. To maximise the objective over the continuous 7-D design box (log-uniform on the concentration/enzyme axes, uniform on T/pH), we use the CMA-ES algorithm from (Hansen, 2016). (However, preliminary results suggest that random sampling also works well on this problem domain.)

Data efficiency experiments. In Fig. 1, we show the performance of various agents vs number of experiments. Following their experimental protocol, we use a stratified 36-task subset (12 domains $\times$ easy/medium/hard) at two seeds. In Fig. 1a, the metric is the held-out nMSE (in log-rate space) averaged over trials and over tasks. In Fig. 1b, the metric is the fraction of rate laws that are mathematically equivalent to the truth. We see that MDA (blue) is much more data efficient than the LLM-AutoSciLab baseline (red). Dropping $\mathcal{M}$-open exploration (grey) degrades *both* exact recovery and interventional prediction toward the LLM-AutoSciLab level, which validates the $\mathcal{M}$-expansion component of MDA. Furthermore, we see that prediction error tracks form recovery, which validates our prediction-focused evaluation framework. Finally, we see that ICL is unable to predict reliably on this problem, even when given the same data as MDA, validating our explicit model-based approach to prediction. See Section B.4 for more detailed results.

Qualitative results. Table 1 shows the laws recovered on three representative domains. MDA returns *interpretable mechanisms* — exactly the true form for substrate inhibition, and the correct inhibition/saturation structure elsewhere (although sometimes with a spurious extra factor). LLM-

Domain	True law	MDA recovers	LLM-AUTOSCILAB recovers
substrate inhib.	$\frac{k\,\mathrm{Enz}\,C_A}{K_m+C_A+C_A^2/K_i}$	*same form* ✓	$\mathrm{Enz}\,(a\,r^{C_A}+\dots)$ ×
noncompetitive	$\frac{k\,\mathrm{Enz}\,C_A}{(1+C_I/K_i)(K_m+C_A)}$	Hill×noncomp. ≈	$10^{\,0.87\log(0.5\sqrt{\mathrm{Enz}/(C_I+\cdot)})}$ ×
Michaelis–Menten	$\frac{k\,\mathrm{Enz}\,C_A}{K_m+C_A}$	+ spurious $e^{-E_a/RT}$ ≈	$10^{\,0.43\log(\mathrm{Enz}\,T^{0.37}/\cdot)}$ ×

Table 1: **Representative recovered laws** (best config, B=60). Parenthesised value is held-out RMSLE; ✓ = symbolic form recovered, ≈ = correct structure with a spurious extra factor, × = mechanistically wrong. MDA recovers the mechanism in every case; LLM-AUTOSCILAB's PySR fits the numbers with unphysical expressions, which get low RMSLE but are symbolically meaningless.

AUTOSCILAB's PySR instead returns numerically-fit but mechanistically *meaningless* expressions — nested $10^{\,a\log(\cdot)}$ and stretched-exponential forms — that can score a low RMSLE while being symbolically wrong.

4.2 FORCEBENCH: DISCOVERING FORCE LAWS

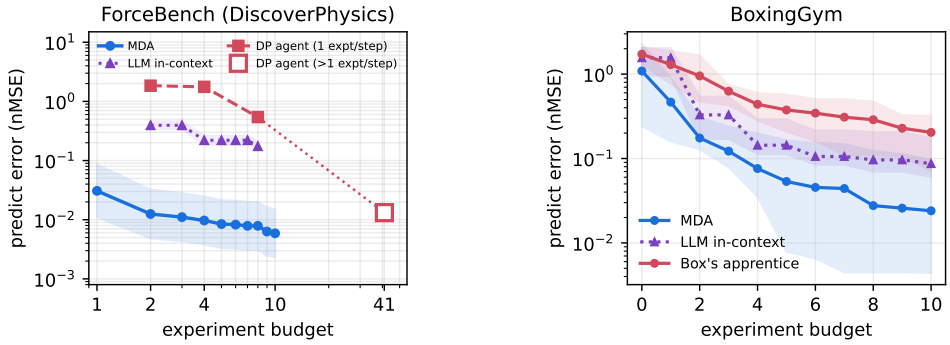
Benchmark. In this section, we give a brief description of FORCEBENCH, which is our wrapper on top of the DISCOVERPHYSICS benchmark from (Wiemann et al., 2026). (We do not change the underlying benchmark, merely the interface, to make it compatible with our other benchmarks.) FORCEBENCH requires an agent to infer an unknown but novel force law governing the behavior of two or more particles in a 2d space. The agent can control the initial location and velocity of one of the particles, as it is launched, as well as a few other environment parameters. (In practice we discretize the design space into a fixed menu of 13 different combinations, listed in Table 4.) Following the paper, the performance of the learned model is assessed on a test set which probes the model's predictive performance in novel experimental settings beyond the training set. We report this in terms of the normalized MSE (nMSE = MSE/test-trajectory variance). Training observations carry Gaussian noise of σ=0.05 (5% of the test-particle trajectory variance), matching §3.2 of Wiemann et al. (2026); the held-out test trajectories used at evaluation time are noise-free.

Baseline. The baseline agent from the paper (Wiemann et al., 2026), which we call DP-AGENT, is a pure LLM approach, that asks the LLM to propose an experiment at each step, and — at the end — to generate a model in the form of some Python code. The parameters of this model are then fit to the observed data by the environment, before it is evaluated.

MDA. The MDA agent assumes the unknown force can be represented as a Green's function F, and asks the LLM to propose various candidates using the same generic, domain-agnostic prompt as the DP-AGENT baseline (see Section H for the prompts, shown side by side with the baseline's). It then derives the acceleration using Newton's law, and then solves the resulting deterministic ODE using diffrax. The agent assumes the observations are just a noisy version of the true trajectory, and hence uses the tractable Gaussian likelihood in Eq. (14). It then does posterior inference over F (and its parameters θ) following the MDA recipe, and uses the posterior at each step to design experiments. Finally, it submits its best estimate of the model $\hat{F}$ to the environment, to get scored in the same way as the baseline.

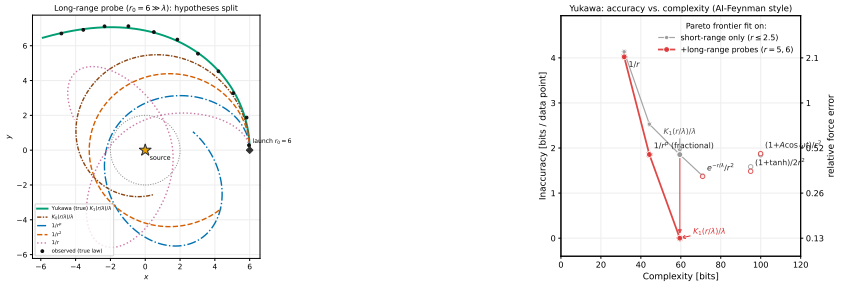
Data efficiency experiments. In Fig. 2a, we show the performance of 3 agents vs number of experiments: MDA, the baseline Discover-Physics (DP) agent, and the ICL baseline. For each of the 6 worlds, we sample 3 random initial conditions, and roll out 3 trajectories per IC. The metric is mean nMSE averaged over trials and all six of the two-particle worlds. (See Fig. 8 and Fig. 9 for detailed per-world performance plots.) All agents use the same design space, and use Opus 4.7 (the best model reported in (Wiemann et al., 2026)). We see that MDA is substantially more data efficient. See Table 7 for a list of the laws discovered by each agent after $B = 8$ experiments.

Example: Yukawa world. In this section, we study the YUKAWA world in more detail. It has a force law of the form $F = q_i q_j K_1(r/\lambda)/\lambda$, where K_1 is the modified Bessel function and $\lambda = 2$.



(a) FORCEBENCH (force laws). (b) BOXINGGYM (regression).

Figure 2: **Data efficiency on dynamical and regression domains.** Held-out interventional prediction error (nMSE) vs. experiment budget. (a): FORCEBENCH— MDA (blue, best-so-far) vs. the Discover-Physics agent — throttled to one experiment per round, and extrapolated to its native *batched* protocol (open square, $\sim$41 experiments) — and a model-free ICL forecaster; log–log axes. (b): BOXINGGYM (four regression problems) — MDA (best-so-far) vs. Box's Apprentice and a model-free ICL forecaster, out to B=10. Bands are mean $\pm$ SE for FORCEBENCH, and median over domains $\pm$ IQR for BOXINGGYM (robust to a single catastrophic domain). Opus-4.7.



(a) Trajectories under a long-range probe. (b) Pareto curve.

Figure 3: **YUKAWA world.** (a) Predicted trajectories under different hypotheses when the probe is launched with $r_0 = 6$. The Yukawa model (green) matches the empirical data. (b) Accuracy-complexity Pareto frontier for models discovered by MDA in the YUKAWA physics environment. (Figure based on (Udrescu & Tegmark, 2020, Fig 1)).

This means the force is very hard to distinguish from a power law when $r \leq \lambda$; but if the agent chooses the target particle's launch radius $r_0 > \lambda$, it can detect the difference between hypotheses, as shown in Fig. 3a. In Fig. 3b we plot a Pareto curve, showing the error vs complexity for different hypotheses before (gray) and after (red) the critical long-range experiment. The error is measured in bits, and is computed from the relative error in the estimated force: $\left|\log_2(F_{\text{pred}}/F_{\text{true}})\right|$. The complexity is also measured in bits, $-\log_2 p(m \mid \mathcal{D})$, following the minimum description length (MDL) principle. (In contrast to a syntactic complexity metric such as the Halstead metric of (Kasenberg et al., 2026), MDL rewards a law only insofar as the evidence supports it.) We see that after the critical experiment, the true model, $K_1(r/\lambda)/\lambda$, jumps out, since it has 0 error and relatively small complexity — a true "aha" moment for the agent.

4.3 BOXINGGYM

Benchmark. In this section we give a brief summary of the BOXINGGYM benchmark from (Gandhi et al., 2025). The benchmark implements ten scientific domains as generative probabilistic models, represented in the PYMC probabilistic programming language. An agent interactively chooses designs (for up to 10 steps), observes outcomes, and is scored by its predictive performance. We exclude three of their domains — EMOTION and MORAL-MACHINES which use an LLM as part of the "true model", and PREDATOR-PREY, which is very similar to our physics benchmark — leaving us with seven. We partition these seven domains into two clusters. The first cluster consists of standard GLMs, where the agent needs to learn the form of the function that defines the mean of the

(scalar) output, analogous to learning the symbolic laws in the physics and chemistry domains. The second cluster consists of latent variable models, where the number of parameters can grow with the size of the data. We evaluate predictive performance on a held-out set, derived from novel inputs, and report the nMSE. (Some other metrics are discussed in Section D.1.)

Baseline. The agent used in the BOXINGGYM paper (Gandhi et al., 2025) is called BOX'S APPRENTICE, and is based on the method from (Li et al., 2024a). At each step, this prompts the LLM to synthesize a single model $\hat{m}$ (represented as a PYMC program) given the data so far, fits its parameters to data, injects $\hat{m}$ (with a residual critique) into the LLM's context, and asks the LLM to choose the next experiment directly ("where should we observe next to best improve the model?"). At the end, it uses the generated model $\hat{m}$ to make predictions, similar to MDA.

MDA. The MDA agent assumes the mean of the GLM's output can be represented as some form of symbolic expression, and asks the LLM to propose various candidates (see Section H for details of the prompt). For the latent variable models, the MDA agent asks the LLM to generate NUMPYRO code, which can be used to define a prior and likelihood, as needed by SMC. (We used NUMPYRO instead of PYMC because we found it to be faster and more numerically robust.)

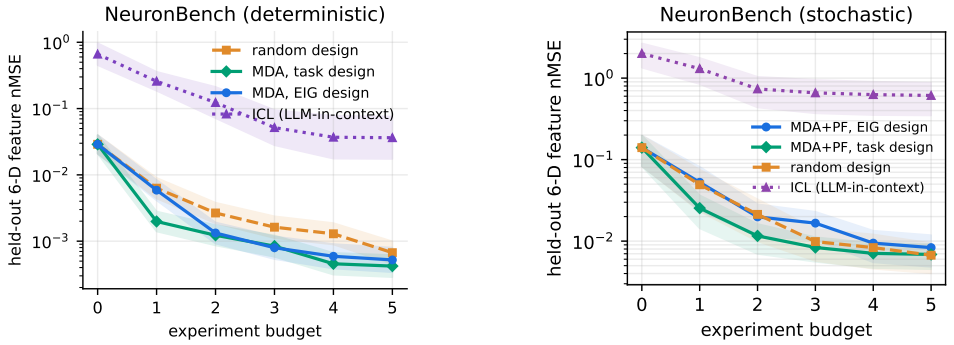
Data efficiency experiments. In Fig. 2b, we show results on the GLM domains. (For results on the latent variable models, see Section D.) We see that MDA is substantially more data efficient than Apprentice. Surprisingly, the ICL baseline also beats the Apprentice. However, the error bars are large, since these are all small domains. (See Fig. 16 for performance plots on each world separately.)

4.4 NEURONBENCH: DISCOVERING ION-CHANNEL MECHANISMS

Benchmark. We design a new benchmark, NEURONBENCH, by creating 6 "mystery neurons", based on the generalized Hodgkin-Huxley (HH) model, which are a set of nonlinear ODEs for describing the spiking behavior of neurons (see Section E.1 for details). Each mystery neuron is designed to use a novel membrane mechanism, so the LLM cannot rely just on its memory of textbooks it has read. The experimental protocol allows the agent to specify the input signal (an electrical current) over time; we give it a menu of 9 templated signal shapes. (It can optionally apply 3 different kinds of ion channel blockers, but these have no effect on the current benchmark, so we exclude them from the design space.) We also create a stochastic version of the benchmark, by adding finite-channel gating noise, turning the true model into an SDE. We call this benchmark NEURONBENCHSTOCH. Since predicting the exact voltage trace $y_{1:T}$ is very hard (especially for the stochastic case), we only ask agents to predict a set of 6 features derived from the trajectory, defined in Eq. (67). The target vector $F(y_{1:T})$ contains things like the number of spikes in the trace in the pre and post phase. (These are the most biologically salient characteristics of the signal.) We compute the nMSE on this target vector as in Eq. (1) under a set of novel perturbations to the system.

MDA. The MDA agent is told to model the data using "a single-compartment conductance-based (Hodgkin-Huxley) neuron with voltage-gated channels, but with a potentially novel mechanism". The LLM proposes various candidates (see Section H for details of the prompt) and then converts this into an ODE or SDE. For the ODE case, the latent dynamics are deterministic, so the likelihood $p(y_{1:T}|a, m, \theta)$ is tractable. For the SDE case, the latent dynamics are stochastic, so MDA uses the particle filter to approximate the likelihood. We consider choosing experiments to either maximize the EIG, or to maximize the VoI where the utility function is defined in terms of task-specific prediction loss of the target, $F(y_{1:T})$. (See Section A.9 for a detailed discussion.) We also compare to a random design.

Data efficiency experiments. In Fig. 4 we show prediction error (in terms of the 6d feature vector) vs number of experiments for the 3 MDA variants and an ICL baseline. (Since this is a new benchmark, there is no previous SOTA to compare to.) We see that MDA is much better than ICL. In the deterministic case, we see that the EIG design is best, whereas in the stochastic case, the task-driven VoI design is slightly better.



(a) Deterministic NEURONBENCH. (b) Stochastic NEURONBENCHSTOCH (full SMC^3).

Figure 4: **Data efficiency on NEURONBENCH.** Held-out *6-D feature-forecast* error (nMSE, the $F(y)$ vector of Eq. (67)) vs. experiment budget. We show mean performance $\pm$ SE. All methods use Opus-4.7.

5 RELATED WORK[5]

- **Causal models for interventional prediction.** Predicting "what if" questions using causal models is discussed at length in Pearl (2009). Recently Richens & Everitt (2024) proved that an agent that can robustly predict across a full range of interventions (distribution shifts) must have implicitly learned a causal world model. We instead explicitly represent the causal model, so that we can leverage prior knowledge from LLMs (Kıcıman et al., 2024; Ban et al., 2025), reason over our uncertainty using Bayesian methods, and provide an *interpretable model* to the user.
- **Bayesian experimental design.** Choosing the most informative experiment is the classical model-discrimination objective of (Lindley, 1956), reviewed in (Chaloner & Verdinelli, 1995; Rainforth et al., 2024; Huan et al., 2024). Recently (Choudhury et al., 2026) proposed to combine BED with LLMs, but our approach is very different, since we use the LLM to propose an explicit probabilistic model, which we can evaluate using standard Bayesian machinery, whereas they perform all probability calculations implicitly using the LLM itself.
- **LLMs for scientific discovery.** LLMs have been used to fit scientific models to static datasets, often augmented with literature review, using blackbox optimization (Romera-Paredes et al., 2024; Wahl et al., 2026; Kasenberg et al., 2026; Aygün et al., 2026; Gottweis et al., 2026; Xie & Wilson, 2026; Song et al., 2025). In addition there is some work on actively collecting datasets for model fitting using agent-designed experiments (Piriyakulkij et al., 2024; Huang et al., 2025; Abhyankar et al., 2026; Elteto et al., 2026; Prystawski et al., 2026; Jagadish et al., 2026; Bisht et al., 2026; Fu et al., 2026; Ghareeb et al., 2026). Our work is in the latter camp, differing mainly in how we handle the $\mathcal{M}$-open regime inside SMC, the diversity of domains, and by the fact that we beat SOTA methods based on LLMs.
- **Benchmarks for interactive scientific discovery.** Various benchmarks evaluate agents that learn scientific laws by *interactive experimentation*: we build on DISCOVERPHYSICS (Wiemann et al., 2026), ACTIVESCIBENCH-CHEM (Kabra et al., 2026), and BOXINGGYM (Gandhi et al., 2025), and add our own NEURONBENCH. Other relevant benchmarks include NEWTONBENCH (Zheng et al., 2026), SCIENCE-GYM (Cerrato et al., 2026), and SCIGYM (Duan et al., 2025).

6 DISCUSSION

We have shown how to combine LLMs with SMC^3 and BED to learn mechanistic world models in a data efficient manner. The framework has two main limitations. In the $\mathcal{M}$-open regime, recovery is bottlenecked by whether the LLM proposes a form that covers the truth; and the nested SMC^3 is compute-intensive, with cost growing as $O(B^2 N_m)$ from full-batch re-fitting, which caps the practical experiment budget on the stochastic benchmark. Future work includes addressing these limitations by expanding the set of methods for creating new hypotheses, and amortizing the inference using SBI methods (e.g., (Cranmer et al., 2020; Radev et al., 2022; Deistler et al., 2025)).

[5]For more related work, see Section G.

A METHOD: FURTHER DETAILS

A.1 THE AGENT-ENVIRONMENT INTERFACE

Algorithm 1 sketches the agent/environment interface at a high level, in the style of BOXINGGYM's Fig. 2 (Gandhi et al., 2025) but in our notation. We make several changes: (i) We just evaluate using prediction accuracy, and drop their explanation metric and EIG metric, for simplicity and to be consistent with the rest of the paper; (ii) we evaluate after every step, to get a learning curve, rather than a single number; (iii) we add an update-state method, since MDA sequentially updates its belief state.

Algorithm 1 The agent/environment interface (after BOXINGGYM Fig. 2 (Gandhi et al., 2025), in our notation). After each experiment we call L-PRED, which forecasts the held-out query outcomes and scores them against the truth, yielding a learning curve $\ell_{\text{pred}}(B)$ of held-out prediction error.

env $\leftarrow$ ENV.INIT()
$C \leftarrow$ env.DESCRIBE()
$\mathcal{D}_0 \leftarrow$ env.INIT-DATA()
$F \leftarrow$ env.TARGET() ▷ the evaluation functional; Eq. (3)
agent $\leftarrow$ AGENT.INIT(LLM, $C, \mathcal{D}_0, F$)
$\ell_{\text{pred}}(0) \leftarrow$ env.EVAL-PREDICTOR(agent.GET-PREDICTOR()) ▷ hold-out predictive loss
$\ell_{\text{model}}(0) \leftarrow$ env.EVAL-MODEL(agent.GET-MODEL()) ▷ structural recovery of $\hat{m}$
for $r = 1 \dots B$ **do**
  $a_r \leftarrow$ agent.DESIGN()
  obs$_r \leftarrow$ env.STEP(a_r)
  agent.UPDATE-STATE(a_r, obs$_r$)
  $\ell_{\text{pred}}(r) \leftarrow$ env.EVAL-PREDICTOR(agent.GET-PREDICTOR()) ▷ Algorithm 2
  $\ell_{\text{model}}(r) \leftarrow$ env.EVAL-MODEL(agent.GET-MODEL()) ▷ recovery of the best-so-far $\hat{m}$
end for

function EVAL-PREDICTOR($\hat{F}$) ▷ score a predictor on the held-out query set $\mathcal{Q}$
  return $\frac{1}{|\mathcal{Q}|} \sum_{a_q \in \mathcal{Q}} \text{LOSS}\big(F(m^\star(a_q)),\ \hat{F}(a_q)\big)$ ▷ $m^\star$ is the true model
end function

A.2 EVALUATION

Given $\mathcal{D}$, the agent is evaluated using

$$\mathcal{L} = \mathbb{E}_{a \sim \mathcal{Q}}\, \mathbb{E}_{y \sim p^*(\cdot|a)} \big[\, \ell\big(F(y),\ \hat{F}(a)\big)\, \big] \tag{3}$$

where the held-out test experiments are drawn from the *query distribution* $a \sim \mathcal{Q}$, disjoint from the agent's own designs, p^* is the true system and $\hat{F}(a)$ is the agent's Bayes forecast. If we use squared error and assume $F(y) = y$, this becomes

$$\mathcal{L} = \mathbb{E}[(y - \hat{F}(a))^2] \tag{4}$$

In this case, the optimal estimate is the posterior mean

$$\hat{F}(a) = \mathbb{E}_{p(m,\theta|\mathcal{D})}\big[Y \mid a\big], \tag{5}$$

For binary targets, where $F(y) = y \in \{0, 1\}$, the optimal estimate is $\hat{F}(a) = P(Y{=}1 \mid a) = p$, so the loss becomes the Brier score:

$$\mathcal{L} = \mathbb{E}[(y - p)^2] \tag{6}$$

For time series, we average the per-step loss across time:

$$\mathcal{L} = \mathbb{E}[\frac{1}{T} \sum_{t=1}^{T} (y_t - \hat{y}_t)^2] \tag{7}$$

Different flavours of predictors. The predictive loss Eq. (3) is evaluated by EVAL-PREDICTOR on the predictor $\hat{F}$ = GET-PREDICTOR(flavour) of Algorithm 2. For the regression benchmark (BOXINGGYM) we use the BMA flavour, the full posterior-predictive $\mathbb{E}_{p(m,\theta|\mathcal{D})}[F(Y) \mid a]$. For the NEURONBENCH and CHEMBENCH we use the MAP flavour: we predict by plugging in the single Occam-MAP structure $\hat{m}$ *and* its MAP parameters $\hat{\theta}$. FORCEBENCH is a special case — its observable is a trajectory the environment integrates — so EVAL-PREDICTOR obtains $\hat{F}(a_q)$ by rolling out the submitted $(\hat{m}, \hat{\theta})$ ODE (whereas the DP-AGENT baseline submits only $\hat{m}$ and lets the environment fit $\hat{\theta}$).[6]

Structural recovery. Prediction asks whether the agent *forecasts* well; *recovery* asks the arguably more interesting question of whether it found the right *form*. We score recovery with a single, domain-general EVAL-MODEL test shared by CHEMBENCH and FORCEBENCH: parse the discovered symbolic law $\hat{m}$ into canonical form using SymPy, refit its free constants using least squares, then compare its predictions to the ground-truth *function* on a fixed grid of points. We declare the two functions *mathematically equivalent* if the log-scale RMSLE is below a small tolerance (0.02 for CHEMBENCH, Eq. (52); 0.05 for FORCEBENCH).[7]

The two benchmarks differ only in how the ground-truth function is sourced: CHEMBENCH's oracle *exposes* its true rate law as part of the environment, whereas FORCEBENCH does not. Instead FORCEBENCH specifies each world's force through a field PDE (Poisson, Helmholtz, fractional Laplacian, Kaluza–Klein) with no exposed $F(r)$. So we *measure* the true force numerically — release a unit probe at rest and read its initial acceleration — which needs no hardcoded closed form, and matches the exotic non-power-law forces (Yukawa's screening, the Kaluza–Klein force) exactly.[8] Because the metric probes $F(r)$ from the discovered model rather than a symbolic string, it applies unchanged to the external DP-AGENT baseline: we run *its* fitted `discovered_law()` through the same release-at-rest procedure and compare the resulting $F(r)$ up to a coupling scale. This yields a like-for-like recovery head-to-head (Fig. 10), which is more informative than either method's absolute rate: MDA recovers the force law more often than the trajectory-fitting baseline on the clean radial worlds, while both fail on the screened (Yukawa) and higher-dimensional (Kaluza–Klein) forces.

For NEURONBENCH we use a different structural recovery metric: its structural label is a coarse hand-coded channel classification — inward/outward, depolarisation- vs. hyperpolarisation-activated, transient vs. persistent — read off the discovered channel's fitted reversal potential, activation slope, and inactivation. It therefore applies even to the *anonymous* channels $\mathcal{M}$-open discovery produces. However, it can only distinguish $\sim$8 classes (and "unknown" on degenerate parameters), so it is a coarse metric that we do not emphasize.

A.3 MAIN MDA LOOP

Overview. The MDA algorithm is shown schematically in Fig. 5. It is a sequential Bayesian experiment-design loop, where we use an LLM to propose model structures; but every numeric quantity — the parameter posterior, the marginal evidence, the model posterior $p(m \mid \mathcal{D})$, the VoI design score, and the forecast — is computed by Bayesian inference, never by the LLM. In a nutshell, MDA is a kind of *SMC sampler* (Del Moral et al., 2006) combined with a Value-of-Information Bayesian Experiment Design loop. (SMC is described in (Chopin & Papaspiliopoulos, 2020; Naesseth et al., 2019), and BED is described in (Rainforth et al., 2024).) See Algorithm 2 for the high level pseudocode; we will describe the individual functions below.

[6] We did a control where MDA also only submits $\hat{m}$ and lets the environment fit $\hat{\theta}$, to match the DP-AGENT protocol, but it did not make much difference to the results (details omitted for brevity).

[7] We refit rather than test symbolic equality because the discovered constants are learned from data, so they never match the true constants exactly — recovery is about the form *up to constants*, i.e., it asks the question "does some constant assignment make $\hat{m} \equiv m^\star$?", which is exactly the least-squares fit. Also note that SymPy equality is undecidable for the transcendental forms that appear (Bessel, exp).

[8] Comparing the force *function* on a grid rather than the rolled-out trajectory also makes the metric *decoupled from the dynamics*, so a world whose trajectory is chaotic or near-singular (e.g., the extra-dimension, ether and dark-matter worlds) can still register a clean recovery.

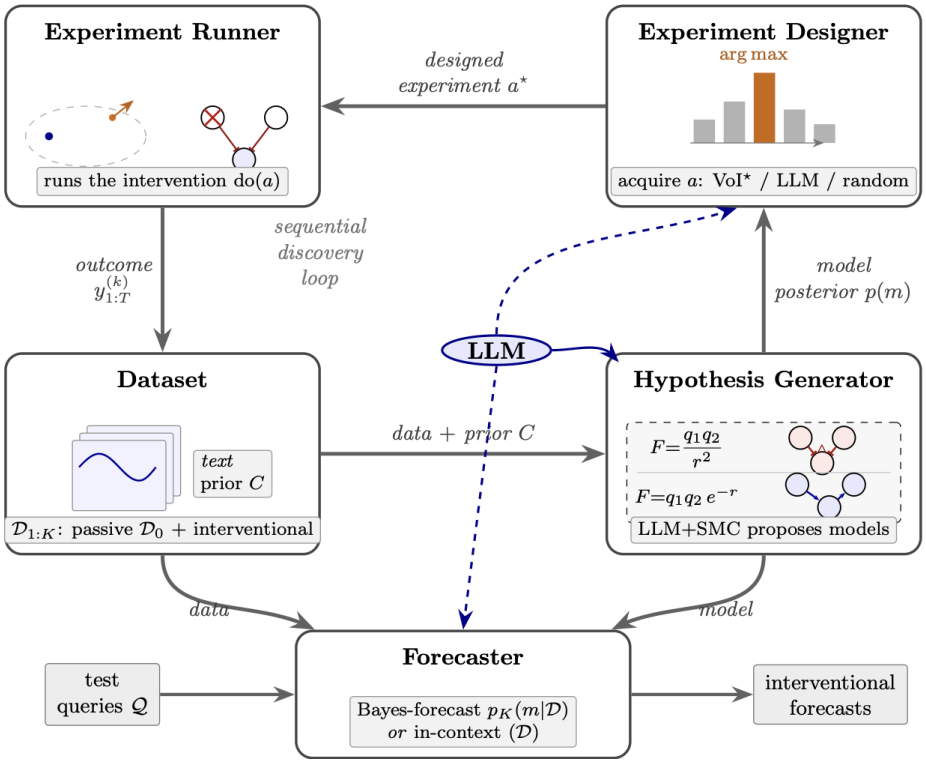


Figure 5: **The MDA discovery loop.** See text for details. (Based on (Elteto et al., 2026, Fig.2).)

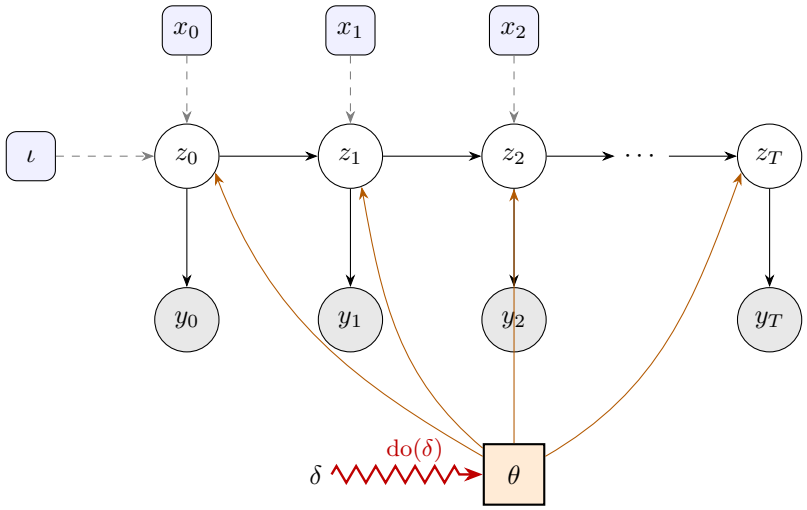


Figure 6: **The world as a controlled, intervenable state-space model** (Eq. (8)). A latent state z_t (white) evolves under the mechanism θ (orange; it parameterizes *every* transition) and emits a lossy, noisy observation y_t (grey) — in general only $y_{1:T}$ is seen. Optional exogenous inputs/covariates x_t (blue, dashed) and the initial condition ι (which sets z_0) are shifts in the *inputs* to a fixed mechanism. An intervention $\mathrm{do}(\delta)$ is categorically different: the lightning bolt strikes θ itself, changing the mechanism to θ'.

SSM. The agent assumes the unknown environment can be represented by a state space model, as shown in Fig. 6. This corresponds to the following probabilistic model m:

$$z_{t+1} \sim p\big(z_{t+1} \mid z_t,\, x_t; \mathrm{do}(\delta, \theta)\big), \qquad y_t \sim p\big(y_t \mid z_t;\, \theta\big), \qquad z_0 \sim p(z_0 \mid \iota). \tag{8}$$

where $\mathrm{do}(\delta, \theta)$ represents the parameters of the system after applying intervention δ, z_t are the hidden states, x_t are the optional inputs, and y_t are the observed outputs. (For simpler models, there might not be any latent variables — equivalent to $z = \text{const}$ — or there might not be any temporal

Algorithm 2 The MDA agent (member functions). Defines the agent methods invoked by the interface of Algorithm 1. The agent carries state across calls: the hypothesis space $\mathcal{M}$, the model posterior $p(m \mid \mathcal{D})$, the accumulated data $\mathcal{D}$, the observation model LOGLIK, the LLM (the structure proposer/verbaliser), and the pool size N_m. Config: $\mathcal{M}$-open error threshold τ_e, model-search rounds per step R_m (the inner loop of UPDATE-STATE; $R_m{=}1$ recovers a single gated expansion), and cumulative expansion cap $N_e^{\max}$ (counted by n_e); pool cap N_m (sub-routines cross-referenced inline). The *only* state carried sequentially across steps is the hypothesis space $\mathcal{M}_r$: given $\mathcal{M}_r$ and $\mathcal{D}_{0:r}$, the model posterior $p(m \mid \mathcal{D}_{0:r})$, its parameter posteriors, and the evidences $Z_m^{(r)}$ are all recomputed *full-batch*, so the belief is never a degradable sequential filter. Each step thus runs the model search to (gated) convergence on the current batch — the analog of batch-refitting $p(\theta \mid m, \mathcal{D}_{0:r})$ — before designing the next experiment.

function INIT(LLM, C, $\mathcal{D}_0$, F)
  store LLM, C, F; $\mathcal{D} \leftarrow \mathcal{D}_0$; $n_e \leftarrow 0$
  $\mathcal{M} \leftarrow$ INIT-HYP-SPACE(LLM, $C, \mathcal{D}_0$) ▷ initial pool (Algorithm 7)
  choose LOGLIK $\in$ {LOGLIK-DET, LOGLIK-PF} ▷ det. / stochastic latents
  store optimizer OPT
  $p(m \mid \mathcal{D}) \leftarrow$ MODEL-POSTERIOR($\mathcal{D}, \mathcal{M}$; N_m, LOGLIK) ▷ Algorithm 6
end function

function DESIGN(self)
  return OPT. $\arg\max_{a \in \mathcal{A}}$ VoI(a; $p(m \mid \mathcal{D})$) ▷ Algorithm 9
end function

function UPDATE-STATE(self, a, obs)
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(a, \text{obs})\}$ ▷ absorb datum: $\mathcal{D}$ is now $\mathcal{D}_{0:r}$
  expanded $\leftarrow$ FALSE
  for $i = 1$ **to** R_m **do** ▷ model-search rounds (ModelSMC-style)
    $e^* \leftarrow$ PREDICTIVE-CHECK($\mathcal{D}, \mathcal{M}, p(m \mid \mathcal{D})$) ▷ pool adequate on $\mathcal{D}_{0:r}$? (Algorithm 8)
    if $e^* \le \tau_e \ \vee \ n_e \ge N_e^{\max}$ **then break** ▷ stop if adequate (n_e gate)
    end if
    $\mathcal{M} \leftarrow$ EXPAND-HYP-SPACE(LLM, $\mathcal{M}, \mathcal{D}, C$; N_{new}) ▷ propose local mutations
    $p(m \mid \mathcal{D}) \leftarrow$ MODEL-POSTERIOR($\mathcal{D}, \mathcal{M}$; N_m, LOGLIK) ▷ full-batch re-judge
    n_e += 1; expanded $\leftarrow$ TRUE
  end for
  if $\neg$ expanded **then**
    $p(m \mid \mathcal{D}) \leftarrow$ MODEL-POSTERIOR($\mathcal{D}, \mathcal{M}$; N_m, LOGLIK) ▷ refit on the new datum
  end if
end function

function GET-MODEL(self)
  return $\hat{m} \leftarrow \arg\max_m \ p(m \mid \mathcal{D})$ over any round ▷ best-so-far Occam-MAP $\hat{m}$
end function

function GET-PREDICTOR(self, flavour=MAP) ▷ return a predictor $\hat{F}(\cdot)$
  if flavour = MAP **then**
    return $\hat{F}\colon a \mapsto \mathbb{E}[F(Y) \mid a, \hat{m}, \hat{\theta}]$ ▷ plug-in MAP $\hat{m}, \hat{\theta}$
  else
    return $\hat{F}\colon a \mapsto \mathbb{E}_{p(m,\theta \mid \mathcal{D})}[F(Y) \mid a]$ ▷ Bayes model average, Eq. (2)
  end if
end function

dynamics — equivalent to $T = 1$.) An experiment *design* a is the choice of initial conditions ι, the inputs $x_{1:T}$ and the optional perturbations δ. (The SSM assumption is without loss of generality, since any non-Markovian model can be converted to Markov form, as long as the latent state space is allowed to grow.)

Latent dynamics. The distribution over latent paths is given by

$$p(z_{1:T}|a,\theta,m) = \prod_{t=1}^{T} p(z_t|z_{t-1},a,\theta,m) \tag{9}$$

where the initial condition z_0 is specified in a.

For some cases, the latent dynamics are deterministic, and are given by an ODE:

$$p(z_t|z_{t-1},a,\theta,m) = \delta(z_t - m^t(a,\theta)) \tag{10}$$

$$m^t(a,\theta) = \text{ODE-solve}(t,m,\theta',z_0,x_{1:t}) \tag{11}$$

where $\theta' = \text{do}(a.\delta,\theta)$ are the optionally perturbed parameters, $z_0 = a.\iota$ is the initial condition, and $x_{1:T} = a.x_{1:T}$ are the inputs.

In other cases (e.g., NEURONBENCHSTOCH), $p(z_t|z_{t-1},x_t,\theta)$ will be an implicit distribution, that we can sample from but cannot evaluate pointwise.

Likelihood. We usually assume the complete-data conditional likelihood is given by

$$p(y_{1:T}|z_{1:T},m,\theta) = \prod_{t=1}^{T} \mathcal{N}(y_t|f_o(z_t),\sigma^2) \tag{12}$$

where f_o is the observation model. Marginalizing out the latent variables gives the observed-data likelihood:

$$p(y_{1:T}|a,m,\theta) = \int p(y_{1:T}|z_{1:T},a,m,\theta)p(z_{1:T}|a,m,\theta)dz_{1:T} \tag{13}$$

If the latent dynamics are deterministic, the likelihood is tractable:

$$p(y_{1:T}|a,m,\theta) = \prod_{t=1}^{T} \mathcal{N}(y_t|m^t(a;\theta),\sigma^2) \tag{14}$$

where $m^t(a;\theta)$ is defined in Eq. (11). By contrast, for the stochastic models in Section F, the path integral over $z_{1:T}$ is generally intractable, so we must use approximations, which we discuss below.

SMC^3. To see why we need SMC, note that MDA has to deal with 3 levels of unknowns: the latent states $z_{1:T}$ (where we may have $T = 1$ for static systems), the parameters θ, and the model structure $m \in \mathcal{M}$. Bayesian inference at the top level requires that we compute

$$p(m \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid m)p(m)}{p(\mathcal{D})} \tag{15}$$

This is computed by the MODEL-POSTERIOR function in Algorithm 6.

To compute the marginal likelihood or evidence $Z_m = p(\mathcal{D} \mid m)$, we have to marginalize out the parameters:

$$p(\mathcal{D} \mid m) = \int p(\mathcal{D} \mid m,\theta)p(\theta \mid m)d\theta \tag{16}$$

This is computed using tempered SMC in Algorithm 5.

Finally, to compute the (observed data) likelihood, we have to marginalize out the latents:

$$p(\mathcal{D} \mid m,\theta) = \int p(\mathcal{D} \mid z,m,\theta)p(z \mid m,\theta)dz \tag{17}$$

This is computed by bootstrap particle filtering (a special case of SMC) in Algorithm 4. Using PF to estimate $p(\mathcal{D} \mid m,\theta)$ inside of an outer SMC algorithm over θ is known as SMC^2 (Chopin et al., 2013). This computes an unbiased estimate of

$$Z_m^{(r)} = p(\mathcal{D}_{0:r} \mid m) = \iint p(\mathcal{D}_{0:r} \mid m,\theta,z)\, p(z \mid m,\theta)\, p(\theta \mid m)\, dz\, d\theta, \tag{18}$$

Because the inner PF likelihood is *unbiased*, plugging the estimate $\hat{Z}_m$ in place of the exact Z_m leaves the outer sampler's target distribution invariant — the *pseudo-marginal* principle (Andrieu &

Roberts, 2009) that makes the nested estimator valid.[9] Since we nest SMC^2 inside of an outer SMC algorithm over models, we call our method SMC^3.

Note that, if the latent variables are missing, or can be computed deterministically, then we don't need the bottom-most PF layer to compute the likelihood; in this case we end up with a different form of SMC^2, where we use SMC over models and parameters. If the set of models is fixed and finite, we can eliminate the top-most SMC layer, ending up with just SMC^1 over the parameters.

Outer-most loop. The outer (model inference) loop, at step r, is approximating the following target distribution

$$\pi_r(m) \;\propto\; \pi(m)\, Z_m^{(r)} \tag{19}$$

We represent π_r by a population of at most N_m model particles $\mathcal{M} = \{m_i\}$ (the model pool). Three moves carry it from π_{r-1} to π_r. *(i) Reweight:* MODEL-POSTERIOR recomputes each evidence on the *full* data and sets $w_i \propto \pi(m_i)\,\hat{Z}_i^{(r)}$. Because the weight is the marginal evidence rather than an incremental path weight, it is *independent of the particle's history* — the population never degenerates and there is no ancestral bookkeeping; given the pool, $\{w_i\}$ is the exact restriction of π_r to $\mathcal{M}$ (up to the inner estimate of Z). *(ii) Birth:* when the pool is inadequate, EXPAND-HYPSPACE draws N_{new} new particles from the LLM proposal $q(\cdot \mid \{(m_j, \rho_j)\}, \mathcal{D})$ conditioned on the whole pool's residuals (the SMC-S kernel (Piriyakulkij et al., 2024)) — a birth move in a trans-dimensional sampler that enlarges the support. *(iii) Resample, adaptively:* birth+prune fires *only* when the predictive check fails, $e^* > \tau_e$ — an adaptive schedule keyed to a statistic of the current population and data, exactly as ESS-triggered resampling is keyed to the weights (Del Moral et al., 2012); when it fires, $\mathcal{M} \leftarrow \text{TOP-}N_m(\mathcal{M};\, \pi_r)$.

Note that, unlike traditional SMC, resampling is needed not for variance control (since the weights are exact and history-free) but for *support* control. Thus "resampling" is a form of *hypothesis search* — births propose, TOP-N_m prunes — and it runs on demand, gated by the task's own predictive check rather than a fixed schedule. Thus our method differs from the standard data tempering / IBIS method of (Chopin, 2002), as well as standard recursive Bayesian filtering, since we compute the exact full-batch evidence $Z_m^{(r)}$ for each particle m at each step r, rather than incremental path weights $p(y_r \mid m, \mathcal{D}_{0:r-1}) = Z_m^{(r)}/Z_m^{(r-1)}$. This avoids particle degeneracy, and loses nothing computationally, since the parameters need to be integrated out over all the data, which rules out a recursive solution.

In terms of correctness of this algorithm, we note that TOP-N_m is a *deterministic* truncation of the support, not stochastic resampling, so it is only exact on the retained mass; its bias is bounded by the discarded mass $\varepsilon_r = \sum_{m \notin \text{top-}N_m} \pi_r(m)$, which is monitorable and $\to 0$ as $N_m \to |\mathcal{M}|$. We can easily replace this by multinomial resampling to N_m to get an unbiased sampler at higher variance. In addition, consistency requires the birth kernel to *cover* the adequate model in the limit — the one place the LLM must be trusted, an assumption rather than a theorem. Given the coverage assumption and unbiased inner estimates, the outer level will correctly target (19).

A.4 PRIORS

In each benchmark, the candidate structures are LLM-proposed, so each has its own free parameters θ and declared bounds. The joint prior factorises as

$$p(m, \theta) = p(m) \prod_{k=1}^{C_m} p_k(\theta_k), \tag{20}$$

with a *uniform* prior p_k on each coefficient over its declared bounds, and a *structure* prior

$$p(m) \propto e^{-\lambda C_m}, \tag{21}$$

[9] The pseudo-marginal invariance uses the unbiased PF estimate $\hat{Z}_m$ *directly* (the β=1 weight). Within an inner *likelihood-tempering* pass (Algorithm 5) the intermediate weights raise the estimate to a fractional power $e^{\Delta\beta\,\hat{\ell}}$, which is biased by Jensen's inequality, so the intermediate targets are only approximately pseudo-marginal; *data* tempering (adding observations one at a time) would restore exactness. We use likelihood tempering for its adaptivity and rely on a large latent-particle count to keep the bias small.

Algorithm 3 LOGLIK-DET — the exact log-likelihood for *deterministic* latent dynamics (Eq. (14)): one noise-free rollout of m from the known initial state gives the latent path, and the emission is a product of independent Gaussians (N_z=1, no marginalisation).

def LOGLIK-DET$(y_{1:T}, a, m, \theta) \to \log \ell$

1: $z_t \leftarrow m^t(a;\theta)$ for $t = 1 \dots T$ ▷ one noise-free rollout

2: **return** $\sum_{t=1}^{T} \log \mathcal{N}\big(y_t;\ f_o(z_t),\ \sigma\big)$ ▷ Eq. (14)

Algorithm 4 LOGLIK-PF — bootstrap particle filter for $\log \hat{\ell} = \log \widehat{p}(\mathcal{D} \mid m, \theta)$, the unbiased likelihood estimate for *stochastic* latent dynamics that replaces the exact-likelihood line of Algorithm 5 (turning it into a pseudo-marginal sampler over $(\theta, z_{1:T})$). It samples N_z latent-state paths $z_{1:T}^{1:N_z}$ over the T observation steps; the transition is sub-stepped (for NEURONBENCH, the Euler–Maruyama discretisation of the Fox–Lu gating SDE, Eq. (73)). All the θ-dependence is in the transition (line 3); the observation noise σ is fixed, so the emission (line 4) does not depend on θ. The sum over the T observation steps is the loop (lines 2–7); the filter runs in $O(N_z T)$ time.

def LOGLIK-PF$(\mathcal{D}, m, \theta) \to \log \hat{\ell}$

1: $z_0^j \leftarrow z_0$ for $j = 1 \dots N_z$; $\log \hat{\ell} \leftarrow 0$

2: **for** each observation $t = 1 \dots T$ **do**

3: $z_t^j \sim p\big(z_t \mid z_{t-1}^j, \theta\big) \quad \forall j$ ▷ stochastic transition (Euler–Maruyama)

4: $w_t^j \leftarrow \mathcal{N}\big(y_t;\ f_o(z_t^j),\ \sigma\big) \quad \forall j$ ▷ emission on observed coord. $f_o(z_t)$=V_t; σ fixed

5: $\log \hat{\ell}$ += $\log\big(\frac{1}{N_z}\sum_j w_t^j\big)$ ▷ incremental log-marginal

6: resample $\{z_t^j\} \propto \{w_t^j\}$

7: **end for**

8: **return** $\log \hat{\ell}$

where C_m is the number of free parameters of m and λ is a per-parameter Occam penalty (in nats; the per-domain values are in Table 2, chosen on a validation set). The posterior over structures is then $p(m \mid \mathcal{D}) \propto Z_m\, e^{-\lambda C_m}$, with $Z_m = \int p(\mathcal{D} \mid \theta, m)\, p(\theta \mid m)\, d\theta$ the marginal likelihood computed by Algorithm 5. When $\mathcal{M}$-open expansion adds a new structure it enters with the same unnormalized prior $e^{-\lambda C_m}$ and the posterior is renormalized over the current finite pool; there is no separate catch-all mass, so the pool is always a self-normalized approximation to the open-ended structure prior.

This explicit complexity term is an *additional* regularizer beyond the marginal likelihood: on near-deterministic data a more flexible form can win Z_m simply by fitting the observation noise, so the evidence alone under-penalises flexibility. (Penalising instead the *representation* length of the mechanism — string length or Halstead complexity (Halstead, 1977) — worked less well, since it ignores the flexibility of the underlying "elementary" functions.)

A.5 LIKELIHOOD USING PARTICLE FILTERING

In this section, we discuss how to compute the likelihood of the data. For deterministic dynamics, and Gaussian observations the likelihood is given in Eq. (14), and can be computed by Algorithm 3. For stochastic dynamics, and Gaussian observations the likelihood is given in Eq. (13). In this case we use the bootstrap particle filter algorithm shown in Algorithm 4 to compute the unbiased approximation

$$\hat{p}(y_{1:T} \mid m, \theta) = \prod_t \big(\tfrac{1}{N_z} \sum_i w_t^{(i)}\big) \tag{22}$$

where w_t^i is the weight of particle i.

In Section E.3.1 we discuss a way to approximate the likelihood that is much faster than PF, known as synthetic likelihood. However, we leave evaluation of this method to future work.

Algorithm 5 Adaptive-tempering SMC over N_p parameter particles for each model m. Returns particles, weights, and $\log Z_m$. ESS $= (\sum W_i)^2 / \sum W_i^2$; η is the target ESS fraction, R_p the number of rejuvenation moves per temperature rung, and the annealing schedule runs for at most J_p rungs — so LOGLIK is called $O(N_p R_p J_p)$ times (it is cheap, and LLM-free). ℓ_i is a *log*-likelihood; LOGLIK$(\mathcal{D}, m, \theta)$ is a *plug-in* argument — LOGLIK-DET (Algorithm 3) for deterministic dynamics, or the particle filter LOGLIK-PF (Algorithm 4) for stochastic dynamics — so this SMC is agnostic to how the likelihood is formed. Parameter priors are uniform over the proposer's declared bounds, so the random-walk Metropolis rejuvenation accepts on the tempered-likelihood ratio alone, with proposals outside the support rejected.

def PARAM-POSTERIOR$(\mathcal{D}, m;$ LOGLIK$) \to (\{\theta_i\}, \{W_i\}, \log Z_m)$
sample $\theta_i \sim p(\cdot \mid m)$ for $i = 1..N_p$; $W_i \leftarrow 1/N_p$
$\beta \leftarrow 0$; $\log Z_m \leftarrow 0$
$\ell_i \leftarrow$ LOGLIK$(\mathcal{D}, m, \theta_i)$ ▷ plug-in LOGLIK-DET/-PF; Alg. 3,4
while $\beta < 1$ **do** ▷ $\le J_p$ annealing rungs
  pick $\Delta\beta \le 1 - \beta$ by bisection so $\mathrm{ESS}\big(\{W_i\, e^{\Delta\beta\, \ell_i}\}\big) = \eta N_p$
  $\log Z_m \mathrel{+}= \log \sum_i W_i\, e^{\Delta\beta\, \ell_i}$ ▷ evidence increment
  $W_i \leftarrow W_i\, e^{\Delta\beta\, \ell_i} / \sum_j W_j\, e^{\Delta\beta\, \ell_j}$; $\beta \mathrel{+}= \Delta\beta$
  resample $\{\theta_i\} \propto \{W_i\}$; $W_i \leftarrow 1/N_p$
  for $r = 1 \dots R_p$ **do** ▷ RW-Metropolis at temperature β
    propose $\theta_i' \sim q(\cdot \mid \theta_i)$; $\ell_i' \leftarrow$ LOGLIK$(\mathcal{D}, m, \theta_i')$
    $(\theta_i, \ell_i) \leftarrow (\theta_i', \ell_i')$ w.p. $\min\{1, e^{\beta(\ell_i' - \ell_i)}\}$
  end for
end while

Algorithm 6 MODEL-POSTERIOR — Bayesian model averaging over the current hypothesis pool $\mathcal{M}$ by *marginal evidence*, recomputed on the full data (no LLM). $O(N_m)$ per-structure parameter-SMC fits (Algorithm 5); the structure posterior is a softmax over evidences — an exact-weight, history-free reweighting, the outer level of the SMC^3 sampler (main text below).

def MODEL-POSTERIOR$(\mathcal{D},\ \mathcal{M};\ N_m,$ LOGLIK$) \to \big(p(m \mid \mathcal{D}),\ \hat{p}(\mathcal{D})\big)$
for $m_i \in \mathcal{M}$ **do**
  $(\{\theta_{ij}\}, \{W_{ij}\},\ \log \hat{Z}_i) \leftarrow$ PARAM-POSTERIOR$(\mathcal{D},\ m_i;$ LOGLIK$)$ ▷ full-batch evidence
end for
$p(m_i \mid \mathcal{D}) \leftarrow \mathrm{softmax}_i\big(\log \hat{Z}_i + \log \pi(m_i)\big)$ ▷ structure prior π
$\mathcal{M} \leftarrow$ TOP-$N_m\big(\mathcal{M};\ p(m \mid \mathcal{D})\big)$ ▷ evidence-prune pool to N_m
return $p(m \mid \mathcal{D})$ over the pruned pool, $\hat{p}(\mathcal{D}) = \sum_i \pi(m_i)\, \hat{Z}_i$

A.6 Marginal likelihood using tempered SMC

To compare models with different numbers of parameters, we need to marginalize out their parameters by computing the evidence:

$$Z_m = p(\mathcal{D}_{0:B} \mid m) = \int \left[\prod_{r=0}^{B} p(y_{r,1:T} \mid m, \theta)\right] p(\theta \mid m) d\theta \tag{23}$$

where $p(y \mid m, \theta)$ is the observed data likelihood computed above. To compute Z_m, we use tempered SMC with an adaptive tempering schedule, as shown in Algorithm 5. This also returns the posterior over the parameters for each model:

$$p(\theta | m, \mathcal{D}) \approx \sum_{i=1}^{N_p} W_i \mathbb{1}[\theta = \theta_i] \tag{24}$$

A.7 Model posterior using SMC-S

The main inference procedure over models, given the current hypothesis class $\mathcal{M}$, is shown in Algorithm 6. This just applies Bayes rule to compute $p(m \mid \mathcal{D})$ for each $m \in \mathcal{M}$, and then keeps the top N_m candidates.

Algorithm 7 EXPAND-HYP-SPACE — grow the pool with LLM-proposed structures conditioned on the fit residuals of the *whole* pool (SMC-S (Piriyakulkij et al., 2024); a batch of N_{new} new forms). Scoring and pruning are deferred to the next MODEL-POSTERIOR (Algorithm 6); this is the sole LLM call.

def EXPAND-HYP-SPACE(LLM, $\mathcal{M}$, $\mathcal{D}$, C; N_{new}) $\to \mathcal{M}'$

$\rho_j \leftarrow$ RESIDUALS$(m_j, \mathcal{D})$ for every $m_j \in \mathcal{M}$ ▷ per-structure fit report

$\{m'_l\}_{l=1}^{N_{\text{new}}} \sim$ LLM$\big(m \,\big|\, \{(m_j, \rho_j)\}_{m_j \in \mathcal{M}}, C, \mathcal{D}\big)$ ▷ LLM proposes a batch jointly

return $\mathcal{M} \cup \{m'_l\}$

Algorithm 8 PREDICTIVE-CHECK — score how well the current belief predicts the target F, returning the median residual e that drives the $\mathcal{M}$-open trigger of Algorithm 2. The freshly observed datum is scored *prequentially* (the current MAP model m^* has not yet seen y); with the BACKTEST flag (default on), the stored data $\mathcal{D}_{0:r-1}$ are also re-scored against m^*, and the check returns the median error over the combined set. The per-datum residual $\tilde{\ell}(a, b) = \text{median}_k\, |a_k - b_k| / \max(|a_k|, c_k)$ is per-component *relative* error, with a small floor of c_k for numerical stability. This is useful for a target with heterogeneous scales (e.g. NEURONBENCH's $[n_{\text{test}}, V_{\min}, V_{\text{end}}]$), so the error is not dominated by its largest component. For the identity-target rungs ($F(y_{1:T}) = y_{1:T}$), the median is computed over time steps $k = 1 : T$.

def PREDICTIVE-CHECK$\big(a,\ y,\ \mathcal{D}_{0:r-1},\ p(m \mid \mathcal{D}),\ \text{BACKTEST} = T\big) \to e$

$m^* \leftarrow \arg\max_m p(m \mid \mathcal{D})$ ▷ current (pre-y) MAP model

$\hat{F}(a) \leftarrow \mathbb{E}[F(Y) \mid a,\ m^*, \hat{\theta}^{m^*}]$ ▷ MAP model's target forecast

$E \leftarrow \big\{\, \tilde{\ell}\big(F(y),\ \hat{F}(a)\big) \,\big\}$ ▷ prequential error on the new (a, y)

if BACKTEST **then**

  $E \leftarrow E \cup \big\{\, \tilde{\ell}\big(F(y_j),\ \hat{F}(a_j)\big) \ :\ (a_j, y_j) \in \mathcal{D}_{0:r-1} \,\big\}$ ▷ in-sample residuals

end if

return median(E)

$\mathcal{M}$-open extension. When the current best (MAP) model's residual (error) e^* exceeds a threshold τ_e — an informative but surprising observation the current pool cannot explain (Algorithm 8) — we invoke EXPAND-HYP-SPACE (Algorithm 7): the LLM proposes a batch of N_{new} new structures conditioned on the fit residuals of the *entire* pool. This is the SMC-S proposal kernel $p_r(m_r \mid \{m^i_{r-1}\}, \mathcal{D}_{0:r})$ (Piriyakulkij et al., 2024), which conditions on the whole set of previous particles rather than a single ancestor as in ModelSMC (Wahl et al., 2026); the new structures are scored by evidence and the pool is evidence-pruned in the following MODEL-POSTERIOR call. If $e^* \le \tau_e$ the pool is left unchanged and MODEL-POSTERIOR merely re-scores the existing structures on the new datum — no LLM call. NEURONBENCH, CHEMBENCH, and BOXINGGYM all expand on demand this way, including FORCEBENCH, whose LLM-proposed force-law pool is initialised from the seed orbit and then expanded on the same residual-triggered schedule; the per-domain N_{new}, $N_e^{\max}$, and τ_e are given in Table 2.

The predictive check. The predictive check (Algorithm 8) measures, in target space, how well the current belief predicts $F(Y)$; a large error is what triggers $\mathcal{M}$-open expansion. It always scores the *prequential* error on the freshly observed datum — forecast by the pre-observation MAP model, before y is absorbed. The BACKTEST flag additionally re-scores the stored examples $\mathcal{D}_{0:r-1}$ against that same MAP model (as when back-testing a synthesised program on its training set), and the check returns the median over the combined error set. The median is deliberately robust to a single noisy outcome (we do not want to expand the hypothesis space on one outlier); the price, noted by construction, is that a lone novel-intervention failure moves the trigger only once similar failures accumulate — at which point the running prequential term (each datum forecast *before* it is absorbed) drives the expansion. Working in target space with a per-component *relative* residual means the same check applies to every rung: for the identity-target domains ($F(y)=y$) it is the relative trace error, and for NEURONBENCH it scores the heterogeneous-scale feature vector without the largest component dominating.

Algorithm 9 VOI — the experiment-design score at a under the two-level posterior, with a FLAVOUR selector. All flavours use only the per-particle predictions $\mu_i(a) = \mathbb{E}[Y_a \mid m_i, \theta_i, \mathrm{do}(a)]$ (weights W_i^m within structure m), noise-whitened by the observation covariance Σ_ε. MODEL is Lindley's model-discrimination EIG (default; between-structure disagreement); JOINT keeps within-structure parameter uncertainty; TASK designs for the target F on the query distribution $\mathcal{Q}$.

def VOI$(a,\ p(m, \theta \mid \mathcal{D});$ FLAVOUR$) \to \mathbb{R}_{\geq 0}$

$\bar{\mu}_m(a) \leftarrow \sum_i W_i^m \mu_i(a)$; $\quad \bar{\mu}(a) \leftarrow \sum_m p(m \mid \mathcal{D})\, \bar{\mu}_m(a)$

return
$$\begin{cases} \sum_m p(m \mid \mathcal{D}) \left\| \bar{\mu}_m(a) - \bar{\mu}(a) \right\|^2_{\Sigma_\varepsilon^{-1}} & \text{MODEL (Eq. (34))} \\ \sum_{m,i} p(m \mid \mathcal{D})\, W_i^m \left\| \mu_i(a) - \bar{\mu}(a) \right\|^2_{\Sigma_\varepsilon^{-1}} & \text{JOINT (Eq. (38))} \\ \mathbb{E}_{a_q \sim \mathcal{Q}}\, I\big(F(Y_{a_q}); Y_a \mid \mathcal{D}\big) & \text{TASK (Eqs. (42) and (43))} \end{cases}$$

A.8 SMC HYPER-PARAMETERS AND COMPUTATIONAL COST.

Table 2 shows the SMC parameters used in the experiments. The pool size N_m and the budget B control the overall cost: each of the B rounds, MODEL-POSTERIOR (Alg. 6) re-fits *every* live structure (up to N_m) on the full data by a fresh N_p-particle adaptive-tempering SMC (Alg. 5) — so $O(B\, N_m)$ inner SMC fits. Since each fit re-scores the *full* dataset $\mathcal{D}_{0:r}$, a single fit's cost grows linearly in r, so the total forward-simulation cost across the budget scales as $O(B^2\, N_m)$ — the price of full-batch re-fitting (amortisable by warm-starting or incremental likelihoods, which we do not exploit here). The number of LLM calls is at most $1 + N_e^{\max}$, depending on how many times EXPAND-HYP-SPACE (Alg. 7) is triggered.

quantity	symbol	FORCEBENCH	NEURONBENCH	CHEMBENCH	BOXINGGYM	alg.
pool size	N_m	24	2–5	16	16	6
new structures/round	N_{new}	6	1	4	4	7
parameter particles	N_p	60	200	100	1500	5
rejuvenation moves	R_p	3	3	3	4	5
target ESS fraction	η	0.6	0.5	0.6	0.5	5
max tempering rungs	J_p	80	adaptive	80	60	5
latent particles	N_z	1	1/250	1	1	4
model-search rounds	R_m	2	2	2	2	2
$\mathcal{M}$-open cap	$N_e^{\max}$	8	3	4	4	2
error threshold	τ_e	1.4	0.18	0.05	1.0	2
Occam penalty (nats/param)	λ	2.5	2.0	2.0	1.0	6

Table 2: **Default parameter settings for SMC**, across the four benchmarks; the last column points to the algorithm that consumes each setting. We initialize the hypothesis set using an LLM to propose N_m model particles; this number can vary across worlds. At each step, if the predictive check fails (error is above τ_e), we can optionally add N_{new} new hypotheses using EXPAND-HYP-SPACE (up to $N_e^{\max}$ times), which MODEL-POSTERIOR evidence-prunes back to N_m. We use N_p parameter particles. Most domains use $N_z = 1$ latent particles, meaning they compute the conditional likelihood either deterministically or by lumping the latents in with the parameters; the only exception is NEURONBENCHSTOCH, which uses $N_z = 250$ to integrate out $z_{1:T}$. λ is a model parameter (controlling the prior $p(m)$), not an algorithm parameter, but is listed here for convenience.

A.9 ALGORITHMS FOR EXPERIMENT DESIGN

VoI API. The design step maximises a VOI score over the design space. Algorithm 9 collects its three flavours behind one interface: they read off the same pooled two-level posterior $p(m \mid \mathcal{D})\, W_i^m$ and differ only in *which* uncertainty they reduce — the model index (model-discrimination EIG, the default), the whole latent (m, θ) (joint EIG), or the target forecast on the query distribution $\mathcal{Q}$ (task-aware VoI). We derive these objectives below.

From Value of information to expected information gain. The VoI, first proposed in (Howard, 1966), is a *decision-theoretic* concept defined as follows: for a terminal decision $d \in \Delta$ with utility $U(a, h)$ over the unknown state of nature h, the value of running experiment a is the expected gain

in attainable utility from observing its outcome *before* deciding,

$$\mathrm{VoI}_U(a) = \mathbb{E}_{y\sim p(\cdot|a,\mathcal{D})}\Big[\max_{d\in\Delta} \mathbb{E}_{p(h|\mathcal{D},y,a)}\, U(d,h)\Big] \;-\; \max_{d\in\Delta} \mathbb{E}_{p(h|\mathcal{D})}\, U(d,h), \tag{25}$$

measured in the *units of the task utility* U.

Now let the decision be model *identification*, i.e. report a distribution $q(\cdot)$ over the model index $h = m$ under the log score $U(q,m) = \log q(m)$. The inner maximiser is the posterior, $q^\star = p(M \mid \mathcal{D}, y, a)$, and $\max_q \mathbb{E}_{p(M|\cdot)} \log q(M) = -H[p(M \mid \cdot)]$, so Eq. (25) becomes

$$\mathrm{VoI}_{\log}(a) = \Big(-\,\mathbb{E}_y\, H[p(M \mid \mathcal{D}, y, a)]\Big) - \Big(-\,H[p(M \mid \mathcal{D})]\Big) = I(M; Y_a \mid \mathcal{D}) = \mathrm{EIG}(a). \tag{26}$$

This is known as the *Expected information gain* (EIG), and was first proposed by (Lindley, 1956). (We discuss other forms of VoI below.)

Estimating the EIG. We estimate the EIG as follows. First we rewrite the mutual information of Eq. (26) in its *outcome-entropy* form, $I(M; Y_a \mid \mathcal{D}) = H[Y_a \mid \mathcal{D}] - H[Y_a \mid M, \mathcal{D}]$, in terms of each structure's *posterior predictive* of the outcome,

$$P_m(\cdot) \;:=\; p\big(Y_a \mid m, \mathcal{D}, \mathrm{do}(a)\big) \;=\; \int p\big(Y_a \mid m, \theta, \mathrm{do}(a)\big)\, p(\theta \mid m, \mathcal{D})\, d\theta, \tag{27}$$

i.e. the distribution of the outcome we would observe under design a if structure m were true, with its parameters marginalised over the within-structure posterior. Writing $p(m) := p(m \mid \mathcal{D})$, the marginal predictive is the mixture $p(Y_a \mid \mathcal{D}) = \sum_m p(m)\, P_m$, so $H[Y_a \mid \mathcal{D}] = H\big(\sum_m p(m) P_m\big)$; and conditioning on $M{=}m$ makes $Y_a \sim P_m$, so $H[Y_a \mid M, \mathcal{D}] = \sum_m p(m)\, H(P_m)$. Hence

$$\mathrm{EIG}(a) \;=\; I(M; Y_a \mid \mathcal{D}) \;=\; H\Big(\sum_m p(m)\, P_m\Big) \;-\; \sum_m p(m)\, H(P_m). \tag{28}$$

The first term is the entropy of the *mixture* predictive (the total uncertainty about the outcome); the second is the average entropy *within* a structure (the irreducible outcome noise that cannot help discriminate M). Their difference is the outcome uncertainty attributable to not knowing which structure is true — equivalently $I(M; Y_a \mid \mathcal{D}) = \sum_m p(m)\ \mathrm{KL}\big(P_m \,\big\|\, \sum_{m'} p(m') P_{m'}\big)$, the mean KL from each structure's predictive to the mixture (a generalised Jensen–Shannon divergence), which is manifestly maximised by designs a that drive the P_m apart.

Gaussian approximation. The MI is intractable for general mixture, so we specialise to a Gaussian likelihood model and use a deterministic moment-matching approximation. Let $Y_a \in \mathbb{R}^d$ be the quantity the likelihood conditions on, which could be the raw trace $y_{1:T}$ or some summary. Model its outcome, conditional on the model m, as $Y_a^m = \bar{\mu}_m(a) + \varepsilon$, where

$$\bar{\mu}_m(a) = \mathbb{E}_{\theta|m,\mathcal{D}}[Y_a \mid m, \theta, \mathrm{do}(a)] \tag{29}$$

and $\varepsilon \sim \mathcal{N}(0, \Sigma_\varepsilon)$ is the observation noise ($\sigma^2 I$ in the simplest scalar case). The unconditional distribution is thus the Gaussian mixture

$$Y_a \sim \sum_m p(m \mid \mathcal{D})\, \mathcal{N}(\bar{\mu}_m(a), \Sigma_\varepsilon) \tag{30}$$

with covariance $\Sigma_\varepsilon + \Sigma_\mu(a)$, where $\Sigma_\mu(a) = \mathrm{Var}_{p(m|\mathcal{D})}[\bar{\mu}_m(a)]$ is the $d \times d$ between-class covariance of the mean predictions. The conditional entropy $H[Y_a \mid M] = \frac{1}{2} \ln\big((2\pi e)^d \det \Sigma_\varepsilon\big)$ is exact, but a Gaussian mixture has *no closed-form* differential entropy, so we *approximate* $H[Y_a]$ by the entropy of a single Gaussian of the same covariance (moment matching). The $(2\pi e)^d \det \Sigma_\varepsilon$ cancels in the difference, giving

$$I(M; Y_a \mid \mathcal{D}) = H[Y_a] - H[Y_a \mid M] \approx \tfrac{1}{2} \ln \det\Big(I + \Sigma_\varepsilon^{-1} \Sigma_\mu(a)\Big), \tag{31}$$

In the scalar case $d{=}1$, this becomes

$$I(M; Y_a \mid \mathcal{D}) = \tfrac{1}{2} \ln(1 + \mathrm{Var}[\mu(a)]/\sigma^2) \tag{32}$$

This is an *upper bound* on the true mutual information, since a Gaussian maximises entropy at fixed covariance. Furthermore, it is monotone (in the Loewner order[10]) in the between-class covariance

[10] The *Loewner (partial) order* on symmetric matrices: $A \succeq B$ iff $A - B$ is positive semidefinite (Pukelsheim, 2006). Here $\Sigma_\mu(a) \succeq \Sigma_\mu(a')$ implies Eq. (31) is at least as large at a, so a design that raises the between-class covariance in *every* direction is unambiguously more informative.

$\Sigma_\mu(a)$. This makes it a principled surrogate — a design that raises the between-class covariance can only raise the score — although, since different designs generally induce *incomparable* covariances under the Loewner order, we do not claim its $\arg\max$ exactly coincides with that of the true mixture mutual information.

We see that the above objective requires maximizing $\ln\det$ of the information matrix — equivalently it minimises the volume of the posterior credible ellipsoid (the generalised variance). In optimal experimental design (Chaloner & Verdinelli, 1995; Pukelsheim, 2006), this is called a *D-optimal* design. Alternatively, we can maximise its *trace* (the sum of the eigenvalues of $\Sigma_\varepsilon^{-1}\Sigma_\mu(a)$) rather than its log-determinant: a cheaper, more robust surrogate here, because it needs no matrix determinant and reduces to the scalar $\mathrm{Var}[\mu]/\sigma^2$ dimension-by-dimension. (Maximising the trace of an *information* matrix is a T-optimal-style discrimination criterion; it should not be confused with *A-optimality*, which instead *minimises* the trace of the *inverse* information matrix, i.e. the average posterior variance.) In practice, we therefore use the following

$$a^\star = \arg\max_{a\in\mathcal{A}} \operatorname{tr}\big(\Sigma_\varepsilon^{-1}\Sigma_\mu(a)\big) \tag{33}$$

$$= \arg\max_{a} \textstyle\sum_m p(m \mid \mathcal{D}) \left\|\bar\mu_m(a) - \bar\mu(a)\right\|^2_{\Sigma_\varepsilon^{-1}}, \tag{34}$$

$$\bar\mu_m(a) = \mathbb{E}_{\theta|m,\mathcal{D}}[Y_a \mid m,\theta,\mathrm{do}(a)] \approx \sum_i W_i^m \mu_{m,\theta_m^i}(a) \tag{35}$$

$$\bar\mu(a) = \sum_m p(m \mid \mathcal{D})\,\bar\mu_m(a) \tag{36}$$

where $\|v\|_A^2 = v^\top A v$, and we have assumed the parameter posterior is represented as a set of weighted samples, $p(\theta|m,\mathcal{D}) = \sum_i W_i^m \delta(\theta-\theta_m^i)$. (Note that using per-class *means* $\bar\mu_m(a)$ rather than noisy draws keeps EIG genuine epistemic disagreement — Lindley's intuition (Lindley, 1956) that the best experiment is the one whose outcome current beliefs least agree on.)

Why moment-matching rather than nested Monte Carlo. The standard unbiased EIG estimator is *nested* (prior-contrastive) Monte Carlo:

$$\widehat{\mathrm{EIG}}(a) = \frac{1}{N}\sum_{n=1}^{N} \log \frac{p(y_n \mid m_n, a)}{\sum_{m'} p(m' \mid \mathcal{D})\, p(y_n \mid m', a)}, \qquad m_n \sim p(m \mid \mathcal{D}),\;\; y_n \sim p(Y_a \mid m_n, \mathcal{D}), \tag{37}$$

where the inner sum is the marginal predictive $p(y_n \mid a, \mathcal{D})$. (A further inner loop over the particles θ appears when parameters are not marginalised analytically). This is the estimator BOXINGGYM's `info_gain` evaluates. It is consistent but biased at finite N, costs $O(N|\mathcal{M}|)$ per candidate design (Rainforth et al., 2024; Foster et al., 2021). This yields a noisy, slow, non-smooth objective for the design search, which is why we maximise the fast, closed-form surrogate Eq. (34) instead.

EIG for model and parameters. An alternative acquisition keeps the within-class parameter spread instead of averaging it out. By the law of total variance the *full* two-level predictive variance splits into the between-class term of Eq. (34) plus a within-class one:

$$\operatorname{tr}\Big(\Sigma_\varepsilon^{-1}\,\mathrm{Var}_{p(m,\theta|\mathcal{D})}\big[\,\mathbb{E}(Y_a \mid m,\theta,\mathrm{do}(a))\,\big]\Big) = \underbrace{\sum_m p(m \mid \mathcal{D}) \left\|\bar\mu_m(a) - \bar\mu(a)\right\|^2_{\Sigma_\varepsilon^{-1}}}_{\text{between classes (mechanism disagreement)}}$$
$$+ \underbrace{\sum_m p(m \mid \mathcal{D}) \sum_i W_i^m \left\|\mu_{m,i}(a) - \bar\mu_m(a)\right\|^2_{\Sigma_\varepsilon^{-1}}}_{\text{within class (parameter uncertainty)}}, \tag{38}$$

estimated empirically by the pooled particle variance $\sum_{m,i} w_{m,i} \left\|\mu_{m,i}(a) - \bar\mu(a)\right\|^2_{\Sigma_\varepsilon^{-1}}$ with $w_{m,i} \propto p(m \mid \mathcal{D})\, W_i^m$. Only the between-class term is collapsed by identifying the class. The within-class term is residual parameter uncertainty. If the per-structure parameter posteriors have concentrated, this latter term becomes negligible, so this full variance coincides with Eq. (34). However, optimizing Eq. (38) can be useful if the parameter values contain more useful information than just the model index.

From model-discrimination to task-aware design. So far we have assumed the goal is to identify the true model, so we have optimized EIG $I(M; Y_a)$, spending budget to resolve which *structure* generated the data. But since we ultimately evaluate performance in terms of prediction accuracy of a target variable, we only need to resolve the model *to the extent that it changes the forecast of the target* F on the query distribution $\mathcal{Q}$. The general *task-aware* value of information is the decision-theoretic Eq. (25) with the terminal action set to *forecasting the target* rather than identifying the model: at a future experimental query $a_q \sim \mathcal{Q}$, nature returns observation Y_{a_q}, from which we get feature $F_q = F(Y_{a_q})$; the agent reports a forecast $\hat{F}_q$ scored by a loss $\ell\big(F_q, \hat{F}_q\big)$; the Bayes risk (expected loss for the optimal estimator) for this query is given by

$$\mathcal{R}_\ell(a_q \mid \mathcal{D}) \;=\; \min_{\hat{F}_q} \mathbb{E}_{p(Y_q \mid a_q, \mathcal{D})}\, \ell\big(F(Y_q), \hat{F}_q\big), \tag{39}$$

So the value of running experiment a now is the expected reduction in that risk once the outcome Y_a is observed, averaged over the query distribution:

$$\mathrm{VoI}^{\mathcal{Q}}_{\ell}(a) \;=\; \mathbb{E}_{a_q \sim \mathcal{Q}}\Big[\, \mathcal{R}_\ell(a_q \mid \mathcal{D}) \;-\; \mathbb{E}_{y_a \sim p(\cdot \mid a, \mathcal{D})}\, \mathcal{R}_\ell(a_q \mid \mathcal{D}, y_a, a) \,\Big]. \tag{40}$$

This is exactly Eq. (25) with decision $d = \hat{F}$ and utility $U = -\ell$ evaluated on the target: it designs a to most improve the forecast of F on future queries, marginalising over which structure is true rather than trying to identify it. Two structures that agree on $\mathcal{Q}$ need never be told apart. This is *task-driven* (goal-oriented) experimental design proposed in (Rainforth et al., 2024; Smith et al., 2023). Recent work recasts the resulting expected-future-loss objective into a singly-intractable form, optimisable by stochastic gradients (Rossa et al., 2026), which is directly applicable here since MDA can sample the joint (m, θ, Y) from its posterior; however we leave this to future work.

In this paper, we consider two choices of ℓ, both corresponding to proper losses (so Eq. (40) is non-negative); we discuss these below.

Task-aware design for log score. Under the *log score*, where the terminal action is to return a probability distribution $\hat{p}$ over the target $F(Y)$, the loss is $\ell(F, \hat{p}) = -\log \hat{p}(F)$ and the Bayes risk is

$$\mathcal{R}_{\log}(a_q \mid \mathcal{D}) = -\min_{\hat{p}} \sum_{F_q} p(F_q | a_q, \mathcal{D}) \log \hat{p}(F_q) = H[p(F(Y_{a_q}) \mid \mathcal{D})] \tag{41}$$

Hence Eq. (40) becomes the expected mutual information

$$\mathrm{VoI}^{\mathcal{Q}}_{\log}(a) = \mathbb{E}_{a_q \sim \mathcal{Q}}\Big[H[F(Y_{a_q}) \mid \mathcal{D}] - \mathbb{E}_{Y_a} H[F(Y_{a_q}) \mid \mathcal{D}, Y_a]\Big] = \mathbb{E}_{a_q \sim \mathcal{Q}}\, I\big(F(Y_{a_q});\; Y_a \mid \mathcal{D}\big). \tag{42}$$

Task-aware design for squared error. Under the *squared* score $\ell(F, \hat{F}_q) = \|F - \hat{F}_q\|^2$ the optimal report is the posterior-predictive mean $\hat{F}(a_q)$ (Eq. (5)), so each of the two risks in Eq. (40) is a predictive variance of the target: $\mathcal{R}_\ell(a_q \mid \mathcal{D}) = \mathrm{Var}[F(Y_{a_q}) \mid \mathcal{D}]$ before the experiment, and $\mathrm{Var}[F(Y_{a_q}) \mid \mathcal{D}, y_a]$ after observing its outcome y_a. Write $\mu(a; m, \theta) = \mathbb{E}[F(Y_a) \mid m, \theta, \mathrm{do}(a)]$ for the per-particle predicted target. The VoI is the expected drop between these two variances, which collapses to a single closed form:

$$\mathrm{VoI}^{\mathcal{Q}}_{\mathrm{sq}}(a) \;=\; \mathbb{E}_{a_q \sim \mathcal{Q}}\, \frac{\mathrm{Cov}\big(\mu(a_q),\, \mu(a) \mid \mathcal{D}\big)^2}{\mathrm{Var}\big(\mu(a) \mid \mathcal{D}\big) + \sigma^2}. \tag{43}$$

Why a difference of two variances becomes a ratio. Split each risk by the law of total variance into an aleatoric part $\sigma_q^2 = \mathbb{E}_{m,\theta|\mathcal{D}}\, \mathrm{Var}[F(Y_{a_q}) \mid m, \theta]$ — the irreducible observation noise of the *future* query outcome, independent of what we observe at a — and an epistemic part $\mathrm{Var}_{m,\theta|\mathcal{D}}[\mu(a_q)]$. The aleatoric part is the same before and after, so it cancels in the difference, leaving only the epistemic terms:

$$\mathcal{R}_\ell(a_q \mid \mathcal{D}) - \mathbb{E}_{y_a} \mathcal{R}_\ell(a_q \mid \mathcal{D}, y_a) \;=\; \mathrm{Var}_{m,\theta|\mathcal{D}}[\mu(a_q)] \;-\; \mathbb{E}_{Y_a}\, \mathrm{Var}_{m,\theta|\mathcal{D},Y_a}[\mu(a_q)].$$

A *second* application of the law of total variance, now over the observation Y_a, collapses this to $\mathrm{Var}_{Y_a}\big[\mathbb{E}(\mu(a_q) \mid \mathcal{D}, Y_a)\big]$ — the variance, across the outcomes we might observe, of the *updated*

forecast of $\mu(a_q)$. Under the moment-matching of Eq. (31) we treat $(\mu(a_q), Y_a)$ as jointly Gaussian with $Y_a = \mu(a) + \varepsilon$, so the Bayes update is linear, $\mathbb{E}[\mu(a_q) \mid \mathcal{D}, Y_a] = \bar{\mu}(a_q) + \frac{\mathrm{Cov}(\mu(a_q), Y_a)}{\mathrm{Var}(Y_a)} (Y_a - \bar{\mu}(a))$, and the variance of a linear function is $\mathrm{Cov}(\mu(a_q), Y_a)^2 / \mathrm{Var}(Y_a)$. With $\mathrm{Cov}(\mu(a_q), Y_a) = \mathrm{Cov}(\mu(a_q), \mu(a))$ (the noise ε is independent) and $\mathrm{Var}(Y_a) = \mathrm{Var}(\mu(a)) + \sigma^2$ this is the summand of Eq. (43). So the ratio is *not* a ratio of risks: it is the "explained variance" of a linear–Gaussian update, i.e. the expected squared change that observing Y_a induces in the target forecast — exactly a (non-negative) reduction of the epistemic variance.

The denominator $\mathrm{Var}(\mu(a) \mid \mathcal{D}) + \sigma^2 = \mathrm{Var}(Y_a \mid \mathcal{D})$ is the total predictive variance of the observed outcome. We read all three moments off the *same* pooled two-level particle set $\{(m_i, \theta_i^m)\}$ used everywhere else — the flat weights factorise, $w_i = p(m_i \mid \mathcal{D})\, W_i^m$, so $\bar{\mu}(a) = \sum_i w_i\, \mu_i(a) = \sum_m p(m \mid \mathcal{D})\, \bar{\mu}_m(a)$ agrees with the two-level mean of Eq. (34) — with per-particle predictions $\mu_i(\cdot) = \mu(\cdot; m_i, \theta_i^m)$:

$$\bar{\mu}(a) = \textstyle\sum_i w_i\, \mu_i(a) \tag{44}$$

$$\mathrm{Var}(\mu(a) \mid \mathcal{D}) = \textstyle\sum_i w_i \left(\mu_i(a) - \bar{\mu}(a)\right)^2,$$

$$\mathrm{Cov}(\mu(a_q), \mu(a) \mid \mathcal{D}) = \textstyle\sum_i w_i \left(\mu_i(a_q) - \bar{\mu}(a_q)\right)\left(\mu_i(a) - \bar{\mu}(a)\right). \tag{45}$$

So each candidate a costs only the per-particle predictions at a and at the query points a_q — no nested Monte Carlo. This is the form we use for the (continuous, 6-D) NEURONBENCH target, summing the Σ_ε-whitened per-coordinate contributions as in Eq. (34). (For a binary target the same VoI has the closed form $\mathrm{Cov}(\pi_{a_q}, \pi_a \mid \mathcal{D})^2 / [\bar{p}_a(1 - \bar{p}_a)]$, with the aleatoric $\bar{p}_a(1 - \bar{p}_a) = \mathrm{Var}(Y_a \mid \mathcal{D})$ playing the role of the denominator; it is unused in our experiments.)

Which metric to use, and when. By default we adopt the model-discrimination EIG as a *tractable proxy* for Eq. (40), since resolving M generically resolves F, and it needs only the cheap between-class means $\bar{\mu}_m$. Furthermore, F is often too low-dimensional to design against directly. The proxy is exact when identifying the structure suffices to forecast the target ($M \Rightarrow F$), and imperfect precisely when structures are hard to tell apart yet make similar F-forecasts, or vice versa. Between these two extremes sits the *joint* (m, θ) EIG — the full-predictive-variance objective of Eq. (38), which identifies the whole latent, including directions of θ that never affect F. The three objectives are nested by the data-processing inequality,

$$\underbrace{I(M; Y_a)}_{\text{model-discrimination}} \;\le\; \underbrace{I(M, \theta; Y_a)}_{\text{joint } (m,\theta) \text{ EIG}}, \qquad \underbrace{\mathbb{E}_{a_q \sim \mathcal{Q}} I\big(F(Y_{a_q}); Y_a\big)}_{\text{task-aware}} \;\le\; I(M, \theta; Y_a), \tag{46}$$

so the task-aware objective demands the fewest bits to *act on* F: it is never more informative to estimate the latent than the joint EIG, and its sample-efficiency advantage over the joint EIG shows only when θ carries task-irrelevant directions (a high-dimensional or partly-nuisance latent, or a query distribution that visits only part of design space). We compare all of these design objectives head-to-head on NEURONBENCH in Section 4.4. As we see from Fig. 4, on the deterministic domain, EIG wins out, since inferring M is enough to reliably predict $F(y)$; but on the stochastic domain, the task-driven VoI approach edges slightly ahead, since identifying M is no longer sufficient for predicting noisy outcomes $F(y)$.

Optimising over a large design space. When the design space is large, we can use various gradient free optimizers to pick the design. For continuous spaces a common choice is CMA-ES (Hansen, 2016). For discrete spaces, we can use LLM-driven evolutionary search methods such as FunSearch (Romera-Paredes et al., 2024).

B CHEMISTRY: FURTHER DETAILS

B.1 BENCHMARK

In this section, we describe the chemistry benchmark from (Kabra et al., 2026), which they call "ActiveSciBench-Chem", but which we call CHEMBENCH for short. The goal is to learn a static algebraic function mapping seven controllable inputs (substrate, inhibitor, second substrate and product concentrations, enzyme loading, temperature and pH) to a scalar reaction rate y:

$$y = f\big(C_A, C_I, C_B, C_P, \mathrm{Enz}, T, \mathrm{pH};\, \theta\big), \tag{47}$$

An example function is

$$y = \frac{k\,\mathrm{Enz}\,C_A}{K_m + C_A + C_A^2/K_i} \tag{48}$$

The 57 worlds. There are 57 different true rules (or "worlds"), comprised of 9 canonical single mechanisms (Michaelis–Menten, competitive / uncompetitive / noncompetitive / product inhibition, substrate inhibition, Hill cooperativity, Arrhenius temperature dependence, ping-pong bisubstrate), and 48 compound mechanisms, created from combinations of these elementary mechanisms (e.g., ping-pong×Arrhenius, MM×competitive×Arrhenius and Hill×Arrhenius). The dataset is divided into easy, medium, and hard tiers, based on the mechanisms used and their corresponding parameters (some of which make the response hard to detect).

Prediction accuracy. The primary performance metric used in their paper is the held-out root-mean-squared log-error

$$\mathrm{RMSLE} = \Big[\tfrac{1}{N}\sum_{i=1}^{N}\big(\log(1{+}\hat{y}_i) - \log(1{+}y_i)\big)^2\Big]^{1/2} \tag{49}$$

computed over $N{=}1000$ test points with novel input conditions. Rates span several orders of magnitude, which is why the error is taken in $\log(1{+}\mathrm{rate})$ space.

Our prediction accuracy metric. We use a *normalized* version of this log-error so that it is comparable to the nMSE we report on the other benchmarks: the held-out MSE in $\log(1{+}\mathrm{rate})$ space divided by the variance of the log-rate targets,

$$\mathrm{nMSLE} = \frac{\frac{1}{N}\sum_i \big(\log(1{+}\hat{y}_i) - \log(1{+}y_i)\big)^2}{\mathrm{Var}\big[\log(1{+}y_i)\big]} = \frac{\mathrm{RMSLE}^2}{\mathrm{Var}\big[\log(1{+}y)\big]}. \tag{50}$$

This is an nMSE *in log-rate space* — hence the axis label "nMSE (log-rate)" — and is a monotone transform of the benchmark's Eq. (49): it is the same squared log-error, only variance-normalized and un-rooted.

Exact accuracy. The paper also proposes another metric they call "exact accuracy" (EXACC), which assesses if the law is numerically equivalent to the true law using

$$\mathrm{ExAcc} = \mathbb{1}[\mathrm{RMSLE} < \epsilon] \tag{51}$$

Note that this is computed using the unnormalized RMSLE, to be compatible with their paper. (In the paper they say they use $\epsilon = 0.01$, but in their code they use $\epsilon = 0.05$ for the chemistry domain, so we adopt the latter convention.)

Mathematical equivalence. In the paper, they propose the "symbolic accuracy" (SA) metric, which asks whether the recovered law reproduces the true *form*, independent of its fitted constants; they judge this with an LLM. We instead use a *deterministic* check — and so, to avoid conflating the two, call our metric *mathematical equivalence* (ME). As discussed in Section A.2, we parse the recovered expression into a canonical form with SYMPY, separating its free parameters (rate constants) from the input variables. We then evaluate the true law and the recovered law on a dense noise-free grid over the input box, re-fitting the recovered form's free constants by least squares. Finally we say the estimated model recovers the truth, or is mathematically equivalent to the truth, if

$$\mathrm{ME} = \mathbb{1}[\mathrm{RMSLE(SymPy)} < 0.02] \tag{52}$$

Re-fitting the constants makes the score constant-agnostic (e.g. Michaelis–Menten with K_m=1 vs K_m=2 both match), and lets us compare MDA's parametric forms and the baseline's PySR equations uniformly. Note that ME is a tighter functional-form match than ExAcc, which is a numeric check on the noisy test points. We therefore do not use ExAcc in this paper.

B.2 Parameters and their priors

The LLM proposes multiple candidate rate laws $y = f(C_A, C_I, C_B, C_P, \mathrm{Enz}, T, \mathrm{pH}; \theta)$, and each candidate carries its own free constants θ (rate constants k, Michaelis constants K_m, inhibition constants K_i, Hill coefficients, Arrhenius factors, ...). Each free parameter takes a *uniform* prior over the bounds the proposer declares.

The joint prior factorises as $p(m, \theta) = p(m) \prod_k p_k(\theta_k)$, where the prior on models has the form $p(m) \propto e^{-\lambda C_m}$, where C_m is the number of parameters in model m and $\lambda = 2.0$ is chosen from a validation set. The likelihood uses the benchmark's multiplicative $\sim$1% observation noise, fit in $\log(1+y)$ space (Eq. (50)). Other SMC parameters are listed in Table 2.

B.3 Adaptive pool size

The base method (Algorithm 2) holds the pool cap N_m fixed. ChemBench is the one domain whose large, heavily-explored pool (N_m=16) accumulates near-duplicate, over-elaborated forms, especially since we run out to $B = 60$ steps. Hence in this domain we add an optional *adaptive* shrink step: once the MAP model is confident and high-performing — its posterior mass is high *and* its residual low — we cut the cap to $N_m^{\min}$=6, evicting the over-elaborated duplicates that $\mathcal{M}$-open exploration introduces; this lets the posterior concentrate on the right model. (The pool can re-expand if the MAP error goes above threshold, or the MAP confidence drops below threshold.) In our experiments this shrinkage step gives a small improvement in mathematical equivalence, by damping the over-elaborated near-duplicates that $\mathcal{M}$-open exploration introduces.

B.4 Results

In Fig. 1a, we plot the holdout log-rate nMSE (Eq. (50)) vs number of experiments (which we cap at $B = 60$, following their paper) for MDA and the LLM-AutoSciLab agent from (Kabra et al., 2026); we see that MDA is significantly more data efficient. In Fig. 1b, we plot the mathematical equivalence for the two methods, and again see that MDA wins by a large margin. (See Table 1 for some example laws discovered by the two methods.)

Figure 7 breaks the recovery result down by difficulty tier, and adds an ablation of MDA *without* $\mathcal{M}$-open expansion. Overall, MDA recovers the true form far more often than LLM-AutoSciLab (overall ME $\sim$53% vs. 42% for our re-run of LLM-AutoSciLab, and vs. their *published* 35% on a different, unreleased subset with `gpt-4o-mini`), and does so with far fewer experiments. LLM-AutoSciLab's misses are often numerically accurate but mechanistically meaningless (e.g. recovering $10^{0.87 \log(0.5\sqrt{\mathrm{Enz}/\ldots})}$ at RMSLE=0.001, which passes even the strict 0.01 ExAcc threshold, yet the equivalence check marks it wrong). The advantage is concentrated on the easy (62%) and medium (54%) tiers. On the *hard* tier, whose true mechanisms are compound, MDA *ties* LLM-AutoSciLab (both $\approx$42%) — but dropping $\mathcal{M}$-open still hurts there (33%), so the residual-directed expansion contributes even where exact recovery is hardest, rather than adding little. Cracking the hard compound mechanisms *reliably* likely needs a *more creative proposal distribution* (a broader or more structured search over compositions), rather than more experiments or expansion rounds.

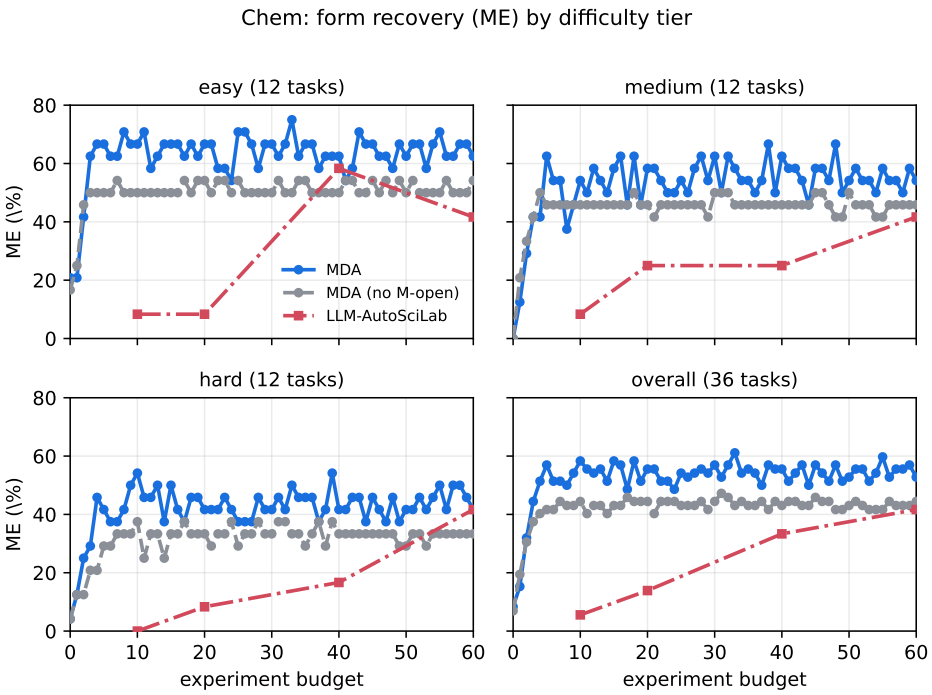


Figure 7: **Mathematical equivalence by difficulty tier** (ChemBench, B=60, 12 tasks/tier $\times$ 2 seeds). MDA vs. the same model *without* $\mathcal{M}$-open expansion ("no M-open") vs. the LLM-AutoSciLab baseline, as a function of experiment budget. MDA is markedly more data-efficient (it plateaus within $\sim$10 experiments) and wins on the *easy* (62%), *medium* (54%), and *overall* (53% vs. 42%) sets. On the *hard* tier, whose true mechanisms are compound, MDA *ties* LLM-AutoSciLab (both $\approx$42%); but dropping $\mathcal{M}$-open hurts on *every* tier — including hard (33% vs. 42%) — so the residual-directed expansion contributes throughout, and the remaining hard-tier gap is limited by the *coverage* of the proposal distribution rather than the budget.

C PHYSICS: FURTHER DETAILS

C.1 DETAILS ON THE BENCHMARK

Overview. FORCEBENCH (which is just our wrapper on DISCOVERPHYSICS from (Wiemann et al., 2026)) requires an agent to infer the unknown force law governing the behavior of two or more particles in a 2d space. Each particle i has an associated kinematic state: position $\mathbf{r}_i$, velocity $\mathbf{v}_i$, and a "generalized charge" $\mathbf{q}_i = (s_i, c_i)$, where s_i is the source charge, controlling how strongly particle i generates the field, and a response charge c_i controlling how strongly it feels the field generated by others. When $s_i = c_i$ for all particles, this reduces to a standard symmetric pairwise interaction. In this symmetric case, q_i either represents a charge (for electric fields) or a mass (for gravitational fields). The pairwise force takes the general form

$$\mathbf{F}_{i \leftarrow j} = F_{\text{mag}}(r_{ij}, \mathbf{q}_i, \mathbf{q}_j, t)\hat{\mathbf{r}}_{ij} \tag{53}$$

where $r_{ij} = ||\mathbf{r}_i - \mathbf{r}_j||$ is the distance between the particles, and $\hat{\mathbf{r}}_{ij}$ is the unit separation vector from source to receiver (so $-\hat{\mathbf{r}}$ is attractive).

There are 11 different laws or worlds, shown in Table 3. We group them into 6 TWOPARTICLEWORLDS, which follow a radial force centered on particle 1, and 5 MULTIPARTICLEWORLDS, which have slightly different semantics, as listed in the table.

#	world	pairwise force magnitude $F(r,t)$	Comments
Two-particle, central, radial (single fixed source at the origin)			
1	GRAVITY	$k\, q_i q_j / r$	Simple attractive force
2	YUKAWA	$k\, q_i q_j\, K_1(r/\lambda)/\lambda,\ \lambda{=}2$	Screened (2D Helmholtz) kernel: $\sim 1/r$ at short range, exponentially suppressed beyond λ
3	COULOMB	$k\, q_i q_j / r^2$	Simple attractive force
4	OSCILLATOR	$k\, q_i q_j \cos(\omega t{+}\phi)/r$	Time-varying coupling that periodically reverses sign
5	FRACTIONAL	$k\, q_i q_j / r^{\,3-2\alpha},\ \alpha{=}\frac{1}{2}\ (\equiv 1/r^2)$	Fractional Laplacian $-(-\nabla^2)^\alpha$ with $\alpha = \frac{1}{2}$
6	EXTRA-DIM	$k\, q_i q_j\, \Phi_{\text{KK}}(r)$	$1/r^2$ (short range) to $1/r$ (long range) transition, defined by the Kaluza–Klein kernel
Non-radial or many-body (superposed background, or N mutually-interacting bodies)			
7	CIRCLE	$k\, q_i q_j / r^{\,3-2\alpha},\ \alpha{=}\frac{3}{4}$	Fractional Laplacian with ring of particles
8	ETHER	$k\, q_i q_j / r$	Central law + global drift, $a_i = -F\,\hat{r}/m_i + \alpha\,\hat{\mathbf{y}}$
9	HUBBLE	$k\, q_i q_j / r$	Central law + position dependent Hubble flow, $a_i = -F\,\hat{r}/m_i + H(\mathbf{r}_i)$
10	DARK-MATTER	$k\, q_i q_j / r,\ q_j{\in}\{1, \underline{5}\}$	Hidden number of other particles
11	THREE-SPECIES	$k\, q_i q_j / r,\ q_j{\in}\{\underline{1, 3, -2}\}$	3 hidden classes (one repulsive) + 5 neutral probes

Table 3: **The eleven public DiscoverPhysics worlds in a single notation.** $F(r,t)$ is the pairwise force magnitude between a receiver of charge q_i and a source of charge q_j at separation r (Green's function of a 2D field equation; k a coupling, λ a screening length, α a fractional order, ω, ϕ a temporal modulation); $\Phi_{\text{KK}}(r)$ is the Kaluza–Klein image-sum kernel, that crosses over from $1/r^2$ at short range to $1/r$ at long range. Underlined quantities are *hidden* (worlds 10–11: the source charges are concealed; the task is to infer them). Observations are corrupted with fixed Gaussian position noise: the six two-particle worlds use σ=0.05 (matching §3.2 of (Wiemann et al., 2026)), and the five "extra" worlds use σ=0.03; held-out test trajectories are noise-free.

From Newton's second law, $\mathbf{F} = m\mathbf{a}$, we can derive the acceleration $\mathbf{a} = (a_x, a_y)$ of a particle as follows:

$$\mathbf{a} = -F_{\text{mag}}\hat{\mathbf{r}}/m \tag{54}$$

If there are multiple particles, we sum the forces:

$$\mathbf{a}_i = \sum_{j \neq i} F_{ij}\hat{\mathbf{r}}_{ij}/m_i \tag{55}$$

From this, we can derive the velocity by integration, and hence generate the trajectory of each particle from its initial conditions.

API. The benchmark requires the agent to submit a Python function that returns the predicted trajectory. The function must satisfy the following signature:

```
def discovered_law(pos1, pos2, p1, p2, velocity2, duration, **params):
 ...
 return trajectory
```

Here `params` are free parameters of the law which can be fit to the collected data by the FORCEBENCH environment before it calls the above function. The agent can choose the initial position of particles 1 and 2, and the velocity of particle 2. (The velocity of particle 1 is fixed at $(0, 0)$.) The meaning of the control knobs p_1 and p_2 varies across the worlds: sometimes they represent masses, sometimes charges (see Table 3 for details).

Baseline DP-AGENT. The baseline method from (Wiemann et al., 2026), which we call DP-AGENT (for "Discover Physics Agent"), uses an LLM to generate code which computes the acceleration function $\mathbf{a}$, from which it derives the trajectory by integration. In MDA, we instead estimate F_{mag}, and then derive $\mathbf{a}$ using Newton's law in Eq. (54), which we pass to the integrator. (MDA also estimates its own parameters, using the posterior mean associated with the submitted model, rather than using the environment's fitting function; both agents thus submit *fitted* laws, so the comparison isolates the model form and the design policy, not the parameter-fitting backend.) In Table 7, we give examples of the generated force laws from both methods. (We reverse engineer a symbolic F_{mag} from the LLM's $\mathbf{a}$ source code using a coding agent, to make it easier to compare to MDA's symbolic output.)

Prediction metrics. We score each run's held-out prediction error as the *normalized* MSE of Eq. (1) on the query set $\mathcal{Q}$. For FORCEBENCH the query set is the benchmark's held-out test conditions — probe orbits spanning *varied* charges, masses, and launch geometries, disjoint from the experiments the agent ran — evaluated against the true *noise-free* response (nMSE $=$ MSE$/$Var, with Var the variance of the held-out test trajectories, so scores are comparable across worlds with different total particle travel). Because the query set spans conditions the agent never ran, low nMSE reflects generalization of the recovered mechanism — its functional form *and* its fitted parameters — across the input/intervention space, not interpolation of the training orbits.

Explanation metrics. Of course, a low held-out MSE does not *certify* a correct model: a law can be "right for the wrong reasons," fitting observed orbits without capturing the mechanism. (Wiemann et al., 2026) proposed to fix this by asking each agent to return a text explanation to accompany its predicted law; this is then evaluated using an LLM judge. However, we have found this metric to be unreliable. In particular, we noticed the explanation score is essentially flat in the number of experiments, and often moves *non-monotonically* (more data making it worse). In this paper, we therefore focus on evaluating models in terms of their prediction performance, but we ensure the test set has novel perturbations, which (Richens & Everitt, 2024) showed to be a way to test if a predictive agent has a correct causal world model.

C.2 THE DESIGN SPACE

An experiment (action a) is a single probe launch in the benchmark's own API: the probe is released from position $(r_0, 0)$ with velocity v, under two scalar coupling knobs (p_1, p_2) whose roles (source charge, probe inertia, ...) are part of what must be discovered.

For MDA, we discretize the design space to make VoI maximization easier. For TWOPARTICLE-WORLDS, we use a fixed menu of 13 configurations shown in Table 4. These were automatically chosen by an LLM to cover the relevant dimensions. The design space for MULTIPARTICLE-WORLDS is shown in Table 5.

The baseline DP-AGENT uses an LLM to choose any experiment it likes, without being constrained to our menu. Despite this, it did worse than MDA. As a control, we made a modified version of DP-AGENT where we forced the LLM to choose actions from MDA's menu; this did not make any noticeable difference.

action a	r_0	v (launch)	p_1	p_2	purpose
1–8	$\{1.5, 2, 3, 4, 5, 6, 8, 10\}$	$[0, 0]$ (radial drop)	1	1	radial profile (short→long r_0)
9–10	$\{2, 4\}$	$[0, 0.4]$ (tangential)	1	1	orbit shape (angular momentum)
11	4	$[0, 0]$	2	1	identify the role of p_1
12	3	$[0, 0]$	1	2	identify the role of p_2
13	4	$[0, 0]$	2	2	vary both knobs

Table 4: **The design space $\mathcal{A}$ for the 6 TWOPARTICLEWORLDS** — a fixed menu of 13 probe-launch experiments. Each is a probe released from $(r_0, 0)$ with velocity v under coupling knobs (p_1, p_2); VoI selects one per round. The seed launch $\mathcal{D}_0$ is action 3 (the passive radial drop at r_0=3). Rows 1–8 sweep the radial force profile — the long $r_0 \geq 5$ drops probe where any screening has decayed, decisive for Yukawa/extra-dim (VoI selects r_0=5, 6; Figs. 13a, 14); rows 9–10 add angular momentum; rows 11–13 vary the two coupling knobs to identify their roles (source charge vs. probe inertia, a=F/p_2). The held-out interventional test set uses more extreme knob settings ($p_1 \in \{3, 4, 5\}$, $p_2 \in \{3, 5\}$).

world	system	each experiment sets	#	discovers
ether	central + drift $\vec{a}$=$(0, \alpha)$	5 orbiters, $r \in [3, 8]$, v=2.8	6	F, α
Hubble	central + $H\vec{r}$	5 orbiters, $r \leq 8$	6	F, H
circle	11-body ring	ring R, launch v, R-scaled t	6	exponent p
dark-matter	+ K hidden masses	continuous (x, y, v_x, v_y)	∞	# & loc. of masses
three-species	30 bg., hidden couplings	probe *direction*	4	couplings → species

Table 5: **Design spaces for the 5 MULTIPARTICLEWORLDS**. Unlike the fixed 13-launch menu of the two-particle worlds (Table 4), these are heterogeneous. Ether and Hubble use a fixed set of 5-probe orbiter launches on top of a central force plus a background term (a uniform drift α, or a Hubble expansion H); circle sweeps the radius of an 11-body self-gravitating ring with per-radius measurement times (a *wide*-ring sweep breaks the scale degeneracy that otherwise hides the force exponent); dark-matter designs a *continuous* tracer launch (x, y, v_x, v_y) by VoI to localise unseen point masses; three-species probes the 30-particle background from a few directions to recover each particle's hidden coupling, then clusters the couplings into species (count k chosen by BIC). "#" is the number of candidate experiments (∞ = a continuous design box).

C.3 PARAMETERS AND THEIR PRIORS

On FORCEBENCH the candidate structures are *open ended*: the LLM proposes a force magnitude $F(r, q_i, q_j, t; \theta)$, where each model has its own free parameters and bounds. Each free parameter takes a uniform prior over the proposer-declared bounds (Table 6), and the structure prior is the Bayesian-Occam form of Section A.4 (Eq. (21)) with $\lambda = 2.5$ — each candidate has C_m=1–3 free parameters — fit under Gaussian observation noise σ=0.05. As discussed in Section A.4, this explicit complexity prior is what keeps model selection robust to noise-fitting: on Hubble, for example, it converts a near-miss (the pool's spurious time-modulated $1/r$, a small ε fitting the noise) into a clean $1/r$ result.

Quantity	Symbol	Prior / value	Role
Inferred: a candidate's free coefficients θ (prior = Uniform over the declared bounds):			
coupling strength	k	Uniform(0.01, 5)	force magnitude
screening length	λ	Uniform(0.5, 40)	Yukawa / range cutoff
radial exponent	p	Uniform(0.5, 2.5)	power-law falloff $1/r^p$
oscillation frequency	ω	Uniform(0.1, 6)	time-varying force
oscillation phase	ϕ	Uniform(0, 2π)	time-varying force
Fixed (not inferred):			
charge / inertia roles	q_i, q_j, m	q_i=1; p_1, p_2 set charge, inertia	driving force
integrator step	Δt	0.005 (symplectic)	forward model
measurement times	t	$\{0.5, 1, 1.5, 2, 3, 4\}$	readout grid
seed launch	$\mathcal{D}_0$	one passive drop at r_0=3	warm-start data
position noise	σ	0.05 (fixed Gaussian)	likelihood

Table 6: **Parameters and priors for the FORCEBENCH force-law rung**. The candidate laws are LLM-proposed, so parameters θ vary across samples.

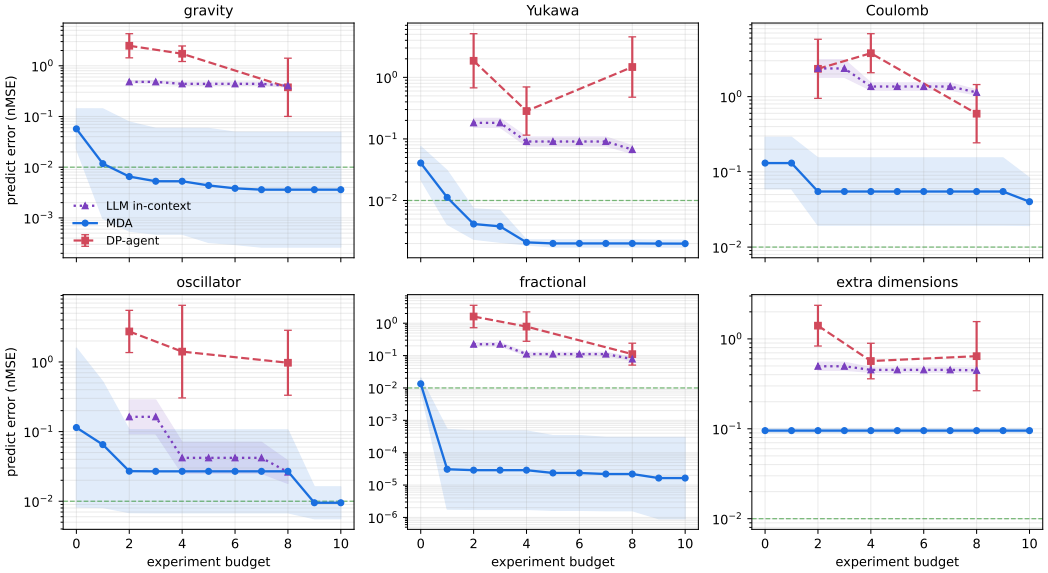


Figure 8: **Per-world data efficiency on TWOPARTICLEWORLDS** (Opus 4.7; observation noise σ=0.05). One panel per world, comparing MDA (solid) against the model-free LLM in-context forecaster (dotted) and the external DiscoverPhysics agent (dashed, at its throttled budgets $2/4/8$); all curves are best-so-far held-out nMSE vs. budget (green dashed line = the 0.01 target). MDA sits one-to-two orders of magnitude below both baselines in every world. The initial value at $B = 0$ is the fit to $\mathcal{D}_0$ before any experiments. (On *extra dimensions* the best-so-far predict error is flat because the B=0 seed fit already attains it; MDA nonetheless *finds the correct functional form* (inverse-square) by B=1 — a case where finding the right form is not rewarded by the held-out predict metric here.) Uncertainty is ± 1 SE in $\log_{10}$ over 3 seeds.

C.4 RESULTS

Data efficiency curves. In Fig. 8 we plot the data efficiency curves for each of the six TWOPARTICLEWORLDS, from which the aggregated results in Fig. 2a are obtained. In Fig. 9 we plot similar curves for each of the five MULTIPARTICLEWORLDS. We see that MDA beats the LLM agent by a large margin.

Note that the DiscoverPhysics benchmark lets an agent submit a *batch* of experiments each round, so its nominal 16-round budget collects far more than 16 experiments; run un-throttled in its native batched protocol, it uses $\sim$41 experiments and reaches nMSE 0.013, essentially reproducing the paper's number (for Opus) of nMSE 0.01. MDA reaches a comparable error in only about 4 experiments — a $\sim 10\times$ data-efficiency advantage — as shown in Fig. 2a.

Discovered laws. Table 7 shows the laws discovered by MDA and DP-AGENT after $B = 8$ experiments on TWOPARTICLEWORLDS. Looking at the details of the discovered laws, we see that sometimes the result looks different from the truth but is mathematically equal. For example, on FRACTIONAL the truth is $F = kq_iq_j/r^{3-2\alpha}$ with $\alpha = 0.5$, and MDA proposes the simpler but equivalent expression $F = kq_iq_j/r^2$. Following the DiscoverPhysics convention, Table 7 additionally reports the fraction of runs whose nMSE clears the paper's 0.1 threshold (%pass$_{<0.1}$); we drop the benchmark's second gate — an LLM-judged explanation score ≥ 0.9 — because we found it unreliable (Section C.1).

Figure 10 visualizes the recovery rate for MDA and DP-AGENT on TWOPARTICLEWORLDS, by testing if the method's discovered force reproduces the truth on a grid of points, up to a coupling scale, with an error below RMSLE ¡ 0.05. We see that MDA gets the force perfectly correct much more often than DP-AGENT. Both fail on Yukawa (exponential screening), oscillator, and extra-dimensions (Kaluza-Klein), even though MDA does well numerically on the first two of these domains; this shows that exact form recovery is a strictly harder problem than prediction.

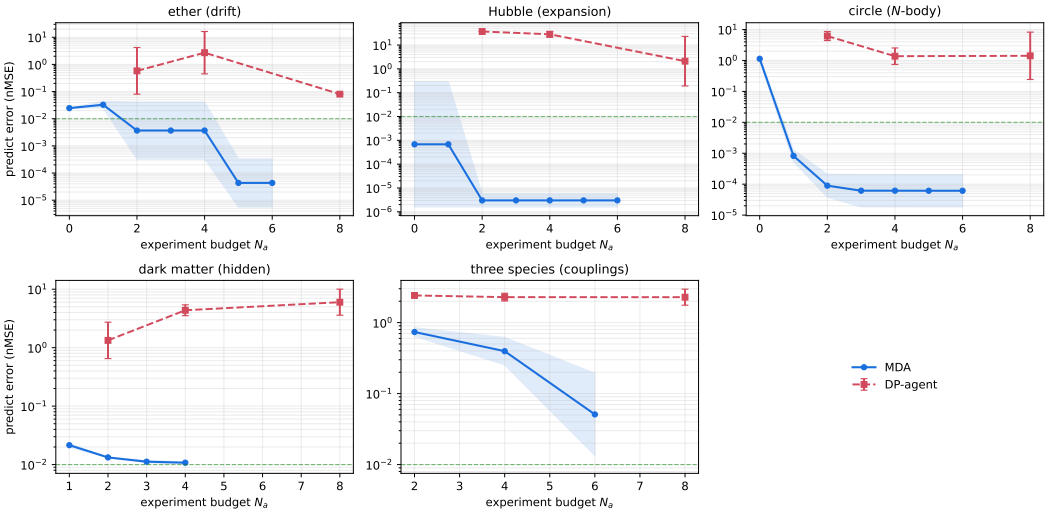


Figure 9: **Data efficiency on the five MULTIPARTICLEWORLDS** (Opus 4.7; observation noise σ=0.05). Held-out *best-so-far* nMSE vs. experiments for MDA (solid, mean±SE over 3 seeds) against the external DiscoverPhysics agent (dashed, at its throttled budgets $2/4/8$); both on the *same* nMSE scale (normalised by each world's ground-truth trajectory variance). MDA is orders of magnitude better at every budget on all five worlds. (*three species* is solved by a batched least-squares fit over the probed particle-groups, plotted at budgets $2/4/6$ groups; *dark matter* converges within a few probes, so its curve is short.) The residual nMSE on the ether, dark-matter, and three-species worlds is the intrinsic ceiling of their scoring (near-singular free-fall, a chaotic 25-body system, and a degenerate species-coupling readout), *not* a discovery failure: MDA recovers the correct drift law, the fractional ring exponent, the hidden monopole (K=1), and the species couplings.

C.5 INTERACTIVE APP

To make the task concrete, we built *PhysicsPlayground*, a self-contained web app that lets a user play a simplified version of the game that the agent must solve. Figure 11 shows a screenshot. The top row is a *transduction puzzle* in the style of ARC-AGI but for a physical law: two *training* experiments (a launch radius r_0 and the resulting orbit $r(t)$, the raw trajectory the discovery algorithm fits) and two test forecasts — a launch at a new radius, and a launch under a *perturbed source* ($\mathrm{do}(\mathrm{mass}\times 2)$), the interventional "what if" the method targets. At the bottom of the screen is the playground, where the user can launch their own orbits, read the animated trajectories, and work out the force law. Finally they submit their forecast for each test launch in the top right, and they can then choose to reveal the truth to self-score.

C.6 EXAMPLE: COULOMB WORLD

In this section, we visualize behavior of MDA when applied to COULOMB world, as shown in Fig. 12. On the left, we show what happens when a probe is launched near the source. At unit charge (p_1=1) the true law $k\,q_i q_j/r^2$ and the charge-blind overfit k/r^2 trace the *same* orbit (grey) — fit on unit-charge data, they are identical there, so *no* probe placement, at any radius or launch, can tell them apart. Turning the source charge to p_1=4 — a $\mathrm{do}(a)$ on the mechanism — scales the true law's force fourfold (green) while the overfit is unmoved (orange): the orbits split, and that split is what the observations measure.

On the right, we plot the VoI over a 2d slice of the design space, namely the release radius r_0 (an initial condition) $\times$ source charge p_1 (an intervention knob). The red $\times$ are the *seed drops*: the unit-charge probes the agent has already collected (the initial, un-designed observations both laws are fit to). VoI is ≈ 0 all along the unit-charge axis, and rises only with the charge, so MDA's VoI-driven design step reaches for a *charge intervention* (green ring), not a farther probe. Thus we see that changing a causal (mechanism) knob, not just the initial location, is needed to distinguish a correct law from a curve-fit.

method	best submitted law	nMSE	%pass$_{<0.1}$
gravity — *True law* $F{=}k\,q_iq_j/r$, $k{=}0.16$			
MDA	$F{=}$`k*qi*qj/r` k=0.159	$4.4{\times}10^{-5}$	100
LLM	`ax = -C*p1*x/(p2*r2)` C=0.158	$3.8{\times}10^{-1}$	22
Yukawa — *True law* $F{=}k\,q_iq_jK_1(r/\lambda)/\lambda$, $k{=}0.16$, $\lambda{=}2$			
MDA	$F{=}$`k*qi*qj*k1(r/lam)/lam` k=0.152, lam=2.064	$8.3{\times}10^{-4}$	100
LLM	`ax = F * dx / r` n=4.06, a=1.0, b=1.0	$3.3{\times}10^{0}$	33
Coulomb — *True law* $F{=}k\,q_iq_j/r^2$, $k{=}1$			
MDA	$F{=}$`k*qi*qj/r**2` k=0.986	$5.5{\times}10^{-2}$	56
LLM	`f = -p1 * p2 / (r2 * r)` eps=0.002	$2.8{\times}10^{-1}$	22
oscillator — *True law* $F{=}k\,q_iq_j\cos(\omega t{+}\phi)/r$, $k{=}0.80$, $\omega{=}\pi/2$, $\phi{=}0$			
MDA	$F{=}$`qi*qj*k*cos(w*t + phi)*exp(-r/lam)/r` k=1.096, lam=4.548, w=1.569, phi=-0.035	$1.3{\times}10^{-2}$	100
LLM	`ax = F * dx / r` G=0.01, n=5.0, a=1.0, b=1.0	$9.7{\times}10^{-1}$	33
fractional — *True law* $F{=}k\,q_iq_j/r^{3-2\alpha}$ ($\equiv 1/r^2$), $k{=}0.16$, $\alpha{=}\frac{1}{2}$			
MDA	$F{=}$`k*qi*qj/r**2` k=0.159	$2.5{\times}10^{-6}$	100
LLM	`a = -k * p1 / (p2 * r2)` k=0.035	$1.1{\times}10^{-1}$	56
extra-dim — *True law* $F{=}k\,q_iq_j\Phi_{\mathrm{KK}}(r)$, $R{=}0.5$			
MDA	$F{=}$`k*qi*qj/r**2 + sigma*qi*qj` k=0.254, sigma=0.023	$9.4{\times}10^{-3}$	100
LLM	`a_mag = k * p1 / (p2 * r)` k=0.163	$6.4{\times}10^{-1}$	22
grand mean MDA		$9.2{\times}10^{-4}$	93
grand mean LLM		$5.4{\times}10^{-1}$	31

Table 7: **Laws discovered for the six FORCEBENCH worlds by MDA and the pure LLM agent** (Opus 4.7, $B = 8$ experiments; regenerated from the same runs as Fig. 2a). For each world we show the true force law and each method's best (lowest-error) submitted law with its fitted parameters (MDA submits a force magnitude F; the LLM writes an acceleration line, of which we show a_x or the radial a, whichever is simpler). *nMSE* is the DiscoverPhysics normalized MSE (MSE/test-trajectory variance), geometric mean over the 9 runs. %pass$_{<0.1}$ is the fraction of runs with nMSE below the paper's 0.1 threshold (dropping their explanation score).

C.7 EXAMPLE: YUKAWA WORLD

In this section, we illustrate MDA's performance on YUKAWA. The true force law uses the screened kernel $K_1(r/\lambda)/\lambda$. It is nearly indistinguishable from a simple power law for short-range data ($r \le \lambda$), and only deviates — via exponential screening — at longer ranges. Thus an experiment with a radius confined within λ cannot tell them apart while one reaching past λ can (see Figs. 13a and 13b). VoI therefore chooses a long-range experiment, $r_0{=}5, 6$, after which the true law becomes apparent.

In Fig. 14 we plot a Pareto curve, showing the error vs complexity for different hypotheses before (gray) and after (red) the critical long-range experiment. The error is measured in bits, and is computed from the relative error in the estimated force: $\left|\log_2(F_{\text{pred}}/F_{\text{true}})\right|$. The complexity is also measured in bits, $-\log p(m \mid \mathcal{D})$, following the minimum description length (MDL) principle. We see that after the critical experiment, the true model, $K_1(r/\lambda)/\lambda$, jumps out, since it has 0 error and relatively small complexity — a true "aha" moment for the agent.

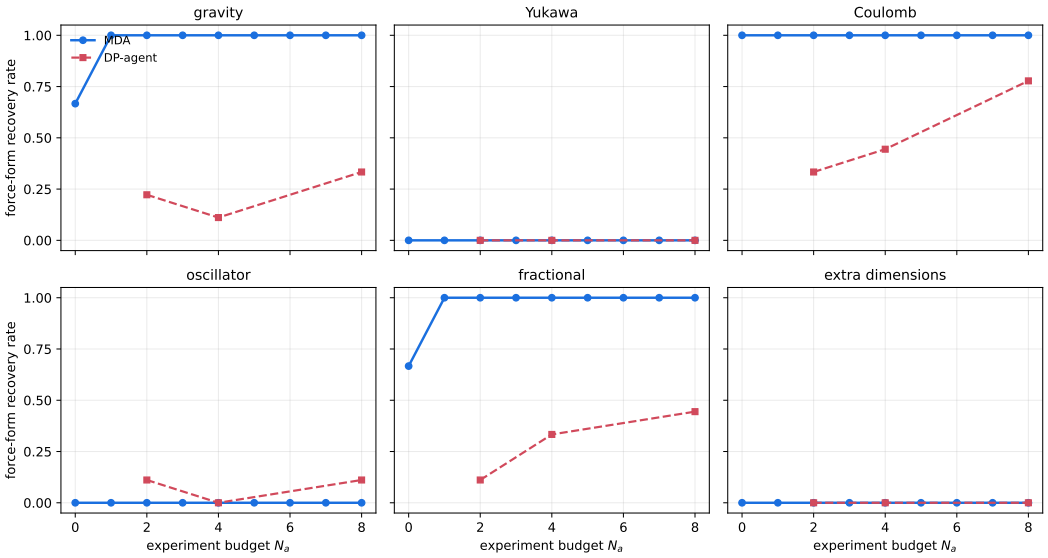


Figure 10: **Force-law recovery on the two-particle FORCEBENCH worlds: MDA vs. the DP-AGENT baseline.** Best-so-far *recovery rate* vs. budget — the fraction of runs whose discovered force reproduces the world's true $F(r)$ on a radial grid (up to a coupling scale, RMSLE$<$0.05), scored by the *same* domain-general metric for both methods. We measure each world's true $F(r)$ numerically (a probe released at rest $\rightarrow$ initial radial acceleration), then test MDA's LLM-proposed $F_{\text{mag}}(r)$ with its constants refit, or the DP-AGENT's runnable `discovered_law()` with its fitted parameters, against it. MDA recovers the force *law* more often than the trajectory-fitting baseline on the clean radial worlds (gravity, Coulomb, fractional), where the baseline attains low prediction error with a force form that is nonetheless wrong. Both methods fail on Yukawa (exponential screening) and extra dimensions (Kaluza–Klein), and both largely miss the oscillator's spatial exponent — recovery is a strictly harder bar than prediction. MDA: Opus-4.7, EIG design, 3 seeds; DP-AGENT at its throttled budgets $2/4/8$, 9 runs each.

C.8 EXAMPLE: DISCOVERING HIDDEN PARTICLES

In this section we give a simplified example of the DARK-MATTER world, where the challenge is to identify both the number and location of hidden particles.

The domain. In this domain, various probes (particles) move in a *known* static 2D Poisson field: a source of coupling q at position s pulls a probe at x with force $q/(2\pi\|x-s\|)$ toward s, and the field superposes over sources. One *visible* source of known coupling sits at the origin; the world also contains K *hidden* sources whose positions and couplings are concealed. A probe released from rest therefore falls not toward the visible source but toward the *total* mass, so it appears to accelerate toward empty space — the dark-matter tell (Fig. 15, middle, arrows). The structure m is the count K; its parameters are the $3K$ hidden coordinates and couplings. The design knob $a \in \mathcal{A}$ is the probe launch configuration (x, y).

In this section, we consider a simplified form where the true world has one visible source ($q = 2$ at the origin) and one hidden mass ($q = 4$ at $(3.5, 2)$). The method is handed three *seed* probes released far from the hidden mass, so they feel it only as a weak far-field deflection: enough to reveal that *some* unseen mass exists, but too little to say *where*. It must (i) decide how many hidden masses there are, (ii) localize the one that exists, and (iii) choose where to place the next probe.

Model selection and localization. We fit 3 models, corresponding to $K \in \{0, 1, 2\}$, using adaptive-tempering SMC (Algorithm 5, $N_p = 1000$) to compute the marginal evidence. The evidence is decisive (Fig. 15, left): $p(K{=}1 \mid D) \approx 0.97$, with $K{=}0$ excluded outright (its visible-only field cannot bend the probes toward empty space) and $K{=}2$ rejected by the Occam factor (fitting a second, redundant mass buys a negligible likelihood gain for a three-parameter prior-volume penalty). The recovered mass sits at $(3.53, 1.97) \pm (0.13, 0.09)$ with coupling 3.99 ± 0.07 — correct in count, position, and strength.

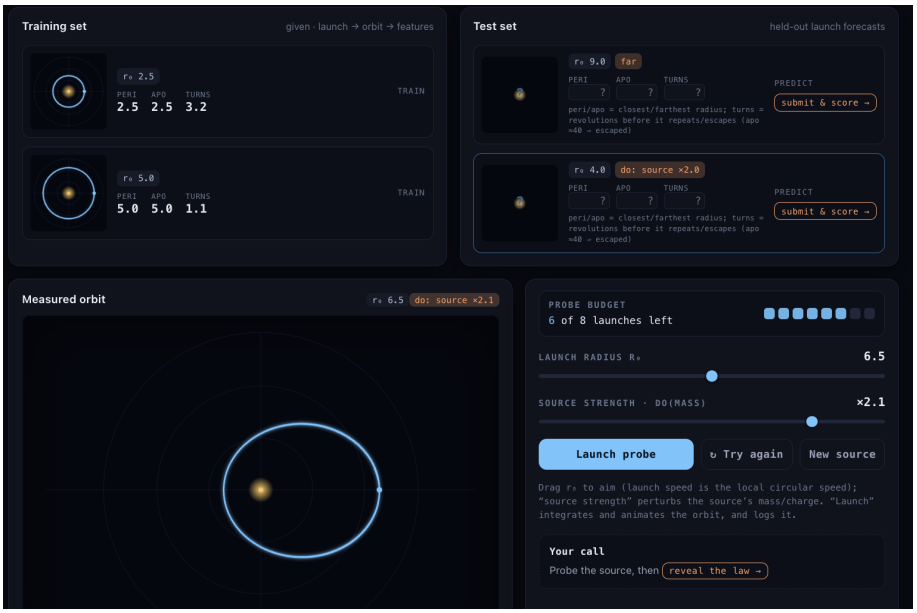


Figure 11: **App for FORCEBENCH.** The goal is to identify a hidden central-force law from a few probe orbits, then predict held-out launches — including one under a perturbed source. Training orbits (top left), held-out interventional test forecasts (top right), and the reader's own budgeted experiment bench with an animated measured orbit (bottom). Available at https://claude.ai/code/artifact/565fe6cc-a355-4c19-bf7e-b44e766cf87e.

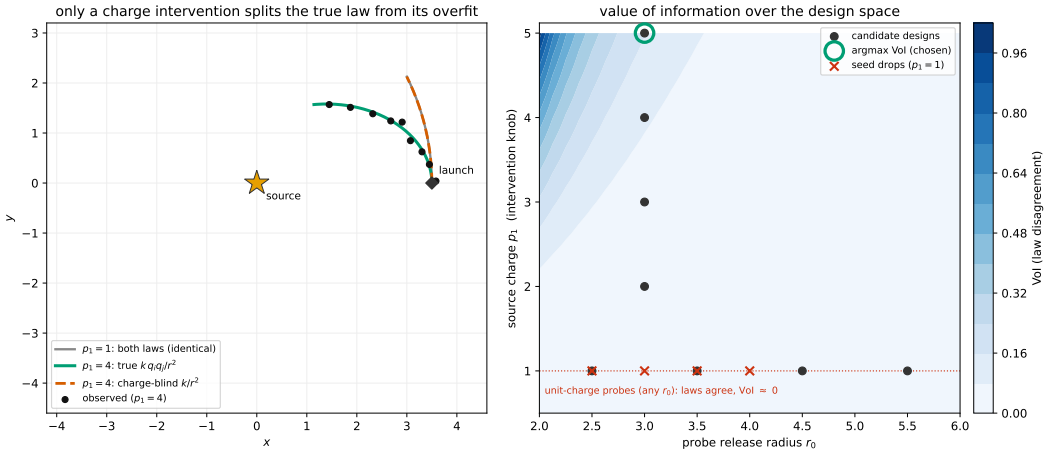
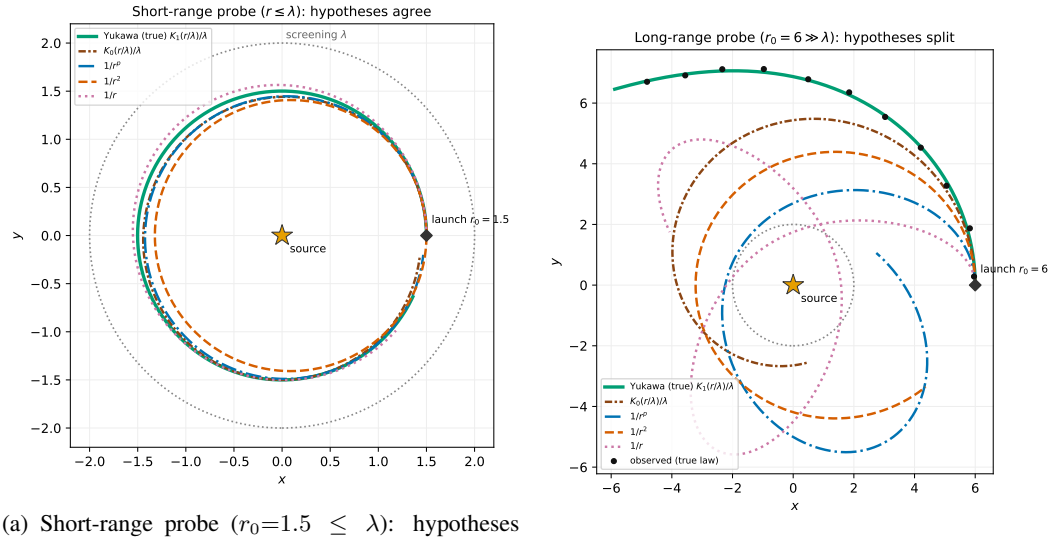


Figure 12: **Visualising COULOMB world and its design space**.

Where to look, in a 2D design space. Which placement best localizes the mass? We score each candidate by the query-relevant VoI (Eq. (43)) for a downstream query — a test probe released near the hidden mass, whose outcome depends on the hidden-mass position. The VoI landscape (Fig. 15, middle) peaks sharply on the hidden mass: a probe placed there measures it directly, while probes on the far side (visible-dominated) are nearly useless. Running the argmax probe drives $p(K{=}1)$ to 1.0 and collapses the position posterior from $\sigma \approx 0.11$ to ≈ 0.01 (Fig. 15, right) — the localization the seed probes could not reach.



(a) Short-range probe (r_0=1.5 $\leq$ λ): hypotheses agree.

(b) Long-range probe (r_0=6 $\gg$ λ): hypotheses split.

Figure 13: **Probe orbits under the candidate force laws for YUKAWA world.** The screened Yukawa kernel $K_1(r/\lambda)/\lambda$ and the power laws fit to the short-range seed data nearly coincide for $r \leq \lambda$ and diverge only beyond it. (a): launched within the screening length, every candidate traces almost the same orbit — they cannot be told apart; (b): launched well beyond it (matching the long-range probes of Fig. 14), the true screened kernel (green, with the observed data) has decayed and holds a wide slow arc, whereas the un-decayed power-law near-misses are far too strong and plunge inward — so a probe reaching past λ discriminates them.

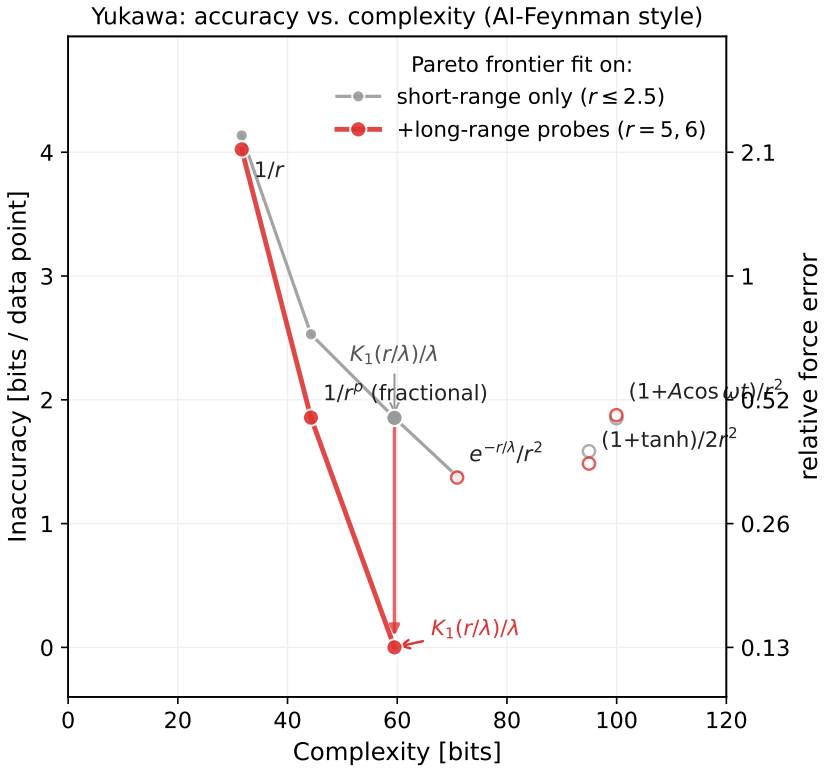


Figure 14: Accuracy-complexity Pareto frontier for models discovered by MDA in the YUKAWA physics environment. Both axes in *bits* on a *linear* scale so the convex corner is obvious. The y-axis is *inaccuracy*, the absolute relative force error in bits, $\left|\log_2(F_{\text{pred}}/F_{\text{true}})\right|$. See Section 4.2 for details. (Figure based on (Udrescu & Tegmark, 2020, Fig 1)).

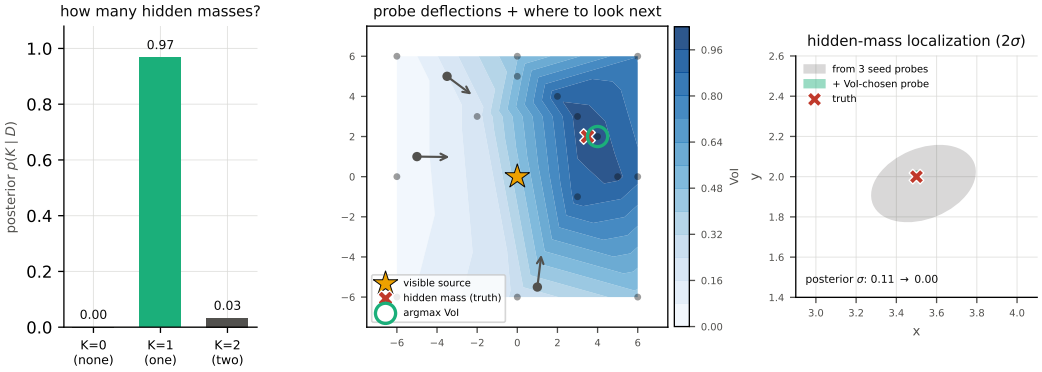


Figure 15: **The hidden-mass rung. Left:** trans-dimensional model selection $p(K \mid D)$ from the seed probes — the deflections demand a hidden mass ($K{=}0$ excluded), and Bayesian Occam rejects the surplus second mass ($K{=}2$). **Middle:** the scene. The visible source (star) sits at the origin, but the seed probes (released from the dots) deflect toward empty space (arrows) — toward the hidden mass (red cross). The blue field is the query-relevant VoI over candidate next-probe placements; it peaks on the hidden mass, and the argmax (green ring) sits essentially on it. **Right:** the $K{=}1$ posterior over the hidden-mass position (2σ ellipses): the three seed probes localize it only loosely, and the single VoI-chosen probe collapses the uncertainty onto the truth.

Table 8: The seven BOXINGGYM domains we run MDA on. Φ is the standard-normal CDF; $\sigma(\cdot)$ the logistic. The first five domains are covariate-regression domains with global parameters. Below the line we show IRT and LOCATION, which also have latent variables whose number grows with the data size (IRT: one ability per student and one difficulty per question) or model size (LOCATION: one position θ_k per source). We use code synthesis (in NUMPYRO) to represent these latent-variable models.

domain	ground-truth generative model	design a
DUGONGS	$p(y\|t;\theta) = \mathcal{N}(\mu = \alpha - \beta\lambda^t, \sigma^2)$	age $t \in [0, 5]$
DEATH	$p(y\|t;\theta) = \text{Binom}(N, \eta = 1 - e^{-\theta t})$	time $t \in (0, 2)$
PEREGRINES	$p(y\|t;\theta) = \text{Poiss}(\lambda = e^{\alpha+\beta_1 t+\beta_2 t^2+\beta_3 t^3})$	time $t \in [0, 5]$
HYPERBOLIC	$p(y\|R_i, R_d, D;\theta) = \text{Ber}(\varepsilon + (1-2\varepsilon)\Phi(\frac{R_d/(1+kD)-R_i}{\alpha}))$	$(R_i, R_d, D) \in \mathbb{R}^3$
SURVIVAL	$p(y_p\|t_p, x_p;\theta) = \text{Ber}(\sigma(t_p\, \lambda_0 e^{\beta\, x_p}))$	patient p
IRT	$p(y_{sq}\|\theta) = \text{Ber}(\sigma(\theta_q^{\text{disc}}(\theta_s^{\text{abil}} - \theta_q^{\text{diff}})))$	student s, question q
LOCATION	$p(y\|x,\theta) = \mathcal{N}(\mu = b + \sum_{k=1}^{K} \frac{\alpha}{c+\|x-\theta_k\|^2}, \sigma^2)$	$x \in \mathbb{R}^2$

D BOXINGGYM: FURTHER DETAILS

D.1 DETAILS ON THE BENCHMARK

In this section, we briefly describe the BOXINGGYM benchmark from (Gandhi et al., 2025). This implements ten scientific domains as generative probabilistic models; an agent interactively chooses designs (for up to 10 steps), observes outcomes, and is scored by (i) held-out predictive loss; (ii) the *EI-regret* of its chosen designs (compared to the EIG estimated using the ground truth model and the best of 100 random designs); and (iii) an *explain-to-a-novice* model-discovery metric. However, reporting EI-regret does not make much sense for us, since we actively optimize it, and we found the "explain-to-a-novice" metric gave very high variance results, similar to our results on the explanation metric in the physics domain (Section C.1). Thus we just report predictive loss (under novel perturbations), to be consistent with the rest of this paper.

Domains. Two of the ten domains (EMOTION, MORAL-MACHINES) use a large language model *as the participant-simulator*; they test elicitation of an LLM's implicit preferences rather than discovery of a mechanistic generative process, so we exclude them from our experiments. We also exclude the Lotka-Volterra (predator-prey) example, which requires learning an ODE, since this problem domain is already covered by our physics experiments (see Section 4.2). This leaves us with the seven domains shown in Table 8.

We partition these seven domains into two clusters. The first cluster consists of standard GLMs, where the agent needs to learn the form of the function that defines the mean of the (scalar) output, analogous to learning the symbolic laws in the physics and chemistry domains. We use tempered SMC to compute the marginal likelihood $p(\mathcal{D}|m) = \int p(\mathcal{D}|m,\theta)p(\theta \mid m)d\theta$. Because this GLM likelihood is tractable (no particle filter or synthetic likelihood), each particle is cheap and we use a large N_p=1500 parameter particles for a smooth evidence/VoI estimate (Table 2).

The second cluster consists of latent variable models, where the number of parameters can grow with the size of the data. To represent these latent variable models, we use a probabilistic programming language (PPL). Since the datasets are small, we group all these latents together with the fixed parameters, and again use tempered SMC to compute $p(\mathcal{D}|m)$.

Initial data $\mathcal{D}_0$ and query set $\mathcal{Q}$. Every BOXINGGYM run is seeded with a passive set $\mathcal{D}_0$ of n_0 designs drawn uniformly at random from the candidate pool (n_0=2 for the GLM/latent domains; the LOCATION panels of Fig. 24 use a shared n_0=3 seed set so every agent starts from identical data), after which the agent designs its experiments. The query set $\mathcal{Q}$ on which we score the held-out predictive loss is the set of *unobserved* designs: the held-out (s, q) response cells for IRT, the held-out probe locations $x \in \mathbb{R}^2$ for LOCATION, and fresh covariate points for the GLM domains — always disjoint from the experiments the agent ran.

Box's Apprentice. We reimplement the BOX'S APPRENTICE agent from (Li et al., 2024a), which was used to create the results in (Gandhi et al., 2025). In more detail, following their code, we design the Apprentice as follows: at each step it prompts the LLM to synthesize a single model $\hat{m}$ from the

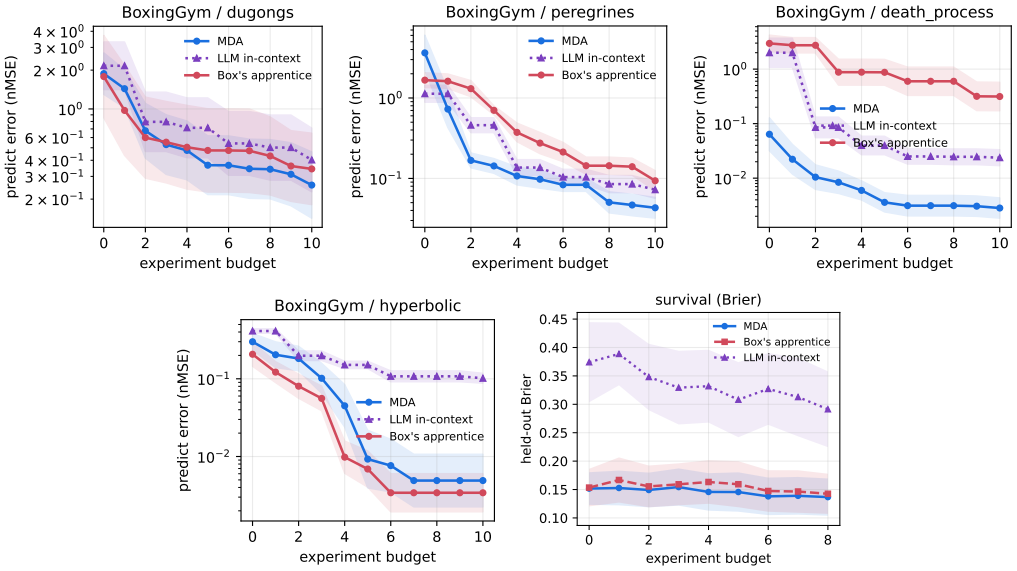


Figure 16: **BOXINGGYM learning curves for the five covariate-regression domains.** MDA, Box's Apprentice, and the model-free ICL baseline (best-so-far; mean $\pm$ SE; held-out nMSE against the noise-free mean $\mathbb{E}[Y \mid x]$, except SURVIVAL, whose Bernoulli outcome is scored by the proper Brier score).

data so far, fits $\hat{m}$, injects $\hat{m}$ (with a residual critique) into the LLM's context, and asks the LLM to choose the next experiment directly ("where should we observe next to best improve the model?"). At the end, it uses $\hat{m}$ to make predictions, similar to MDA (but different to the ICL forecaster). Both agents use the same LLM, namely Opus-4.7.

D.2 PARAMETERS AND THEIR PRIORS

Each candidate is a mean function $\mu_\theta(x)$ which is passed to the GLM likelihood. The likelihood is tractable in closed form — Gaussian with a known scale σ for the real-valued domains, or Binomial/Bernoulli for the count/binary domains — so no particle filter is needed. The parameters are given a uniform prior over their range (proposed by the LLM), and are integrated out using N_p=1500-particle tempering SMC run per model. See Table 2 for other SMC parameter settings.

D.3 RESULTS ON GLMS

The results on the 5 GLM domains are shown in Fig. 16. MDA is better than or on par with Apprentice on all five domains — by orders of magnitude on DEATH-PROCESS, more modestly on the noisier DUGONGS, and only marginally (within overlapping bands) on SURVIVAL — and it ties on HYPERBOLIC, a binary-choice domain that both agents solve almost perfectly. We also see that MDA consistently beats the model-free ICL baseline across all five domains; the apprentice is itself out-forecast by the model-free ICL baseline on DEATH-PROCESS and PEREGRINES. In addition, MDA is much more stable in its predictions than Apprentice, because MDA does Bayes model averaging over a posterior, rather than using a single MAP estimate. (Both average over their parameters.)

D.4 INFERENCE OVER LATENT VARIABLE MODELS USING A PPL

In this section, we describe how to extend MDA to work with models represented as code, using a probabilistic programming language. The original paper used PYMC, but we use NUMPYRO, which we found to be faster and more computationally robust.[11] The LLM proposes each candidate as a NUMPYRO program — a `def model(ctx)` that declares the priors over latents (z, θ)

[11] Our SMC routine calls the PPL model per candidate (model particle) and per round (experiment step), but this corrupts the underlying compilation state of PYMC after a few dozen fits, and precludes multi-seed / high-budget sweeps.

```
# BoxingGym ground truth (PyMC): a 2PL IRT model over an S x Q grid.
with pm.Model():
    alpha = pm.Normal('alpha', 0, 1, shape=S)       # per-student ability      (latent)
    beta  = pm.Normal('beta',  0, 1, shape=Q)       # per-item  difficulty     (latent)
    gamma = pm.Normal('gamma', 0, 1, shape=Q)       # per-item  discrimination (latent, 2PL)
    p = pm.math.invlogit(gamma[None, :] * (alpha[:, None] - beta[None, :]))
    responses = pm.Bernoulli('responses', p=p, shape=(S, Q))
```

Figure 17: **The IRT ground-truth generative model** (BOXINGGYM, in PYMC; S=Q=6, `mode=2pl`). Note all three of ability, difficulty, *and* discrimination are $\mathcal{N}(0,1)$ latents — Eq. (56) — so there are no fixed parameters, and because the latent scale is 1 the induced $P(\text{correct})$ sits near $\frac{1}{2}$.

with `numpyro.sample` (a `numpyro.plate` for the per-unit latents) and registers the expected observable at every candidate design as a `numpyro.deterministic('mu',...)` node. We compile it into exactly the callables MDA's loop consumes (prior sampler, prior log-density, expected observable) by reading NUMPYRO's own `biject_to` transform for each site, so the tempered SMC does its random-walk moves in the *unconstrained* space — where a positive scale, a $[0,1]$ probability, or a plate of positions all mix — while the model reads back constrained values. The observation family (Bernoulli or Gaussian) is specified in the context (part of the benchmark design). The resulting PPL model is then passed to our standard adaptive-tempering SMC, to compute the marginal likelihood

$$p(\mathcal{D}|m) = \int p(\mathcal{D}|m,z,\theta)p(z,\theta)dzd\theta$$

This is then passed to the top-most SMC over models, in the usual way. (In Section F, we consider the use of nested SMC to marginalize out a large, time-varying number of latents, $z_{1:T}$, treating them differently from the fixed parameters θ.)

D.5 RESULTS ON ITEM RESPONSE THEORY

Models. The IRT domain requires the agent to predict the probability that student s answers item q correctly. A variety of models have been proposed for this task in the literature, involving latent per-student *ability* and per-item *difficulty*/*discrimination* variables. There is a ladder of models of increasing complexity, including

$$\begin{aligned}
\text{Student}: &\quad p_{sq} = \sigma\big(\theta_s^{\text{abil}}\big) \\
\text{1PL (Rasch)}: &\quad p_{sq} = \sigma\big(\theta_s^{\text{abil}} - \theta_q^{\text{diff}}\big) && \text{(Rasch, 1960)} \\
\text{2PL}: &\quad p_{sq} = \sigma\big(\theta_q^{\text{disc}}\,(\theta_s^{\text{abil}} - \theta_q^{\text{diff}})\big) && \text{(Birnbaum, 1968)} \\
\text{3PL}: &\quad p_{sq} = \theta_q^{\text{guess}} + (1-\theta_q^{\text{guess}})\,\sigma\big(\theta_q^{\text{disc}}(\theta_s^{\text{abil}} - \theta_q^{\text{diff}})\big) && \text{(Birnbaum, 1968)} \\
\text{MIRT}: &\quad p_{sq} = \sigma\big(\theta_q^{\text{disc}\top}\theta_s^{\text{abil}} - \theta_q^{\text{diff}}\big),\ \ \theta_s^{\text{abil}} \in \mathbb{R}^D && \text{(Reckase, 2009)}
\end{aligned} \tag{56}$$

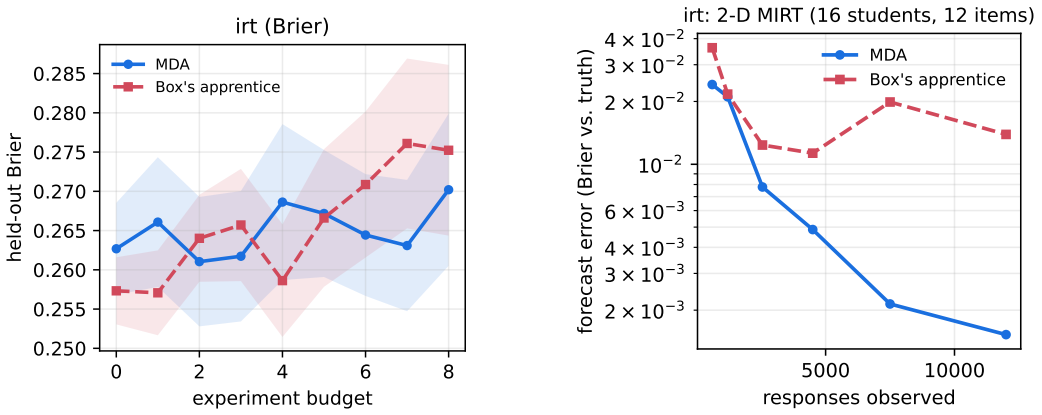
The simplest model just has one ability parameter per student; 1PL (one-parameter logistic, the *Rasch* model) has student ability and question difficulty; 2PL adds a per-item *discrimination* θ_q^{disc} (how sharply the item separates abilities); 3PL adds a per-item *guessing* floor θ_q^{guess}; multidimensional IRT (MIRT) makes ability/difficulty a D-vector. Every such per-unit parameter carries a $\mathcal{N}(0,1)$ prior (guessing is $\text{Unif}(0,1)$) and is *marginalised* out, so there are no fixed parameters in the model. The rungs differ only in *which latents exist*, and hence in their complexity (expressive power).

BOXINGGYM's ground truth is 2PL (see Fig. 17 for their code). The resulting predictive distribution has the form

$$p(Y_{sq}{=}1|m) = \int \text{Ber}\Big(1 \,\Big|\, \sigma\big(\theta_q^{\text{disc}}(\theta_s^{\text{abil}} - \theta_q^{\text{diff}})\big)\Big)\, p(\theta_s^{\text{abil}})\, p(\theta_q^{\text{diff}})\, p(\theta_q^{\text{disc}})\ \mathrm{d}\theta_s^{\text{abil}}\, \mathrm{d}\theta_q^{\text{diff}}\, \mathrm{d}\theta_q^{\text{disc}}. \tag{57}$$

Designs. The agent designs which of the $S{\times}Q$ *cells* (s,q) to query; it sees only the queried cells — a *sparse, partially observed* response grid — and is scored on the held-out cells.

Results. Figure 18a shows the Brier score — defined as $(y-p)^2$, where $y \in \{0,1\}$ is the true outcome and p is the predicted probability — for MDA and Apprentice as a function of number of experiments. Both are similar in performance, but curiously, both get worse with more data. This is an artefact of the domain being very small: there are only 6 students and 6 questions, and only 8 of



(a) default 6×6 IRT (near-chance)

(b) higher-signal 2-D MIRT IRT

Figure 18: **Code-MDA vs. Box's Apprentice on IRT, at two signal levels** (LLM-authored NUMPYRO programs). (a): the default 6×6 BOXINGGYM instance is near-chance (Fig. 17), so there is no structure to discover and MDA and the apprentice tie within noise (held-out Brier ∼0.27, 20 seeds). (b): on the higher-signal 2-D MIRT instance (Fig. 21: 16 students × 12 items with a hidden second ability axis), the forecast error (Brier of the posterior-predictive vs. the true response probabilities) falls monotonically as the response matrix fills.

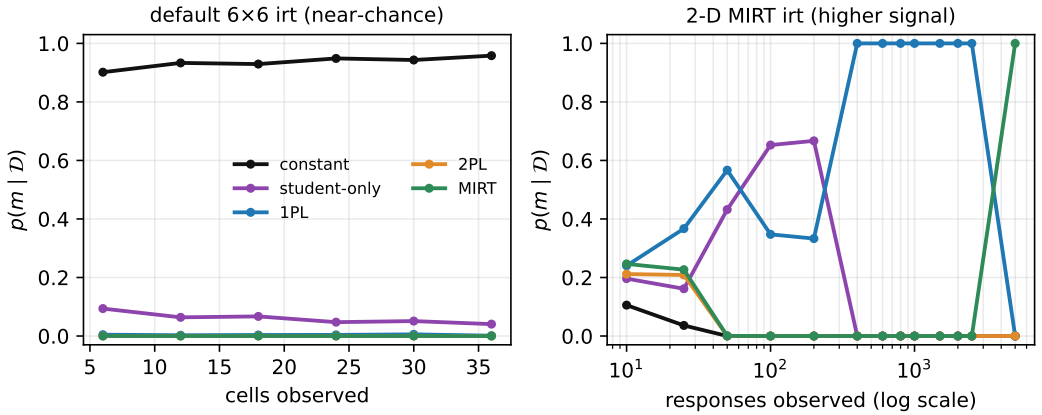


Figure 19: **Posterior over models for the two IRT domains**, as computed by MDA. *Left:* low-signal BOXINGGYM regime — with few experiments / little data, MDA keeps most of its mass on a simple constant model. *Right:* high-signal regime — as the data grows, MDA climbs the complexity ladder, eventually converging on the true 2-D MIRT model.

the cells are observed, and each one only contains a single binary response. MDA's LLM proposes multiple possible models, given the context and observed data (see Fig. 20), but there is not enough evidence to move away from a constant predictor[12], as shown in Fig. 19 (left). This is an example of the Bayesian Occam's razor penalizing overly complex models. (The LLM hallucinates, but the Bayesian machinery keeps the agent honest.) Below we design a larger scale experiment to see if either agent can discover structure if there is enough signal in the data to warrant it.

Results on a larger problem. To stress test the agents in a problem setting where there is more data, we create an example (not part of BOXINGGYM) as follows. The true data generating process is a MIRT model with two latent dimensions, representing the domains of math and English. There are 16 students and 12 items; each question loads onto one of these two domains, as specified by

[12]The constant fits the empirical base rate ≈0.667, whose Bernoulli variance 0.667×0.333≈0.22 is the Brier floor; the measured held-out Brier sits slightly above this (∼0.27) because the true per-cell rates vary, and the fluctuations are small-sample effects. Because the VoI policy queries the most *informative* (not the most representative) cells, a model fit to that active sample drifts slightly from the global base rate, so the held-out Brier can even creep up with budget — a mild active-sampling bias, not a modelling failure.

```
# IRT: per-student/per-item latents via numpyro.plate. Verbatim LLM-generated NumPyro programs (mda2.
    propose_numpyro).
# ctx["features"] is the (C,d) matrix of all candidate designs; the observation
# family (Bernoulli/Gaussian) is a task constant applied externally by the SMC.
import numpyro, numpyro.distributions as dist, jax, jax.numpy as jnp

def constant(ctx):   # Null model: single global success probability
    F = ctx['features']
    C = F.shape[0]
    p = numpyro.sample('p', dist.Beta(1.0, 1.0))
    mu = jnp.ones(C) * p
    numpyro.deterministic('mu', mu)

def rasch(ctx):   # Rasch IRT: student ability minus question difficulty via logistic
    F = ctx['features']
    s_idx = F[:, 0].astype(jnp.int32)
    q_idx = F[:, 1].astype(jnp.int32)
    with numpyro.plate('students', 6):
        ability = numpyro.sample('ability', dist.Normal(0.0, 1.5))
    with numpyro.plate('questions', 6):
        difficulty = numpyro.sample('difficulty', dist.Normal(0.0, 1.5))
    logits = ability[s_idx] - difficulty[q_idx]
    mu = jax.nn.sigmoid(logits)
    numpyro.deterministic('mu', mu)

def twopl(ctx):   # 2PL IRT: per-question discrimination scales ability minus difficulty
    F = ctx['features']
    s_idx = F[:, 0].astype(jnp.int32)
    q_idx = F[:, 1].astype(jnp.int32)
    with numpyro.plate('students', 6):
        ability = numpyro.sample('ability', dist.Normal(0.0, 1.5))
    with numpyro.plate('questions', 6):
        difficulty = numpyro.sample('difficulty', dist.Normal(0.0, 1.5))
        discrimination = numpyro.sample('discrimination', dist.LogNormal(0.0, 0.5))
    logits = discrimination[q_idx] * (ability[s_idx] - difficulty[q_idx])
    mu = jax.nn.sigmoid(logits)
    numpyro.deterministic('mu', mu)
```

Figure 20: **The IRT candidate programs, as *verbatim* LLM-generated NUMPYRO code.** Each is one rung of the ladder Eq. (56): the model index m selects *which* per-unit latents are present (a `numpyro.plate` over students / questions), so the programs have different latent counts (1, S, $S{+}Q, \ldots$). The `deterministic('mu',...)` node returns $P(\text{correct})$ at every (s, q) cell; the evidence-SMC marginalises the plated latents to score each program. The LLM also emits 3PL and MIRT (Eq. (56)) across proposals.

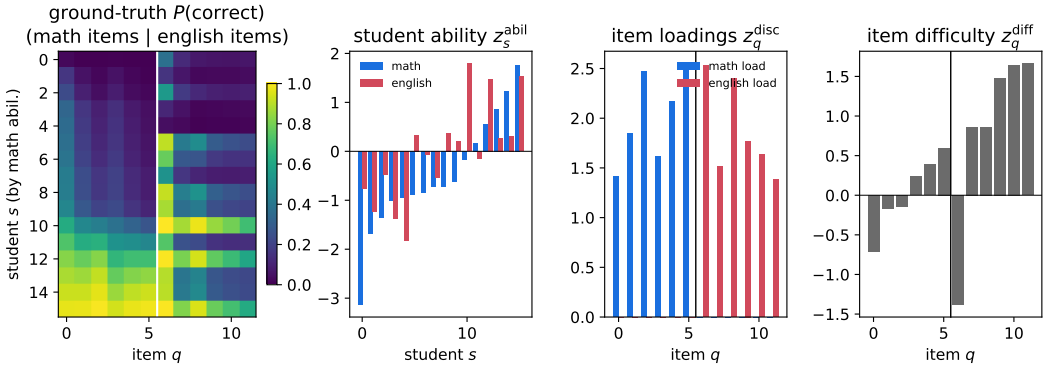


Figure 21: **Ground truth for the stronger-signal MIRT IRT instance.** Left: the ground-truth $P(\text{correct})$ response grid (16 students $\times$ 12 items, sorted by math ability; the vertical rule splits the math and english item blocks). Right: the generative latents — a *two*-dimensional student ability (math, english), per-item loadings on each axis, and per-item difficulty. The hidden second (english) ability axis is what a 1PL/2PL model cannot capture and MIRT can.

θ_q^{disc}, and each student has different abilities on these two domains, as specified by θ_s^{abil}. In addition each question has an intrinsic difficulty, as specified by θ_q^{diff}. The resulting parameters are shown in Fig. 21. We also show the mean of the predictive distribution $p(y_{sq} = 1)$, which has a clear block structure (math vs English), as well as a low-rank structure.

At each round r, the agent gets to pick N_r cells to examine; for each cell it observes a binary label $y_{sq} \sim \text{Ber}(\mu_{sq})$. The total number of counts for a cell is then $N_r(s, q)$, of which $K_r(s, q)$ are successful, which the agent models using a Binomial likelihood. The goal of the agent is to recover the underlying model, and to predict performance of heldout (s, q) combinations.

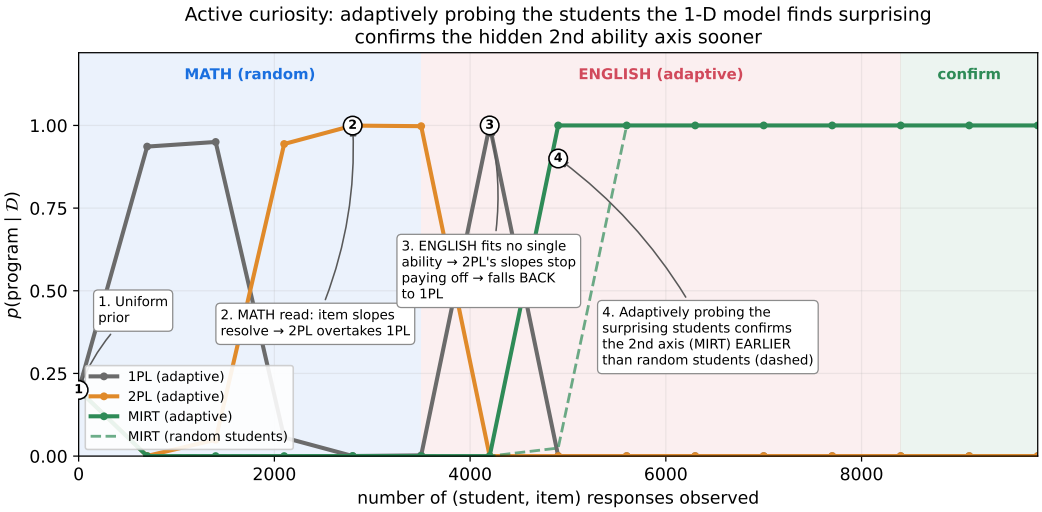


Figure 22: **A timeline through program space (with an "aha moment").** Posterior $p(\text{program} \mid \mathcal{D})$ over the IRT ladder as (s, q) responses accumulate on the instance of Fig. 21. **(2)** once the math items resolve, 2PL overtakes 1PL; **(3)** the english block fits no single ability, so 2PL's per-item slopes stop paying off and the posterior falls *back* to 1PL; **(4)** MIRT is confirmed once the second axis appears — and *adaptively* probing the surprising students (solid) confirms it earlier than probing random students (dashed). (Compare to (Jaynes, 2003, Fig. 4.1).)

In Figure 18b, we let the agent pick $N_r = 150$ cells per round, chosen freely from all 192 cells. The MDA agent maximizes the VoI, so it can re-probe cells whose outcome (or underlying factors) is still uncertain. The Apprentice tries to emulate this behavior using an LLM. We see that MDA is significantly more sample efficient, and adaptively allocates its $N(s, q)$ budget. In Fig. 19 (right), we see that as the sample size increases, the MDA agent shifts its probability mass to more complex models, eventually converging on the true MIRT model.

To make this pattern clearer, we created another scenario based on a curriculum. We let each agent pick $N_r = 700$ cells per round. In phase A, there are 5 rounds, and we require the agent to only grade math questions (giving a total of $5 \times 700 = 3500$ math results). In phase B, there are 7 rounds, and we require the agent to only grade english questions (giving a total of $7 \times 700 = 4900$ english results). In phase C (confirmation phase), there are 2 rounds, and the agent can query any cell it wants (giving a total of $2 \times 700 = 1400$ results). Thus the total number of observations is 9800.

In Fig. 22, we show the agent's journey of discovery, as it sees more structure in the data, and designs experiments to confirm or refute its beliefs along the way. In phase A, it starts out fitting a simple 1PL model, to capture the fact that some students are better at math than others. After getting enough data it switches to a 2PL model, to capture the fact that some questions are more discriminating than others. (The discrimination/slope of an item, θ_q^{disc}, is how sharply it separates high- from low-ability students; in the true model, these values are [1.4, 1.8, 2.5, 1.6, 2.2, 2.5] for the math questions.) Once it enters phase B, it sees that there are students who seemed "smart", but who do poorly on english, so it drops back to the simpler 1PL model, since using a slope of 1 (instead of $\theta_q^{\text{disc}} > 1$) means the predicted ability (pinned to math), θ_s^{abil}, is downweighted, reducing the errors on the English questions. To try to resolve which model is correct, the MDA agent probes the students whose English answers most surprise the current 1PL model (based on per-student residual). This enables it to discover there is a second latent ability axis — an "aha" moment, after which the agent can explain the data very concisely using a 2d MIRT model. From Fig. 22, we see that an agent that actively chooses what experiments to run reaches this aha moment sooner than one that chooses experiments randomly (see solid green line vs dotted green line).

```
# Location: source-count (K) discovery. Verbatim LLM-generated NumPyro programs (mda2.propose_numpyro).
# ctx["features"] is the (C,d) matrix of all candidate designs; the observation
# family (Bernoulli/Gaussian) is a task constant applied externally by the SMC.
import numpyro, numpyro.distributions as dist, jax, jax.numpy as jnp

def one_source(ctx):   # Single point source with unknown location and amplitude over constant background.
    F = ctx['features']
    b = numpyro.sample('b', dist.HalfNormal(1.0))
    alpha = numpyro.sample('alpha', dist.HalfNormal(5.0))
    m = numpyro.sample('m', dist.HalfNormal(1.0)) + 0.05
    theta = numpyro.sample('theta', dist.Uniform(-2.5*jnp.ones(2), 2.5*jnp.ones(2)))
    d2 = jnp.sum((F - theta)**2, axis=1)
    mu = b + alpha / (m + d2)
    numpyro.deterministic('mu', mu)

def two_sources(ctx):   # Two point sources with unknown locations sharing amplitude/scale.
    F = ctx['features']
    b = numpyro.sample('b', dist.HalfNormal(1.0))
    alpha = numpyro.sample('alpha', dist.HalfNormal(5.0))
    m = numpyro.sample('m', dist.HalfNormal(1.0)) + 0.05
    with numpyro.plate('sources', 2):
        tx = numpyro.sample('tx', dist.Uniform(-2.5, 2.5))
        ty = numpyro.sample('ty', dist.Uniform(-2.5, 2.5))
    d2 = (F[:,0:1] - tx[None,:])**2 + (F[:,1:2] - ty[None,:])**2
    mu = b + jnp.sum(alpha / (m + d2), axis=1)
    numpyro.deterministic('mu', mu)

def three_sources(ctx):   # Three point sources with shared kernel parameters.
    F = ctx['features']
    b = numpyro.sample('b', dist.HalfNormal(1.0))
    alpha = numpyro.sample('alpha', dist.HalfNormal(5.0))
    m = numpyro.sample('m', dist.HalfNormal(1.0)) + 0.05
    with numpyro.plate('sources', 3):
        tx = numpyro.sample('tx', dist.Uniform(-2.5, 2.5))
        ty = numpyro.sample('ty', dist.Uniform(-2.5, 2.5))
    d2 = (F[:,0:1] - tx[None,:])**2 + (F[:,1:2] - ty[None,:])**2
    mu = b + jnp.sum(alpha / (m + d2), axis=1)
    numpyro.deterministic('mu', mu)
```

Figure 23: **The LOCATION programs, as *verbatim* LLM-generated NUMPYRO code.** Model discovery is over the *number* of sources K: the LLM writes a ladder `one_source`, `two_sources`, `three_sources`,..., each a `numpyro.plate` of K latent source positions θ_k over the exact inverse-square signal field $b + \sum_k \alpha/(c + \|x - \theta_k\|^2)$ (Eq. (58)). The `deterministic('mu', ...)` node is the mean observable at every candidate probe.

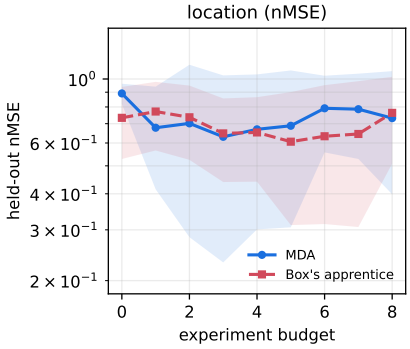
D.6 RESULTS ON LOCATION

The true predictive model for the LOCATION (source finding) problem, at location $x \in \mathbb{R}^2$, is

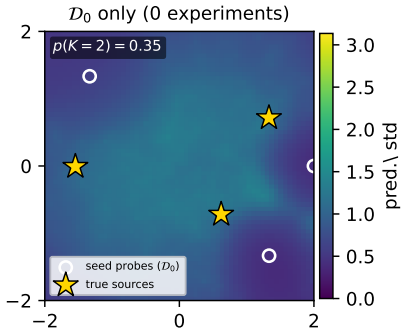
$$p(Y_x \mid m) = \int \mathcal{N}\Big(Y_x \,\Big|\, b + \sum_{k=1}^{K} \frac{\alpha}{c + \|x - \theta_k\|^2},\ \sigma^2\Big) \left[\prod_{k=1}^{K} p(\theta_k)\right] p(\theta_b, \theta_c, \theta_\alpha, \theta_\sigma) d\theta \tag{58}$$

with latent source positions $\theta_k \sim \mathcal{N}(0, I_2)$ and fixed globals (b, c, α, σ) all marginalized out. The model structure m corresponds to the number of sources K. The agent-generated code for each of these models is shown in Fig. 23.

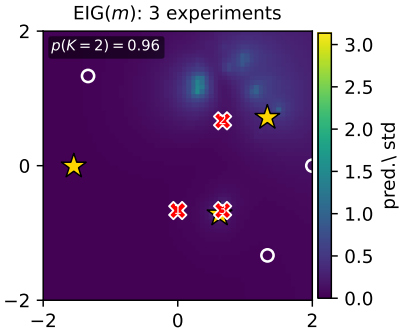
The performance of MDA and Apprentice on this domain are shown in Fig. 24a. Both perform very similarly. The other panels show how the EIG objective causes the agent to place probes in locations that reduce its uncertainty about how many sources there are (i.e., the model identity). We also tried maximizing the EIG for the model and its parameters (Eq. (38)) — which would additionally pin down the source locations θ_k — but we found that the greedy variance objective is too myopic. In particular, because the signal field is near-singular next to a source, it piles probes onto a single hypothesized source (where a few particles' predictions blow up) rather than spreading them, giving unstable results worse than random design. In principle one might think that task-driven VoI, as in Eq. (40), would work better, but it suffers from the same problem: occasionally task-VoI steers a probe into that near-singular region, and then the prediction error blows up, and it takes many seeds to average this effect away, thus making the VoI estimate unreliable. So on this domain active design gives no advantage: the EIG policy and random sampling perform equally well. (Fig. 24(c,d) illustrate where the EIG policy places its probes; the advantage it would buy on a more identifiable field simply vanishes here.)



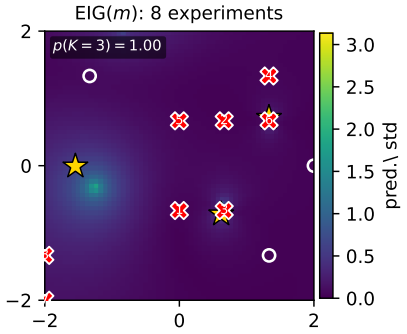
(a) data efficiency



(b) shared $\mathcal{D}_0$ (0 experiments)



(c) 3 EIG experiments



(d) 8 EIG experiments

Figure 24: **Code-MDA on LOCATION (source finding over LLM-authored NUMPYRO programs).** (a): held-out nMSE vs. budget (20 seeds, median $\pm$ IQR/2). The remaining panels show the posterior-predictive std of the log-signal field (viridis) as EIG experiments accumulate data, starting from the *same* shared passive dataset $\mathcal{D}_0$ (white circles $=$ the 3 seed probes given to every agent), with the numbered red $\times$ the probes EIG chooses (drawn on top, so the count matches the title) and gold stars the true $K{=}3$ sources. EIG places probes where the K-source programs disagree, collapsing the field uncertainty and driving the posterior over the source count to the truth ($p(K{=}3 \mid \mathcal{D}){\to}1$) — MDA's evidence-SMC *discovers* the right structure.

E NEURONBENCH: FURTHER DETAILS

In this section, we introduce our new NEURONBENCH benchmark, available at https://github.com/murphyk/neuronbench. (We defer details of its stochastic extension to Section F.) We also give a brief primer on neuron electrophysiology and Hodgkin–Huxley models, to make this section comprehensible to a machine learning audience.

E.1 BACKGROUND

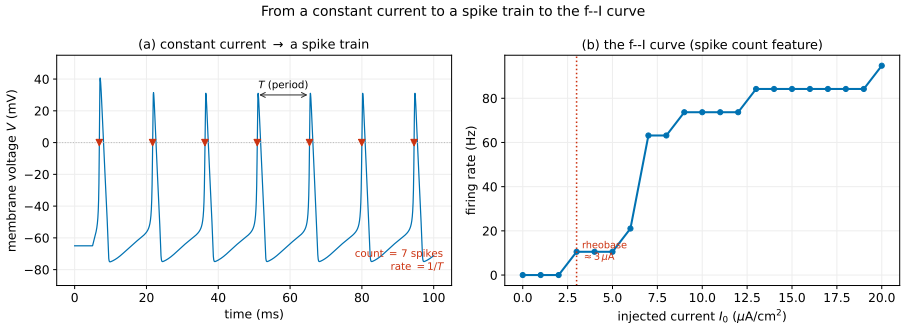


Figure 25: **The f–I curve, and why we count spikes.** *(a)* A constant supra-threshold current makes the model fire a periodic *spike train*; the readout is simply the *spike count* (red markers) — or, per unit time, the firing rate $1/T$. *(b)* Sweeping the injected current traces the *f–I curve* (firing rate vs. current): flat and zero below the *rheobase* (the smallest current that fires, red), then rising. This matches the intuitive "how many spikes" readout.

Primer on neuron electrophysiology. We can view a neuron as a device that turns an injected current into a voltage trace. At rest the membrane voltage V sits near -65 mV. A small (*sub-threshold*) injected current depolarises V a little and it relaxes back — a passive, RC-like response. A large enough (*supra-threshold*) current triggers an *action potential* or *spike*: voltage-gated Na^+ channels open regeneratively, V shoots to $\sim+40$ mV in under a millisecond, then K^+ channels open and pull it back down. Spikes are the neuron's output; their *count* (or rate) as a function of the injected-current amplitude is the *f–I curve* (frequency–current), the standard input–output summary of a cell: see Fig. 25.

Crucially, the behavior of the neuron depends on its inputs, as illustrated in Fig. 26. Here we show the voltage over time, under 3 different experimental conditions: a neuron stimulated with a 10 μA step signal, which generates repeating spikes (blue); the same neuron stimulated with a 2 μA step signal, which fails to trigger a response (dotted black); and the neuron modified by applying TTX blocker and then stimulated with a 10 μA step signal, which also fails to trigger a response (red line). This illustrates why experiment design is critical in this domain.

Primer on Hodgkin-Huxley models. In 1952, Alan Hodgkin and Andrew Huxley invented a model to explain the above behavior, based on their experiments with the giant axon of the squid. Since then, the model has been generalized and is widely used to mechanistically explain the spiking behavior of many kinds of neurons. Hodgkin and Huxley received the 1963 Nobel Prize in Physiology / Medicine for this work.

The model they came up with can be represented as an electric circuit, as shown in Fig. 27. This example contains Na, K, M and L ion channels, but the generalized model can contain different combinations of the 6 channels listed in Table 10, each of which have their own parameters and dynamics. We can write the generalized model as a set of nonlinear ODEs, which follow from

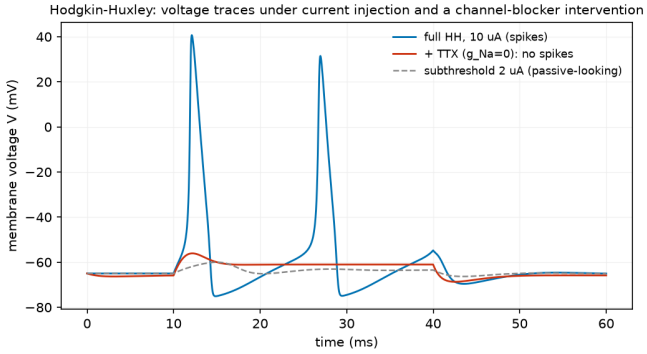


Figure 26: **Example spike traces from a single neuron under different conditions.** Membrane voltage under current injection: a supra-threshold step (10 μA) elicits overshooting action potentials (blue); the sodium blocker TTX (g_{Na}=0, a do on the mechanism) abolishes them (red); a sub-threshold current gives a passive response (grey).

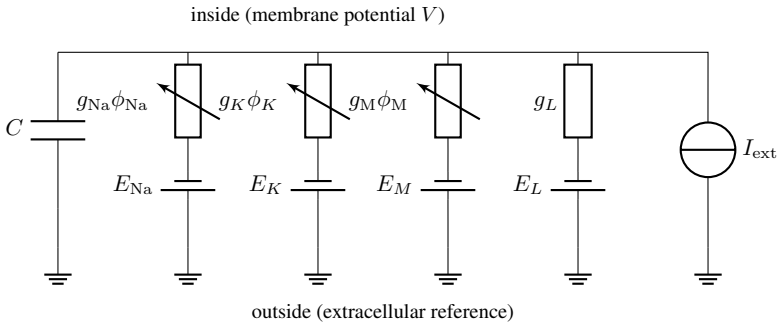


Figure 27: **The Hodgkin–Huxley equivalent circuit (Eq. (59)).** The membrane is a capacitor C; each ion channel is a branch with a variable conductance $g_c\phi_c$ (opening/closing gates ϕ_c) in series with a battery E_c (the reversal potential). The injected current I_{ext} charges the capacitor and flows through the open channels; a *blocker* deletes a branch ($g_c \to 0$). Which branches are present is the *structure*; the conductances g_c are the *parameters*.

Kirchoff's current law:

$$C\frac{dV(t)}{dt} = I_{\mathrm{ext}}(t) \; - \; \sum_{c\in\mathcal{C}} I_c(t) \tag{59}$$

$$I_c(t) = g_c\phi_c(t)(V(t) - E_c) \tag{60}$$

$$\phi_c(t) = m_c^{p_c}(t)\; n_c^{q_c}(t)\; h_c^{r_c}(t) \tag{61}$$

$$\frac{dx_c(t)}{dt} = \frac{T_{x,c}^{\infty}(V(t)) - x_c(t)}{\tau_{x,c}(V(t))},\; x \in \{m, n, h\} \tag{62}$$

Here C is the capacitance, $V(t)$ is the voltage, I_c is the current for channel c, $\mathcal{C}$ is the set of channels associated with this neuron, and $\phi_c(t)$ is the fraction of the channel that is open. Thus the current in the channel is given by $I_c = g_c\,\phi_c\,(V - E_c)$: (maximal conductance) $\times$ (fraction open) $\times$ (driving force). The fraction open ϕ_c (which changes over time) is based on a product of gating terms — denoted by m_c, n_c and h_c — each raised to an integer power (p_c, q_c, r_c; how many independent gates the channel has): see Table 9 for a list of gates, and Table 10 for a list of channels that uses these gates. Each such gating term x_c relaxes towards a voltage-dependent target $T_{x,c}^{\infty}(V)$ with its own time constant $\tau_{x,c}(V)$ (fast for activation, slow for inactivation), given by

$$T_{x,c}^{\infty}(V) = \frac{\alpha_x(V)}{\alpha_x(V) + \beta_x(V)} \tag{63}$$

$$\tau_{x,c}(V) = \frac{1}{\alpha_x(V) + \beta_x(V)} \tag{64}$$

where expressions for α_x and β_x can be found at https://en.wikipedia.org/wiki/Hodgkin-Huxley_model. As an example, the classic spiker is the following three-channel

gate	channel	role	target $T^{\infty}_{x,c}(V)$	speed
m_{Na}	Na^+	activation	rises with depolarisation	fast
h_{Na}	Na^+	inactivation	*falls* with depolarisation	slow
n_K	K^+	activation	rises with depolarisation	slow

Table 9: **The three classic Hodgkin–Huxley gates.** *Activation* gates (m_{Na}, n_K) open as the cell depolarises; the *inactivation* gate (h_{Na}) closes. Each relaxes to a voltage-dependent target $T^{\infty}_{x,c}(V)$ with its own time constant $\tau_{x,c}(V)$; the fast/slow separation between m_{Na} and $\{h_{\mathrm{Na}}, n_K\}$ is what generates and terminates the spike. Other channels (Table 10) carry their own gates m_c, n_c, h_c with the same form but different half-voltages and kinetics.

Channel	carries	current	role in the response	blocker
Na^+	sodium	$I_{\mathrm{Na}} = g_{\mathrm{Na}} m_{\mathrm{Na}}^3 h_{\mathrm{Na}} (V - E_{\mathrm{Na}})$	regenerative spike *upstroke*	tetrodotoxin (TTX)
K^+ (delayed rectifier)	potassium	$I_K = g_K n_K^4 (V - E_K)$	*repolarises* the spike	TEA
Ca^{2+} (high-threshold)	calcium	$I_{\mathrm{Ca}} = g_{\mathrm{Ca}} m_{\mathrm{Ca}}^2 h_{\mathrm{Ca}} (V - E_{\mathrm{Ca}})$	*alternative*, slower spike upstroke	cadmium (Cd)
M-type K^+	potassium	$I_{\mathrm{M}} = g_{\mathrm{M}}\, m_{\mathrm{M}} (V - E_K)$	slow; *spike-frequency adaptation*	XE991
A-type K^+ (transient)	potassium	$I_{\mathrm{A}} = g_{\mathrm{A}} m_{\mathrm{A}}^p h_{\mathrm{A}} (V - E_K)$	transient outward; *delays* firing onset	4-AP
leak	mixed	$I_L = g_L (V - E_L)$	sets the *resting potential*; passive	—

Table 10: **The voltage-gated ion channels — the building blocks.** A *blocker* is a drug that removes one channel by setting its conductance $g_c{=}0$; these are the mechanism-level interventions $\mathrm{do}(a)$ available on this rung (e.g. TTX abolishes Na^+-based spikes but not Ca^{2+}-based ones). The parenthetical drug names in this table refer to these blockers, and are what the design loop gets to apply.

model

$$C\dot{V} = I_{\mathrm{ext}} - g_{\mathrm{Na}} m_{\mathrm{Na}}^3 h_{\mathrm{Na}} (V - E_{\mathrm{Na}}) - g_K n_K^4 (V - E_K) - g_L (V - E_L) \tag{65}$$

Here the Na^+ channel carries an activation gate m_{Na} (cubed) and an inactivation gate h_{Na}, and the K^+ channel a single activation gate n_K (to the fourth). The names m, n, h are historical: what actually distinguishes a gate is its target curve $T^{\infty}_{x,c}(V)$ (whether it *opens* or *closes* as V rises) and its time constant $\tau_{x,c}(V)$. It is the *separation of timescales* — fast m_{Na} activation admitting Na^+ for the upstroke, before the slower h_{Na} inactivation shuts it off and the slower n_K activation repolarises — that makes the spike a transient, regenerative event.

Levels of abstraction. It is worth noting that HH is only one point on a spectrum of models at different levels of abstraction. There are more detailed stochastic models that capture individual cellular responses at a more granular level. There are also simplified models, such as the two-variable FitzHugh–Nagumo model, and the leaky integrate-and-fire model. Finally, if we set the membrane time constant to zero and binarise the output, we get the McCulloch–Pitts unit (McCulloch & Pitts, 1943), $y = \phi(\sum_i w_i x_i - b)$, which is the basis of artificial neural networks. So there is no single "true model". Instead, scientists seek the *coarsest valid causal abstraction* that is sufficient for the things they want to understand or predict (Beckers & Halpern, 2019; Rubenstein et al., 2017).

E.2 Our benchmark

We design a benchmark, NeuronBench, by creating 6 "mystery neurons", each composed of a plain Na+K+leak spiker plus one extra membrane mechanism, chosen from the list in Table 11: five are *novel* mechanisms and the sixth is a recallable textbook M-current control. Each of the five novel mechanisms is deliberately tuned to be *silent under every textbook probe*, i.e., the plain and novel neurons fire *identically* to standard current steps and channel blockers. This requires the agent to propose novel experimental protocols that it has not already memorized.

Mechanism	(g_Z, E_Z)	activation†	inact.†	behavioural signature (revealing protocol)
Z-REBOUND (I_Z)	$(4,\ +120)$	$(-57, 5, 4, 2)$	$(-88, 4, 130)$	spike-count collapse after a hyperpolarising conditioning pulse
H-SAG (I_h)	$(5,\ -30)$	$(-95, -5, 140, 1)$	—	voltage sag + post-inhibitory rebound on a hyperpolarising step
NA-FATIGUE	—	slow inactivation added to h_{Na}		use-dependent spike-count run-down over paired long pulses
CA-REBOUND (I_{CaT})	$(3.2,\ +120)$	$(-54, 6, 2, 2)$	$(-87, 4, 22)$	low-threshold rebound *burst* on release from hyperpolarisation
D-TYPE (I_D)	$(9,\ -77)$	$(-30, 10, 3, 1)$	$(-80, 5, 200)$	delayed / suppressed firing after a hyperpolarising pre-pulse
TEXTBOOK-M (I_M)	$(2.5,\ -77)$	$(-35, 10, 60, 1)$	—	spike-frequency adaptation on a long step (recallable by name)

Table 11: **The six worlds of NEURONBENCH**. Each current is added to a Na+K+leak spiker via Eq. (66). †the activation/inactivation columns are the tuples $(V_{1/2}, k, \tau, \text{power})$ and $(V_{1/2}, k, \tau)$ of Eq. (66). Conductances g_Z in $\mathrm{mS/cm^2}$; reversals E_Z, half-voltages and slopes in mV; time constants in ms; p, q are gate powers (q=1 when an inactivation gate is present, else 0). All are tuned to be *indistinguishable from the plain spiker under textbook steps and blockers* and separable only by the matched non-textbook protocol in the last column (a hyperpolarising conditioning pre-pulse for the de-inactivating currents). NA-FATIGUE adds no channel: it slows the inactivation of the existing Na^+ gate h_{Na}. The I_M control is a standard non-inactivating K^+ current the LLM *can* name and probe.

Specification of the novel channels. Each novel channel has roughly the same gated form as the textbook ones:

$$I_Z = g_Z\, m_Z^p\, h_Z^q\, (V - E_Z), \quad T^{\infty}_{m,Z}(V) = \sigma\Big(\tfrac{V - V^m_{1/2}}{k_m}\Big), \quad T^{\infty}_{h,Z}(V) = \sigma\Big(-\tfrac{V - V^h_{1/2}}{k_h}\Big), \tag{66}$$

with $\sigma(u) = 1/(1 + e^{-u})$ the logistic (Boltzmann) sigmoid, half-voltages $V^m_{1/2}, V^h_{1/2}$, slopes $k_m, k_h{>}0$, and fixed time constants τ_m, τ_h: activation rises with V and inactivation falls, while a *negative* activation slope $k_m{<}0$ instead makes the channel hyperpolarisation-activated (as for I_h), and an inactivation half-voltage $V^h_{1/2}$ below rest makes it *de-inactivated by hyperpolarisation* (available only after a hyperpolarising pre-pulse). This Boltzmann form is generic across the *novel* channels but is *not* the textbook parameterisation shown in Eq. (64), which are monotonic curves of the same qualitative shape but not identical logistic sigmoids.

The task The agent is told that it will be presented with some voltage trace data from a neuron of unknown type, and is asked to propose various candidate mechanisms (the exact prompts are shown in Section H.4). It is also given the menu of stimulation protocols (Table 12) and channel blockers, and a fixed experiment *budget*. From a handful of designed experiments it must *(i) propose its own* candidate mechanisms m and return a posterior $p(m \mid \mathcal{D})$ over them, and *(ii)* forecast the cell's response to held-out interventions it never ran. The truth is never revealed to the agent; it is used only for scoring.

Design space. At each step the agent controls the external current $I_{\mathrm{ext}}(t) = x_t$ by choosing one of the 9 **stimulation protocols** in Table 12: the action set here has exactly *nine* elements. (The benchmark also exposes a single channel blocker — tetrodotoxin (TTX) zeroing g_{Na}, TEA zeroing g_{K}, cadmium (Cd) zeroing g_{Ca}, or none — which would *nominally* give a $9{\times}4$ action set. But the novel mechanisms are by construction *silent under blockers*, exactly as under textbook steps: a blocker deletes a channel branch equally in the plain and novel cells, so it cannot separate them. The blockers are therefore non-discriminating for these worlds, and all runs use the 9 current-clamp protocols only — blockers remain available in the released benchmark and the interactive app but are unused here.)

Initial passive set $\mathcal{D}_0$ and budget. Every run is seeded with a *fixed* passive set $\mathcal{D}_0$ of 2 **experiments** — the two textbook steps present in every world's pool (a brief 12 μA/40 ms step and a long 10 μA/300 ms step, protocols 1–2 of Table 12) — so every agent starts from the same identifiable

#	protocol	segments (Δt ms, I μA)	probes
1	brief step	$(40, 12)$	fast onset
2	long step	$(300, 10)$	spike-frequency adaptation
3	strong step	$(120, 18)$	high-rate firing
4	weak step	$(120, 5)$	near-threshold f–I
5	hyperpol. conditioning + test	$(250, -30), (150, 12)$	de-inactivation / depol. block
6	hyperpol. pre-pulse + weak test	$(250, -30), (120, 0), (60, 6)$	rebound at low drive
7	paired long pulses	$(300, 12), (60, 0), (300, 12)$	use-dependence / slow inactivation
8	depol. conditioning + test	$(250, 15), (150, 12)$	depolarising history
9	brief hyperpol. conditioning + test	$(40, -30), (150, 12)$	fast de-inactivation

Table 12: **The nine-protocol menu of external currents that can be applied** over which VoI is enumerated on NEURONBENCH. Each protocol is a sequence of (duration, amplitude) current segments; a leading hyperpolarising segment is a conditioning pre-pulse. Rows 1–4 are standard current-clamp steps; rows 5–9 are the non-textbook protocols that expose the hidden mechanisms of Table 11.

baseline before VoI takes over; the agent then *designs* N_a further experiments. We run only a *small* budget ($N_a \leq 5$): the discrete menu is small (9 protocols, so $N_a \leq 7$ before it is exhausted), and each *stochastic* evaluation is an expensive SMC^3 rollout (the inner particle filter tracks N_z=250 latent gating paths; see Table 2 for the full SMC settings). Because NEURONBENCH is our own benchmark — we do not compare against external methods on it — a small budget already exhibits the design/inference gains we study, so we cap it for cost.

Evaluation (the query set $\mathcal{Q}$). The agent is scored on a fixed *held-out query set* $\mathcal{Q}$: the benchmark's 6 *test protocols*, a curated set *disjoint* from the 9-protocol design menu (so the agent is always forecasting interventions it never ran). These test protocols are chosen to be discriminative — each is a stimulation sequence under which the candidate mechanisms disagree.

Evaluation (the target features $F(y_{1:T})$). Because a spike is a $\sim$1 ms all-or-none event, a sub-millisecond timing mismatch between model and data produces a $\sim$100 mV pointwise error even for an essentially correct model. Thus, rather than asking agents to predict the exact voltage trajectory $y_{1:T}$ in response to a novel perturbation, we just ask it a target functional $F(y_{1:T})$ of summary statistics, which include spike counts in the test and conditioning windows, their use-dependent run-down, within-pulse adaptation, and two sub-threshold voltage summaries, as illustrated in Fig. 28. These are computed as follows:

$$F(y) = \Big(\underbrace{n_{\text{test}}}_{\text{test spikes}}, \quad \underbrace{n_{\text{pre}}}_{\text{pre-pulse spikes}}, \underbrace{n_{\text{pre}} - n_{\text{test}}}_{\text{run-down}}, \underbrace{n_{\text{test}}^{\text{early}} - n_{\text{test}}^{\text{late}}}_{\text{adaptation}}, \underbrace{\min_t V(t)}_{V_{\min}}, \underbrace{\bar{V}_{\text{end}}}_{\text{steady state}} \Big), \tag{67}$$

where $n_{\text{test}}, n_{\text{pre}}$ count upward zero-crossings of V in the test window (after any conditioning pre-pulse) and before it, $n_{\text{test}}^{\text{early}} - n_{\text{test}}^{\text{late}}$ splits the test window in half, and $\bar{V}_{\text{end}}$ is the mean voltage over the steady-state tail of the trace (the final few percent, after the stimulus ends). This is chosen so that both the rate-signature worlds (NA-FATIGUE, TEXTBOOK-M) and the sub-threshold/burst worlds (H-SAG, CA-REBOUND) leave a signal. Given this target, we compute the MSE as in Eq. (3), which we normalize as in Eq. (1).

Interactive app. Figure 29 shows a screenshot for a web app we built that lets users try this benchmark for themselves. The app is available at https://github.com/murphyk/neuronbench.

E.3 THE AGENT'S MODEL

The agent models the environment using an SSM, as shown in Eq. (8). We give the details below.

E.3.1 SUMMARY STATISTICS AND SYNTHETIC LIKELIHOOD

As we discussed when defining the target feature vector $F(y_{1:T})$ in Eq. (67), it can be useful to summarize the raw voltage trace into a low-dimensional vector of statistics. This is true for inference

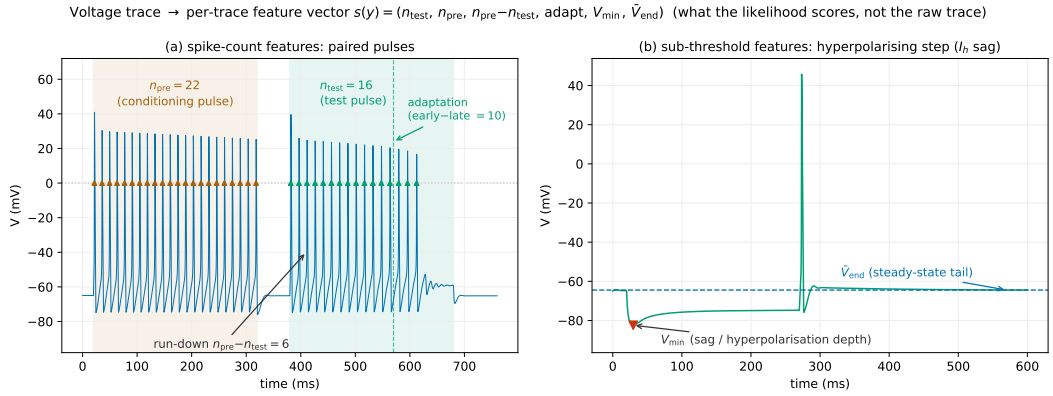


Figure 28: **From a raw voltage trace to the per-trace feature vector $s(y)$ of Eq. (67)**, computed on real NEURONBENCH traces. *(a)* On a paired-pulse protocol the spike-count features are the test- and pre-pulse counts (n_{test}, n_{pre}; upward 0 mV crossings, triangles), their use-dependent *run-down* $n_{\text{pre}}-n_{\text{test}}$ (here the NA-FATIGUE (slow-Na) cell fires less on the second pulse), and the within-pulse *adaptation* (early-half minus late-half of the test window, dashed divider). *(b)* On a hyperpolarising step the sub-threshold features are the voltage minimum $V_{\min}$ (the I_h sag / hyperpolarisation depth) and the steady-state tail $\bar{V}_{\text{end}}$.

Eq. (8)	meaning	instantiation
z_t	latent state	$\big(V(t),$ gating variables $m, h, n, \dots\big)$
y_t	observation	$V(t)$ + noise (voltage recorded; gates hidden)
x_t	exogenous input	injected current $I_{\text{ext}}(t)$ — the stimulus protocol
θ	mechanism parameters	maximal conductances $\{g_c\}$ (and gating kinetics)
m	discrete structure	channel composition
$\text{do}(a)$	intervention	(i) choose the protocol x_t; (ii) blocker: set a g_c=0
Y_a	query / outcome	forecast the spike response to a held-out protocol

Table 13: **The state-space model of Eq. (8) instantiated for NEURONBENCH**. This defines the notation.

as well as evaluation. When performing inference, this is called a *summary vector*, and will be denoted $s(y_{1:T})$. For simplicity, we can set $s(y_{1:T}) = F(y_{1:T})$, but in general they can be different. We say that a summary vector is a *sufficient statistic* if $p(m, \theta|\mathcal{D}) = p(m, \theta|s(D))$. In general our summaries may be insufficient (lossy), but can still be useful for computational reasons. In particular, instead of passing the raw trace $y_{1:T}$ to the LLM (either as part of MDA's proposal distribution, or the ICL baseline), we pass in $s(y_{1:T})$ instead, since LLMs are not very good at dealing with long sequences of raw numbers.

When it comes to Bayesian inference, we can replace the likelihood $p(y_{1:T}|m, \theta)$ with an approximation, $p(s(y_{1:T})|m, \theta)$. One approach to estimating this is known as "Bayesian Synthetic Likelihood" (Wood, 2010; Deistler et al., 2025), which proceeds as follows. For each candidate (m, θ) we draw R traces $(z^{(r)}, y^{(r)}) \sim p(z_{1:T}, y_{1:T} \mid m, \theta)$, throw away z^r, and compute the likelihood using

$$p\big(s(y_{1:T}) \mid m, \theta\big) = \mathcal{N}\big(s(y_{1:T}) \,\big|\, \mu_{m,\theta}, \Sigma_{m,\theta}\big) \tag{68}$$

$$\mu_{m,\theta} = \tfrac{1}{R} \textstyle\sum_r s(y^{(r)}) \tag{69}$$

$$\Sigma_{m,\theta} = \widehat{\text{Cov}}_r\big[s(y^{(r)})\big] + \varepsilon I, \tag{70}$$

with a small ridge ε for conditioning. We can also replace the Gaussian with a neural network normalizing flow model, a technique called *neural likelihood estimation* (Papamakarios et al., 2019).[13]

Instead of manually specifying $s(y_{1:T})$, we can also learn it from data. A simple approach, known as *semi-automatic ABC* (Fearnhead & Prangle, 2012; Jiang et al., 2017), proceeds as follows: draw

[13]NLE is not to be confused with *neural posterior estimation* (Greenberg et al., 2019), which trains an amortized inference network to compute $p(m, \theta|y_{1:T})$. However, this requires generating m and θ, both of which can be hard (especially if the model m is code). In addition, we cannot extract the evidence Z_m from this amortized posterior, so we cannot do model selection. See (Cranmer et al., 2020; Frazier et al., 2024) for more discussion.

Figure 29: **NEURONBENCH**. Screenshot of our app, which lets users interact with the same environment we give our agents (except the agents see numerical data, not images.) The top left is the training set, $\mathcal{D}_{\text{tr}}$, the top right is the test set, $\mathcal{D}_{\text{te}}$, and the bottom row is the interactive environment. The agent can choose a sequence of input currents $x_{1:T}$ by specifying the magnitude and duration of a step pulse (shown in orange). The agent can also choose from a finite set of interventions, corresponding to blocking different ion channels (shown as white boxes). The resulting output voltage $y_{1:T}$ is shown in the green trace. App is available at https://github.com/murphyk/neuronbench.

prior samples $\{(\theta^r, m^r, z^r, y^r_{1:T})\}$ from the model, then train a neural network regression model to predict m_r and θ_r given $s_\phi(y_{1:T})$. For example, we can let $s_\phi(y_{1:T})$ be a 1D CNN applied to the time series following by global average pooling, and then pass this embedding into an MLP $f_w(s)$ with two output heads, one for classifying m and one for predicting the mean of θ_r:

$$(\hat{m}, \hat{\theta}) = (f^m_w(s_\phi(y_{1:T})), f^\theta_w(s_\phi(y_{1:T}))) \tag{71}$$

We then pass $s_\phi(y_{1:T})$ into the above BSL method to compute the likelihood, which is passed to SMC to compute the evidence Z_m. We have some positive preliminary results with this approach, but we leave detailed exploration to future work.

Unfortunately the SA-ABC approach will not work for complex outputs, such as when the model m is code, since it must generate them (or at predict their expected value). In this case, a better alternative is to use *neural ratio estimation* (see e.g., (Hermans et al., 2020; Durkan et al., 2020)), which trains a binary classifier (on forwards-sampled data) to approximate the *likelihood-to-evidence ratio*

$$r_\phi(y, m, \theta) \;=\; \frac{d_\phi}{1 - d_\phi} \;\approx\; \frac{p(y_{1:T} \mid m, \theta)}{p(y_{1:T})}, \tag{72}$$

This conditions on m and θ, so it is a discriminative model, which is much easier to train. We leave exploration of this to future work.

In summary, for NEURONBENCH and NEURONBENCHSTOCH, we use the summary features $s(y_{1:T}) = F(y_{1:T})$ just as input to the LLM and use the raw trace when performing inference with SMC. For all the other benchmarks, we always use the raw data for inference (so $s(y) = y$) and evaluation (so $F(y) = y$).

Quantity	Symbol	Prior / value	Role
Fixed excitable backbone (canonical Hodgkin–Huxley, not inferred):			
Na^+ conductance	g_{Na}	120 (fixed)	spike upstroke
K^+ (delayed rect.)	g_{K}	36 (fixed)	repolarisation
capacitance	C	1.0 μF/cm^2	membrane
reversal potentials	$E_{\mathrm{Na/K/L}}$	$+50/-77/-54.4$ mV	driving forces
gating-curve *forms*	$T^{\infty}_{x,c}(V), \tau_{x,c}(V)$	Hodgkin–Huxley forms	channel identity
Inferred — shared leak nuisance (uniform prior):			
leak conductance	g_{L}	$\mathcal{U}(0.2,\ 0.45)$	resting potential
Inferred — the one extra *channel named by m (uniform priors; g in mS/cm^2, $V_{1/2}$ in mV, τ in ms):*			
I_h (hyperpol. inward)	$g, V_{1/2}, \tau$	$\mathcal{U}(2,10)$, $\mathcal{U}(-105,-85)$, $\mathcal{U}(60,240)$	sag / rebound
T-type Ca^{2+}	$g, V_{1/2}, \tau_h$	$\mathcal{U}(1.5,8)$, $\mathcal{U}(-65,-45)$, $\mathcal{U}(12,180)$	rebound spike
D-type K^+ (de-inact.)	$g, V_{1/2}, \tau_h$	$\mathcal{U}(4,14)$, $\mathcal{U}(-40,-20)$, $\mathcal{U}(100,300)$	onset delay
M-type K^+ (slow)	$g, V_{1/2}, \tau$	$\mathcal{U}(1,6)$, $\mathcal{U}(-45,-25)$, $\mathcal{U}(30,90)$	spike-freq. adaptation
slow Na^+ inactivation	τ_s	$\mathcal{U}(200,600)$	use-dependent run-down

Table 14: **Parameters and priors for NEURONBENCH.** The canonical Na^+/K^+ spiking backbone is held fixed at textbook Hodgkin–Huxley values (g_{Na}=120, g_{K}=36 mS/cm^2), so discovery targets the unidentified *extra* channel that shapes firing: for the structure m present, the agent infers the shared leak conductance and that channel's conductance, half-activation $V_{1/2}$, and (in)activation time constant τ, each under a *uniform* prior over the broad bounds shown. The SMC uses N_p=48 particles (the stochastic worlds use more; Table 2), target ESS =0.6 N_p, 3 random-walk-Metropolis moves per tempering rung (Gaussian proposal SD = 0.5× the current particle SD, clipped to the bounds), a 2.0-nat per-parameter Occam penalty, and the full-trace Gaussian observation model of Eq. (14) with per-sample noise sd = $\max(0.12\,|\cdot|,\ 1)$ (spike count in spikes; voltage samples standardised so 1 unit =3 mV).

E.3.2 EXPERIMENT DESIGN ALGORITHM

Since the design space, discussed in Section E.2, is discrete, we can maximize the VoI exactly by enumerating all options and picking the best. We either use the EIG criterion in Eq. (26) or the task-driven VoI criterion in Eq. (40).

E.3.3 PARAMETERS AND THEIR PRIORS

Table 14 lists the unknown parameters and the prior we use for them. The excitable Na^+/K^+ backbone is held *fixed* at textbook Hodgkin–Huxley values, so discovery targets the unidentified *extra* channel that shapes firing. For the channel named by a candidate structure, the free parameters are its maximal conductance, half-activation voltage, and (in)activation time constant, together with a shared leak conductance; each is given an independent *uniform* prior over the broad bounds in Table 14, deliberately broad since the target is a real cell of unknown size. The membrane capacitance, reversal potentials, and the *functional form* of the gating curves $T^{\infty}_{x,c}(V), \tau_{x,c}(V)$ are fixed by the archetype, and the feature-kernel tolerances σ_j are the fixed observation model.

E.4 RESULTS

Figure 4a shows the aggregate results of MDA using three ways of designing experiments: random, EIG, and task-driven. See that all MDA methods significantly beat ICL. We see that the two VoI methods are consistently better than random design. We also see that EIG is (slightly) better than task-driven design at low budget, since in the deterministic setting identifying the true model already yields an accurate forecast of the target, leaving no nuisance direction for the task-aware objective to exploit. (By contrast, in the stochastic case, which we discuss in Section F, the latent carries forecast-irrelevant nuisance directions, and task-driven VoI *beats* EIG.)

Figure 30 breaks this aggregate down by world. The EIG-vs-task ordering is *not* uniform: task-driven design wins several worlds (`ca_rebound`, `textbook_M`, `h_sag`) while EIG wins `d_type` most decisively (and edges `na_fatigue`), with `z_rebound` a noisy near-tie; these differences cancel in the mean — consistent with $\mathrm{VoI}_U \neq \mathrm{EIG}$ being a regime-dependent effect rather than a uniform win. Both design policies beat random design in every world.

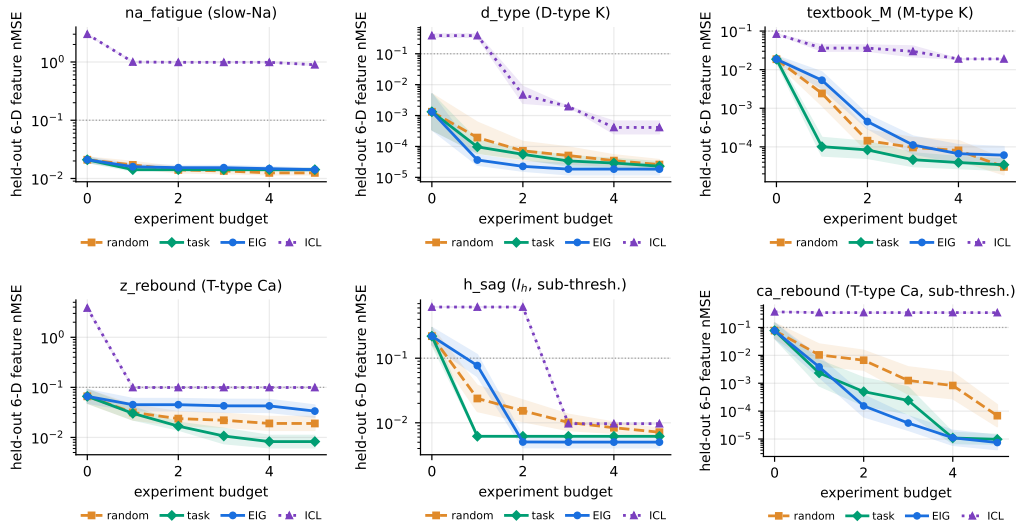


Figure 30: **Per-world deterministic NEURONBENCH data efficiency** (held-out 6-D feature nMSE vs. experiment budget; geo-mean ± 1 SE over 12 seeds) — the per-world decomposition of Fig. 4a. EIG (blue), task-driven (green), and random (orange) design, plus the ICL (LLM-in-context) yardstick (purple, dotted), which forecasts the same 6-D feature vector directly and sits one-to-two orders of magnitude above the model-based policies in every world. The EIG-vs-task ordering is world-dependent: task wins `ca_rebound`/`textbook_M`/`h_sag`, EIG wins `d_type` (and `na_fatigue`), `z_rebound` is a near-tie; both designs beat random throughout. The two sub-threshold worlds (`h_sag`, `ca_rebound`) carry a hidden channel invisible to spike counts, so their feature forecast improves only once the excitable baseline is fit.

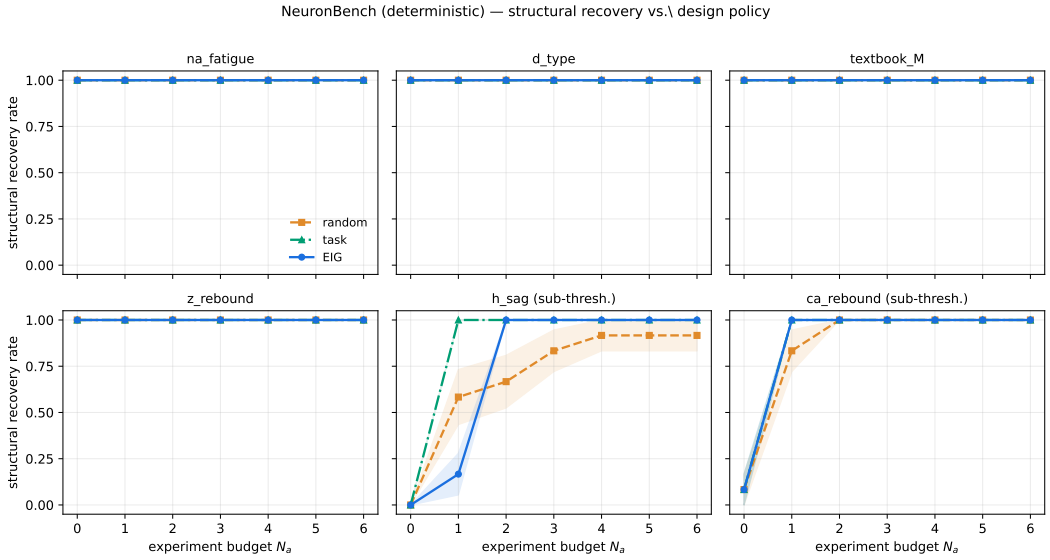


Figure 31: **Deterministic NEURONBENCH: structural recovery vs. design policy.** The companion to Fig. 30, scored by *structural recovery* (best-so-far fraction of seeds whose discovered channel matches the true channel class) for EIG (blue), task-driven (green), and random (orange) design. Recovery saturates to $\sim$100% within one or two experiments for all policies — the full-trace likelihood makes the channel easy to identify once probed — so the design policy separates *less* on recovery than on prediction here; task-driven design still reaches full recovery fastest.

Figure 31 shows the structural recovery per world. We see that the underlying identity of the channel is identified very quickly (because the 6d summary features on the clean trace are very informative), so the remaining experimental budget is spent nailing down the parameters.

F NEURONBENCHSTOCH: FURTHER DETAILS

The worlds in Section E use a *deterministic* Hodgkin–Huxley forward model, so the likelihood is available in closed form (Eq. (14)). Real neurons are stochastic: with a finite number of ion channels, gating fluctuates (channel noise), and the latent dynamics become an SDE. We therefore create a stochastic version of our benchmark, which we call NEURONBENCHSTOCH. This is the regime real experiments occupy, and the one setting our other benchmarks (deterministic ODEs + observation noise) do not exercise.

F.1 STOCHASTIC VERSION OF HODGKIN-HUXLEY

To create a stochastic neuron, we add finite-N_{noise} channel noise via the Fox–Lu diffusion approximation (Fox & Lu, 1994), with the channel count N_{noise} tuning the intrinsic noise from near-deterministic ($N_{\text{noise}} \to \infty$) to strongly stochastic.

In more detail, each gate x_c is really an ensemble of N_{noise} two-state ion channels, each switching open $\leftrightarrow$ closed as a continuous-time Markov chain with the voltage-dependent rates $\alpha_x(V), \beta_x(V)$ of Eq. (64); the deterministic HH gating ODE is the $N_{\text{noise}} \to \infty$ mean-field limit of the open fraction. The *Fox–Lu* diffusion approximation (Fox & Lu, 1994) keeps finite N_{noise} by replacing that mean field with a Langevin (stochastic differential) equation — the deterministic drift plus a Gaussian channel-noise term whose variance scales as $1/N_{\text{noise}}$:

$$dx_c = \left[\alpha_x(V)(1-x_c) - \beta_x(V)\, x_c\right] dt + \sqrt{\frac{\alpha_x(V)(1-x_c)+\beta_x(V)\, x_c}{N_{\text{noise}}}}\; dW_t, \qquad x \in \{m, n, h\}, \quad (73)$$

with dW_t an independent Wiener increment per gate. The diffusion coefficient is the sum of the two transition fluxes divided by N_{noise} (the system-size / Ω-expansion correction to the channel master equation), so more channels means smaller fluctuations and $N_{\text{noise}} \to \infty$ recovers the deterministic gate. We integrate Eq. (73) by Euler–Maruyama and substitute the noisy gates into the membrane equation (59), making N_{noise} a single knob from near-deterministic to strongly stochastic. Fox–Lu is the standard cheap channel-noise model; see Goldwyn & Shea-Brown (2011) for how it compares to exact Markov-chain channel simulation.

F.2 THE BENCHMARK

We convert the deterministic six-world NEURONBENCH (Section E.2) into a stochastic form, NEURONBENCHSTOCH, changing only what a finite channel count forces — the worlds, the hidden mechanisms, the design pool, and the scoring are otherwise inherited unchanged. Relative to the deterministic benchmark the differences are:

- **Stochastic latent dynamics.** The gates evolve by the Fox–Lu SDE (Eq. (73)) rather than the deterministic HH ODE (Eq. (59)), with the channel count N_{noise} as a single noise knob. We sweep a *noise ladder* $N_{\text{noise}} \in \{50, 100, 300, 1000, 3000\}$ from strongly stochastic to near-deterministic; unless noted we use N_{noise}=100 (fairly noisy), and $N_{\text{noise}} \to \infty$ recovers the deterministic benchmark.
- **Partial, noisy observation.** An experiment returns the membrane voltage with additive Gaussian noise (σ=2 mV), sub-sampled every $\sim$4 ms. The deterministic benchmark *also* returns a sub-sampled trace, so sub-sampling itself is not the difference; what changes is that the trace now carries observation noise and is sampled $\sim 40\times$ more coarsely ($\sim$4 ms vs. the deterministic $\sim$0.1 ms), the two together making $p(y \mid m, \theta)$ intractable.
- **Intractable likelihood.** The latent path must be marginalised (see Eq. (75)), so the closed-form Gaussian likelihood (Eq. (14)) of the deterministic benchmark is unavailable.

Everything else is unchanged: the same six worlds and hidden mechanisms, the same *disjoint* held-out set of 6 test protocols the agent never runs, and the same **feature-forecast MSE** on the target vector $F(y)$ of Eq. (67) — now evaluated against the *noisy* cell, with each held-out target estimated as the mean over 200 independent stochastic rollouts. (However, see the section below where we propose an additional evaluation metric.)

Scoring a stochastic forecaster. Because the cell is now stochastic, a forecast is a *distribution*, not a point. We can go beyond computing the mean of the forecast by comparing the agent's distribution $\hat{P} = p(\cdot \mid \hat{m}, \hat{\theta})$ to the oracle's $P^\star = p(\cdot \mid m^\star, \theta^\star)$, computing the **energy distance** between them:

$$D_E(\hat{P}, P^\star) = 2\,\mathbb{E}\,\|X - Y\| - \mathbb{E}\,\|X - X'\| - \mathbb{E}\,\|Y - Y'\|, \qquad X, X' \sim \hat{P},\ \ Y, Y' \sim P^\star, \tag{74}$$

This is the population form of the *energy score* — a strictly proper scoring rule, the multivariate generalisation of the CRPS: $D_E \geq 0$, with $D_E{=}0$ iff the two predictive distributions coincide. For the agent, we compute this by sampling from the full posterior over (m, θ) rather than a single point estimate, as well as sampling the path $z_{1:T}$. We use common random numbers shared between the agent's and the oracle's rollouts for z, which cancels the Monte-Carlo estimation floor and leaves only genuine mechanism discrepancy.[14] We estimate it by the U-statistic over the R paired rollouts, on the target $F(y)$. Unlike the mean nMSE it penalises a forecaster whose *spread* or *shape* is wrong even when its mean is right — e.g. a mechanism that matches the expected spike count but not its trial-to-trial variability — and, being computed from both cells' own rollouts, it directly measures how close the discovered cell's predictive *distribution* is to the true cell's, not merely its average behaviour.

However, on NEURONBENCHSTOCH the energy distance and the feature nMSE turn out to be nearly equivalent (modulo scaling). This is because the agent's and the oracle's predictive *spreads* are similar (same Fox–Lu noise model, same R rollouts), so the within-distribution terms of Eq. (74) roughly cancel and D_E is dominated by the difference in means — exactly what the nMSE measures — so $D_E \approx \sqrt{\text{nMSE}}$ (empirically $\text{corr}(D_E, \text{nMSE}){=}0.99$ over 12 seeds $\times$ 6 worlds). The energy distance would separate from the nMSE only for a forecaster that matched the mean feature but not its trial-to-trial variability, which none of our methods do; we therefore report only the feature nMSE in the figures.

F.3 LIKELIHOODS

The (observed data) likelihood for model m is given by

$$p(y_{1:T} \mid m, \theta) = \int p(y_{1:T} \mid z_{0:T}, m, \theta)\; p(z_{0:T} \mid m, \theta)\; dz_{0:T}, \tag{75}$$

where $y_{1:T}$ is the observed voltage trace and $z_{0:T}$ the latent gating path. This requires marginalising over the stochastic latent path $z_{0:T}$ — a high-dimensional path integral with no closed form, because the Fox–Lu transition density $p(z_t \mid z_{t-1})$ is itself intractable. Below we discuss how to approximate this integral using a bootstrap particle filter (Algorithm 4), as well as various other faster approximations.

Particle filtering. The agent fits a stochastic state-space model (8) where the latent state $z_t = (V_t, \{x_c(t)\})$ (voltage and gates) evolves by the discretised Fox–Lu transition $p(z_t \mid z_{t-1}, a)$ of Eqs. (59) and (73), and the voltage is observed with Gaussian noise, $y_t \sim \mathcal{N}(V_t, \sigma^2)$. Candidate models m differ in *structure* (which channels are present) and in the conductances θ; the channel count N_{noise} (the noise scale) is a known part of the model here. The one-step transition density $p(z_t \mid z_{t-1}, a)$ has no closed form — it is a nonlinear diffusion over the interval — but the bootstrap particle filter never needs it. It only *samples* the transition (one Euler–Maruyama step, i.e. a Gaussian draw on the gates, Eq. (73)) as its proposal, and only *evaluates* the tractable observation density $\mathcal{N}(y_t \mid V_t, \sigma^2)$ to reweight the particles, from which the marginal likelihood Z_m can be estimated. So the intractable-likelihood regime needs only a *simulator* of the latents plus an evaluable observation model, exactly what a mechanistic ODE/SDE provides — no transition density is ever computed.

Why the deterministic likelihood breaks. To illustrate why we cannot just use a deterministic ODE model (and hence a deterministic likelihood, as we did in Eq. (13)), we consider a simple example where we need to distinguish just two hypotheses: a plain Na/K cell vs the novel H-SAG model I_h defined in Table 11. The noisy voltage traces from the two hypotheses are shown in

[14] Strictly, sharing random numbers estimates a *coupled* energy statistic rather than the marginal energy distance of Eq. (74) (whose strict propriety assumes independent samples); we use it as a variance-reduced diagnostic — it is 0 when the mechanisms coincide and tracks the feature nMSE ($\text{corr}{=}0.99$) — not as a calibrated proper score.

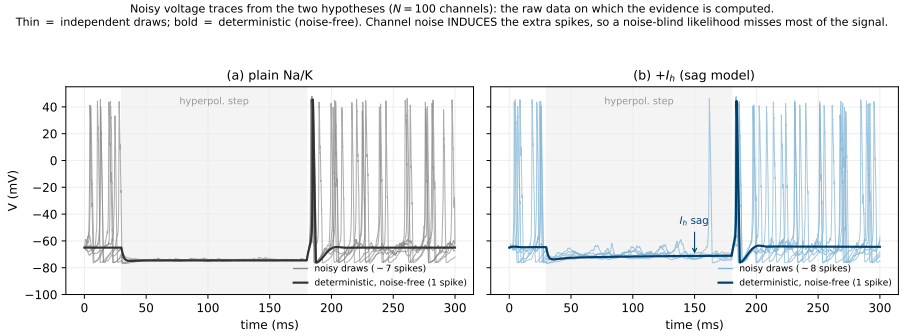


Figure 32: **Stochastic-latent NEURONBENCH: the raw data.** Noisy voltage traces from the two competing hypotheses under a moderate hyperpolarising-step protocol, at N_{noise}=100 channels (thin: independent draws; bold: the *deterministic*, noise-free trace). *(a)* A plain Na/K cell. *(b)* The same cell plus a hyperpolarisation-activated I_h current (the H-SAG model of Table 11), whose only signature is a small depolarising *sag* during the step (arrow). Because that sag is comparable in size to the channel noise, the two hypotheses overlap and cannot be told apart by eye. Note that channel noise *induces* spiking: the deterministic trace fires once where the noisy cell fires ∼7 times, so a noise-blind (deterministic) likelihood misses most of the signal — the failure mode quantified in Fig. 33.

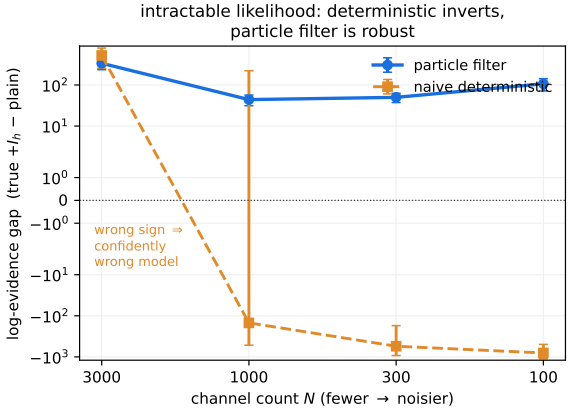


Figure 33: **Stochastic-latent NEURONBENCH: estimating the intractable likelihood by simulation.** We score the I_h-vs-plain decision on the data of Fig. 32, sweeping the channel count N_{noise} (fewer = noisier): the log-evidence gap $\log Z_1 - \log Z_2$ vs. N_{noise}. A likelihood that ignores the process noise (a single deterministic rollout + Gaussian observation, orange) degrades and, below N_{noise}≈1000, *inverts* — a negative gap means it confidently selects the *wrong* mechanism. A bootstrap particle filter (blue), which estimates $p(y \mid m, \theta)$ by propagating the latent gating SDE, stays robustly positive. Bars/points are ±1 SE over independent noise realisations.

Fig. 32 — the I_h sag is subtle relative to the channel noise, so they overlap. In Fig. 33 we plot the log-evidence gap, $\log Z_1 - \log Z_2$, vs noise level N_{noise}, where Z_1 is the evidence for the I_h hypothesis and Z_2 for the alternative Na/K hypothesis. We see that a likelihood that treats the intrinsic channel noise as zero degrades as the noise grows and, at N_{noise}=1000 channels, *inverts*: it confidently selects the wrong mechanism. By contrast, the particle filter stays robustly correct at every noise level.

F.4 PARAMETERS AND THEIR PRIORS

The parameters and priors are as in the deterministic NEURONBENCH (Section E.3.3): conductance-based mechanisms with uniform priors over the declared parameter bounds (Table 14) and a Bayesian-Occam structure prior over the parameter count. The only addition is the *channel-noise* scale of the Fox–Lu gating SDE (fixed, not inferred); because it makes the likelihood intractable, we estimate the evidence with the bootstrap particle filter (N_z=250; Algorithm 4), which renders the parameter level pseudo-marginal, giving the full SMC^3 used here.

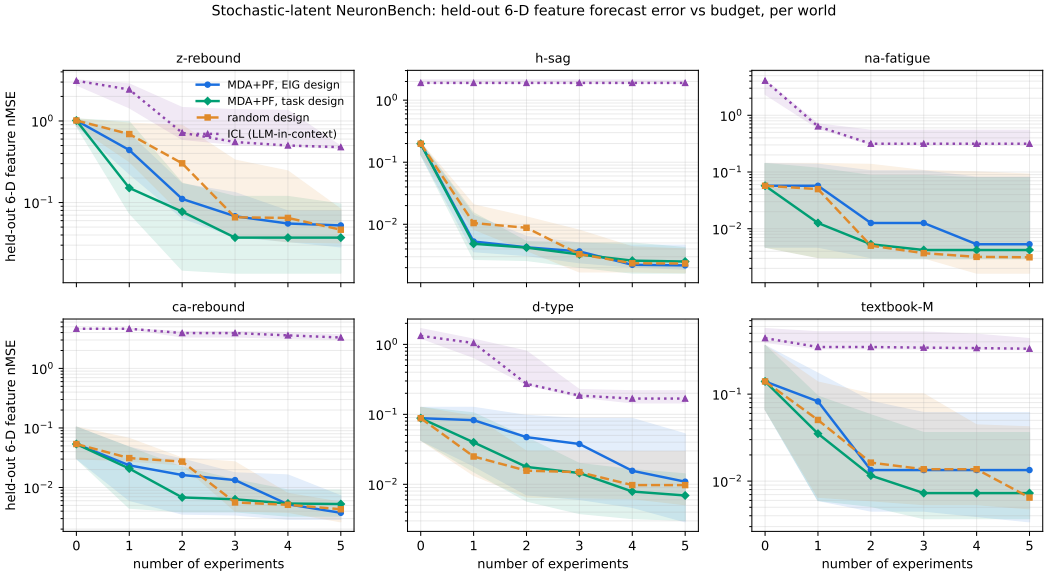


Figure 34: **Open-world stochastic NEURONBENCH: per-world learning curves.** Held-out 6-D feature-forecast nMSE vs. the budget N_a for each of the six worlds (N_{noise}=100; median over 12 seeds; task-aware VoI in green, model-discrimination EIG in blue, random design in orange, the in-context LLM baseline dotted; log scale). All model-based policies sit $\sim$30$\times$ below the LLM baseline in every world. Task-aware design helps most where the *design policy* matters — D-TYPE, TEXTBOOK-M, and (early) Z-REBOUND— the $\text{VoI}_U \neq \text{EIG}$ effect, and ties EIG on the clean worlds (H-SAG, CA-REBOUND); NA-FATIGUE is SNR-limited and all policies cluster.

F.5 RESULTS

Figure 4b shows the nMSE of different methods aggregated over the six worlds, reporting the *best-so-far* error (the error of the best mechanism the loop has found by each budget — the standard experimental-design summary), so the curve isolates sample efficiency from the per-world identifiability floor discussed next. Designed (VoI) experiments reach a low held-out error within one to two experiments — about twice as fast as random design at budgets 1–2 — both converging to $\sim 10^{-2}$ nMSE, while the in-context LLM forecaster plateaus 30–100$\times$ higher.

Figure 34 breaks this out per world (raw, not best-so-far). On three worlds (H-SAG, Z-REBOUND, CA-REBOUND) the designed loop drives the held-out feature error down sharply within one to two experiments. On the three confusable / noise-limited worlds (TEXTBOOK-M, D-TYPE, NA-FATIGUE) it does *not* improve monotonically: the posterior concentrates on mechanisms that explain the collected discriminative protocols but leave a residual error on the disjoint held-out features. This is a genuine identifiability floor, not a point-estimate artefact — it survives the posterior-predictive readout unchanged — so more experiments cannot lower it, even though structural recovery on these worlds remains robust.

Figure 35 shows the structural recovery per world. Unlike the deterministic case, we see that there is genuine uncertainty about the latent mechanism's identity; this gets resolved over multiple experiments, except for na-fatigue, which remains ambiguous. Interestingly, in d-type, both the task-VoI and random designs beat EIG.

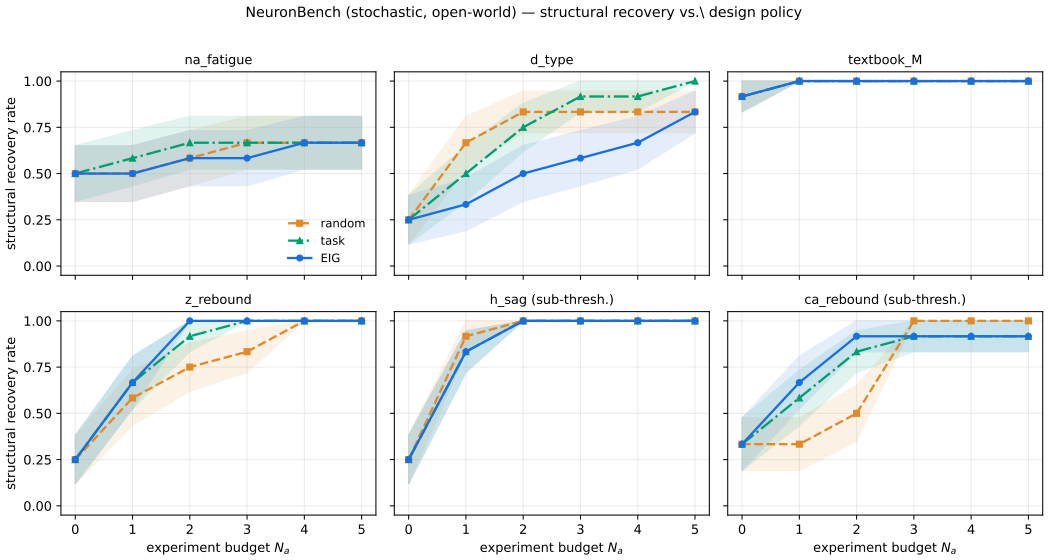


Figure 35: **Open-world stochastic NEURONBENCH: structural recovery vs. design policy.** As Fig. 34 but scored by structural recovery (best-so-far fraction of 12 seeds whose discovered channel matches the truth). Unlike the deterministic case, recovery does *not* saturate, and the design policies separate per world: task-driven design leads on D-TYPE and CA-REBOUND, EIG on Z-REBOUND, and NA-FATIGUE stays at its SNR-limited plateau — the same $\mathrm{VoI}_U \neq \mathrm{EIG}$ regime-dependence seen in prediction, but more visible on recovery.

G FURTHER RELATED WORK

Here we expand on connections to prior work that we did not have space for in Section 5.

G.1 OTHER INTERACTIVE LEARNING BENCHMARKS

Various benchmarks have been developed to evaluate agents that learn about a world by *interactive experimentation*. In this paper, we build on DISCOVERPHYSICS (Wiemann et al., 2026), ACTIVESCIBENCH-CHEM (Kabra et al., 2026), and BOXINGGYM (Gandhi et al., 2025). In the future we plan to investigate SCIGYM (Duan et al., 2025), a systems-biology dry lab in which an agent recovers the missing reactions of a curated SBML network under a fixed experiment budget, scored on both structural recovery and simulated-trajectory error — the same design-plus-recovery axes we study here.

Another notable benchmark is the ARC-AGI-3 challenge (ARC Prize Foundation, 2026). In this challenge, the agent interacts with a 2d grid world, by controlling the input (keyboard / mouse) sequence $x_{1:T}$ and observing the output sequence of images, $y_{1:T}$. This constitutes its training set, $\mathcal{D}_{\text{tr}} = \{(x^r, y^r)\}$. It is then given a test input, x_{test}, and must predict the corresponding output distribution $p(y_{\text{test}}|x_{\text{test}}, \mathcal{D}_{\text{tr}})$. This is a classic transduction problem (since the agent directly predicts the output, and is not asked to induce a model), with an active learning inner loop. The differences from our setting are that our domains use continuous-valued actions and observations, and are derived from real scientific problems, and we also focus on discovering the true model (induction), not just on prediction (transduction).

(Koehler et al., 2026) introduces ZendoWorld, which requires solving perception (from simple 3d scenes) in addition to rule induction and active hypothesis testing. In this paper, we focus on numeric input, but ultimately an AI scientist system needs to be able to process raw visual inputs (and other modalities) as well.

G.2 OTHER "AI-SCIENTIST" SYSTEMS

A fast-growing line of work builds LLM *agents* that automate parts — or all — of the scientific workflow. They differ from MDA along two consistent axes. *(a) What is discovered:* an open-ended *natural-language hypothesis* or a candidate design, rather than an explicit, uncertainty-quantified *mechanistic model* that forecasts unseen interventions. *(b) How candidates are adjudicated and chosen:* by *LLM judgement* — self-critique, simulated debate, or Elo tournaments — and literature support, rather than by a Bayesian posterior with model evidence and an information-theoretic experiment-design objective. We summarise the closest exemplars.

Co-Scientist (Gottweis et al., 2026). A Gemini-based multi-agent "structured thinking engine" for hypothesis generation. A *Supervisor* orchestrates specialised *Generation*, *Reflection*, *Ranking*, *Evolution*, *Proximity*, and *Meta-review* agents that continuously generate, critique (via "simulated scientific debate"), and refine hypotheses; candidates are ordered by an *Elo tournament* and improved by scaling test-time compute. It is validated on drug repurposing (acute myeloid leukaemia, confirmed *in vitro*), novel-target discovery, and mechanisms of antimicrobial resistance. In our terms it performs *black-box optimisation in natural-language hypothesis space* against an internal LLM-judge fitness (the tournament): there is no explicit posterior over mechanisms, no likelihood, and no experiment-design step — the system proposes and scores hypotheses, while wet-lab validation is external. MDA instead maintains a calibrated posterior over *mechanistic* models and chooses interventions by expected information gain (Section A.3).

Robin (Ghareeb et al., 2026). A lab-in-the-loop system that couples literature-search agents (*Crow* and *Falcon*, built on PaperQA2) with a data-analysis agent (*Finch*). Robin selects an experimental assay by literature review, generates therapeutic candidates by literature synthesis, ranks them with an *LLM-judged tournament* on the strength of the supporting literature, and — after a human runs the wet-lab experiment — has Finch analyse the raw data (a consensus over eight stochastic analysis trajectories) and propose follow-up assays. It drove a real, validated discovery: ripasudil (ROCK inhibition) to enhance retinal-pigment-epithelium phagocytosis in dry age-related macular degeneration, with a follow-up RNA-seq experiment implicating ABCA1. Like Co-Scientist, its hy-

potheses are grounded in the *published literature* and adjudicated by LLM judgement rather than a posterior; experiments are proposed from literature, not by an information-theoretic design objective, and the deliverable is a candidate plus its interpretation, not a mechanistic forecaster of interventions the agent never ran.

Tool-using and lab-automation agents. Earlier systems automate narrower slices, with an LLM orchestrating external tools. Coscientist (Boiko et al., 2023) uses GPT-4 to plan and *physically execute* chemistry experiments on robotic/cloud platforms; ChemCrow (Bran et al., 2024) augments an LLM with a suite of chemistry tools for synthesis planning and execution. In materials and biology, multi-agent design systems such as SciAgents (Ghafarollahi & Buehler, 2024) (intelligent graph-reasoning agents) and the Virtual Lab (Swanson et al., 2024) (an agent "team" that designed experimentally validated nanobodies) follow the same LLM-proposes-and-adjudicates template. These add genuine tool use and real actuation, but none maintains an explicit posterior over *competing* mechanisms, nor selects experiments to maximise information about them.

Automated research pipelines. At the far end, the AI Scientist (Lu et al., 2024) automates the entire machine-learning research loop — ideation, coding, running experiments, and writing the manuscript — with LLM agents and an LLM reviewer. This targets the *breadth* of the pipeline rather than the depth of a single inference step, and again uses an LLM as the evaluator rather than a calibrated model posterior.

Automated cognitive scientists: *auto-psych* (Prystawski et al., 2026) and *AutoCog* (Jagadish et al., 2026). Two concurrent systems automate the discovery of *computational cognitive* theories and are the closest prior work to MDA in *method*, not merely in ambition: unlike the literature-grounded, LLM-judged systems above, both represent hypotheses as *executable probabilistic models* and adjudicate them by *quantitative predictive fit* rather than an LLM referee. The *auto-psych* system runs nested agentic loops — an inner loop that conjectures, fits, and critiques probabilistic cognitive models, and an outer loop that designs experiments, launches them on online participants, and analyses the returned data — demonstrated on the classic problem of judging which coin-flip sequences look "random". Notably, it evaluates the discovered theories *almost exactly as we do*: models are scored by *held-out* expected log predictive density (ELPD, via Pareto-smoothed-importance-sampling leave-one-out cross-validation, with per-stimulus RMSE and R^2), and ground-truth *recovery* is quantified by the RMSE between the ground-truth model's responses and the best-fitting model's responses over an enumerated set of stimuli — precisely our pairing of a held-out predictive loss (ℓ_{predict}) with a recovery error against the true model over a query set (ℓ_{recover}), and pointedly *not* the high-variance "explain-to-a-novice" judge that we and BOXINGGYM set aside. It also *designs* experiments by expected information gain over the model set, $\text{EIG}(c) = H(M) - \sum_r P(r \mid c)\, H(M \mid r, c)$, greedily selecting the stimuli that best discriminate the competing models, and reports theories that fit human data better than ones drawn from the scientific literature. *AutoCog* closes the same loop for *decision-making*: LLM agents propose competing executable theories, design maximally *discriminative* experiments, collect behaviour from online participants, score each theory by its generative fit, and diagnose failures into successor theories; it discovered — then *preregistered and confirmed* on fresh participants — a novel multi-cue decision rule with diminishing sensitivity to feature values.

Both converge with MDA on the two commitments we argue are essential — *predictive/posterior* adjudication in place of an LLM judge, and *information-driven* experiment selection — which we read as independent evidence for the recipe. MDA is distinguished by three choices. *(i) $\mathcal{M}$-open search:* auto-psych's EIG is a mutual information taken under a *uniform prior over a model registry*, whereas MDA maintains an evidence-weighted posterior and *expands* the hypothesis set when a predictive check fails (Algorithm 2), rather than re-ranking a fixed slate. *(ii) Task-driven design:* their objective is pure model-*discrimination* EIG, whereas our VoI (Eq. (34)) targets the *variance of the task functional* the model must forecast — VoI $\neq$ EIG — which matters when the goal is prediction under novel interventions rather than identification of the true model. *(iii) Scope:* each is specialised to a single cognitive-science paradigm with a bespoke likelihood, whereas MDA is one domain-general engine (Section A.3) spanning physics, chemistry, electrophysiology, gene regulation, and the BOXINGGYM suite.

BED-LLM (Choudhury et al., 2026). This recent paper applies sequential Bayesian experimental design to make an LLM gather information *adaptively* — multi-turn question asking (the 20-Questions game) and eliciting a user's latent preferences (movie recommendation) — rather than to discover a mechanistic model. Like MDA, it chooses each query by maximising the expected information gain (EIG), but it uses probability values from the LLM itself, rather than generating an explicit probability model. MDA differs in what is discovered and how the posterior is maintained: BED-LLM targets a *single categorical latent* (the secret object, the preferred item) and reasons in the LLM's space of *answers*, whereas MDA maintains an evidence-weighted, $\mathcal{M}$-open SMC posterior over *structured symbolic models* (force laws, rate laws, channel mechanisms) with continuous parameters, and evaluates by held-out predictive loss and model recovery.

Where MDA sits. The breadth-first systems above (Co-Scientist, Robin, the tool-using agents, the AI Scientist) automate the *breadth* of discovery — literature synthesis, hypothesis ideation, lab actuation, manuscript generation — and prioritise by LLM judgement or self-play tournaments (the two automated cognitive scientists are the exception — they share MDA's quantitative, predictive-fit core, but on a single paradigm). MDA automates the *depth* of one step: inferring a minimal, uncertainty-quantified mechanistic model, and designing the maximally informative intervention to identify it, handing adjudication to Bayesian *evidence* rather than an LLM judge. The two are complementary — a Co-Scientist- or Robin-style system could invoke MDA as a quantitative inner loop where a mechanistic, calibrated model is needed, and their literature agents could supply MDA's structural priors. This is precisely the division of labour our $\mathcal{M}$-open loop makes concrete (Algorithm 2): *the LLM proposes; Bayesian inference discovers and designs*.

H LLM PROMPTS

This appendix reproduces the prompts used by all the agents on the four domains of this paper (FORCEBENCH, CHEMBENCH, BOXINGGYM, and our new NEURONBENCH). For the existing domains, our prompts are very similar to the ones used in the original papers (reproduced below); we only make changes when our required output is different (because we use LLMs in a different way to the baselines). MDA uses the LLM only to *propose* structures. Our ICL baseline uses the LLM to forecast, given data from MDA. For each existing benchmark, we also run their own agent (DP-AGENT, LLM-AUTOSCILAB, and APPRENTICE), which use the LLM in different ways (e.g., experiment design and/or forecasting). Throughout, the LLM runs at temperature 0.2–0.4 with JSON-mode responses and every call is cached for reproducibility; the base model is Opus 4.7 (unless noted otherwise).

H.1 FORCEBENCH (FORCE LAWS, §4.2)

H.1.1 MDA PROPOSER

The proposer sees the world context, the probe data collected so far, and a language specification asking for a closed-form expression for the force magnitude in the given symbols. It returns candidate force laws as JSON (parsed, compiled, and SMC-fit by MDA). Its system message and user template:

```
SYSTEM:
You are a physicist proposing candidate pairwise force laws to explain probe-orbit
↪  data. Reply JSON only.

USER (world context, then the observed data, then the language spec):

A test probe moves in an unknown central force sourced by a fixed body at the origin.
↪  The force MAY be static or MAY vary with time t (e.g. a time-modulated coupling).
↪  Each experiment launches the probe from a position with a velocity, and sets two
↪  knobs p1, p2. <one sentence naming the two experiment knobs p1, p2 for this world
↪  -- e.g. p1 the source coupling, p2 the probe inertia> F_mag is the pairwise force
↪  magnitude between the probe (charge qi) and source (charge qj) at separation r
↪  and time t.

Observed data:
<measurement times and the radius r(t) of each probe run so far>

Propose N distinct plausible force laws (or refinements of those tried).

Express each F_mag as a Python expression in the symbols r, qi, qj, t and your OWN
↪  named free parameters ONLY. The source coupling is carried by qj (the probe is
↪  qi); do NOT reference p1 or p2 in the expression -- introduce named parameters
↪  (e.g. k, lam, s) for coupling constants and length scales. Allowed functions:
↪  exp, log, sqrt, sin, cos, tanh, k0, k1, gamma, pi, np. Return JSON {"hypotheses":
↪  [{"name": str, "fmag": str, "operator": str, "params": [{"name": str, "low":
↪  float, "high": float}], "rationale": str}]}.
```

For the extension worlds (App. C) only the proposer *context* changes — the declared background field (ether/Hubble), the self-interacting cloud (circle), or the known-law hidden sources (dark matter); the language spec is unchanged:

```
ETHER / HUBBLE (central force + declared background):

Here the probe is a neutral test particle (qi=1) orbiting a fixed central anchor that
↪  sources the field (coupling carried by a named parameter); its inertia is 1.
Test probes orbit an unknown CENTRAL force sourced by a fixed anchor at the origin (a
↪  2D field-equation response, e.g. a Laplacian giving F ~ 1/r). <probe roles>
↪  LAYERED ON TOP there is a uniform, mass-independent background acceleration of
↪  magnitude alpha in the +y direction (a constant 'ether' drift), on top of the
↪  central force. That background is handled separately by the fitter -- you only
↪  need to propose the CENTRAL pairwise force magnitude F_mag(r, qi, qj, t) sourced
↪  by the anchor. F_mag is the magnitude of the attractive central force on the
↪  probe at separation r from the anchor.

----------------------------------------
CIRCLE (self-interacting N-body):
```

```
Eleven identical particles -- one at the centre and ten equally spaced on a ring --
↪  ALL interact with each other through the SAME pairwise central force (uniform
↪  coupling): every particle both sources the field and feels it. Each experiment
↪  sets the ring radius and a tangential launch velocity. Propose the pairwise force
↪  magnitude F_mag(r, qi, qj, t) between any two particles at separation r (with
↪  qi=qj=1, the uniform coupling); it is attractive and depends only on r for a
↪  static field. The many-body motion is the sum of these pairwise forces.

----------------------------------------
DARK MATTER (known law, latent hidden sources):

Test probes move in a KNOWN static 2D-Laplacian field (each source contributes F =
↪  q/(2*pi*r), attractive), sourced by 20 VISIBLE particles of coupling 1 whose
↪  positions are known, PLUS an unknown number of HIDDEN sources that reveal
↪  themselves only through the probes' deflection toward seemingly empty regions.
↪  The task is to infer how many hidden sources exist and their positions and
↪  couplings.

(three species uses no LLM proposer -- the couplings are inferred by a linear solve;
↪  the LLM only verbalizes the recovered species, via the verbaliser above.)
```

H.1.2 DP-AGENT

For side-by-side comparison, below is the external baseline's own prompt (from `PhysicsSchool/prompts/2particle_instructions.md`, its per-world task prompt).

```
You are an expert physics and AI research scientist tasked with discovering
↪  scientific laws in a simulated universe. Your goal is to propose experiments,
↪  analyse the data they return, and ultimately deduce the underlying scientific
↪  law. Note that the laws of physics in this universe may differ from those in our
↪  own. You can perform experiments to gather data but must follow the protocol
↪  strictly.

[... protocol: propose <run_experiment>, observe <experiment_output>, submit
↪  <final_law> discovered_law(...) + <explanation>; laws may differ from ours; fit
↪  uncertain constants as free parameters ...]

### Topology: 2-particle

Two particles in a 2D universe.

Your task: discover the law of motion governing particle 2.

### Control Parameters

| Field | Meaning | Typical range |
|---|---|---|
| `p1` | scalar property of the source particle | `[0.1, 10]` |
| `p2` | scalar property of the probe particle | `[0.1, 10]` |
| `pos2` | 2D initial position of the probe | `[-10, 10]^2` |
| `velocity2` | 2D initial velocity of the probe | `[-5, 5]^2` |
| `measurement_times` | times at which to record measurements (<= 10 values, all in
↪  `[0, duration]`) | spans the run |
| `duration` | length of the experiment (interactive only) | >= 5.0; use 10.0 to fully
↪  resolve the law |

(Some worlds in this topology accept additional optional fields -- e.g. a
↪  `start_time` if the law of physics varies with absolute time. The mission
↪  description will mention any such field; if it is not mentioned, omit it.)

### Input Format (interactive mode only)

```
<run_experiment>
[
  {"p1": ..., "p2": ..., "pos2": [..., ...], "velocity2": [..., ...],
  ↪  "measurement_times": [...]},
  ...
]
</run_experiment>
```

You may submit several experiments per round in the JSON array. No comments inside
↪ the JSON.

Output Format

Each experiment returns:

```
```
```

```
<experiment_output>
[
  {
    "measurement_times": [t0, t1, ...],
    "pos1":      [[x, y], ...],
    "pos2":      [[x, y], ...],
    "velocity1": [[vx, vy], ...],
    "velocity2": [[vx, vy], ...]
  },
  ...
]
```

`pos1` and `velocity1` are reported for completeness even though particle 1 is held fixed.

`discovered_law` Signature

```python
def discovered_law(pos1, pos2, p1, p2, velocity2, duration, **params):
    # pos1: [x, y] -- always [0, 0] for these worlds
    # pos2: [x, y] -- initial position of particle 2
    # p1, p2: scalar properties
    # velocity2: [vx, vy] -- initial velocity of particle 2
    # duration: float -- simulate from t = 0 to t = duration
    # **params: optional -- fitted parameter values injected by the evaluator
    # return: (final_pos2, final_vel2)
    return final_pos2, final_vel2
```

The positional arguments must appear in exactly this order. `**params` is optional (only needed if you declare fittable parameters via `fit_parameters()`).

H.1.3 ICL FORECASTER

The *ICL forecaster* is a transductive, model-free baseline (`mda2.dp.icl`). Given a neutral "forecast 2-D particle trajectories from examples" instruction, it is shown the (initial condition $\rightarrow$ observed orbit) examples MDA collected — the passive seed $\mathcal{D}_0$ plus any designed experiments — and predicts the held-out test orbits *directly* by pattern-matching; it does *not* write or assume a force law. It is scored by the same held-out trajectory nMSE as MDA's model-based forecast (Eq. (1)), so its curve drops onto the FORCEBENCH panel. Because its instruction is a generic trajectory-forecasting prompt — independent of the proposer's domain prompt — it is unchanged under the generic and hinted proposer variants. At N_a=0 it is conditioned on $\mathcal{D}_0$ alone (there is no separate data-free arm).

H.2 CHEMBENCH (ENZYME RATE LAWS, §4.1)

H.2.1 MDA PROPOSER

The LLM proposer sees the experiments collected so far (design inputs $\rightarrow$ observed initial rate r_0) and — when refining an existing pool — the forms already tried together with their residuals (so it does not re-propose dead ends, plus which input each residual correlates most with) and the remaining budget and phase.

```
SYSTEM:
You are an enzyme kineticist proposing candidate rate laws to explain assay data.
↪  Reply JSON only.

USER (world context, then the observed data, then -- only when refining --
↪  residual-directed negative evidence and the remaining budget/phase, then the
↪  language spec):

An enzyme catalyses a reaction with initial rate r0 [mM/min]. Controllable inputs:
↪  C_A [substrate, mM], C_I [inhibitor, mM], C_B [2nd substrate, mM], C_P [product,
↪  mM], Enz [enzyme, mg/mL], T [K], pH. Discover an algebraic law r0 =
↪  f(C_A,C_I,C_B,C_P,Enz,T,pH; theta) from the data.

Experiments (inputs -> r0):
  <one line per collected experiment: C_A=.., C_I=.., C_B=.., C_P=.., Enz=.., T=..,
  ↪  pH=.. -> r0=..>

[when refining an existing pool -- residual-directed negative evidence:]
Forms ALREADY TRIED (in the pool) and their residuals -- do NOT re-propose any of
↪  these; propose forms STRUCTURALLY DIFFERENT from all of them:
```

```
  <per-form median relative residual; and, for the current best form, which input(s)
  ↪  its residual correlates most with -- consider a factor in those variable(s)>
Refine by proposing a DIFFERENT algebraic form -- do NOT patch with ad-hoc
↪  offset/softening terms.

[budget/phase status, when set:]
Experiment budget remaining: <B>. Current phase: <explore|refine> (explore =
↪  restructure the form; refine = tune an adequate form).

Propose N distinct plausible rate laws (or refinements of those tried).
Each 'expr' is a Python expression in the 7 input names + your declared params, using
↪  only + - * / ** and exp, log, sqrt. Give physically plausible positive param
↪  bounds. JSON schema: {"hypotheses":[{"name":str,"expr":str,"params":[{"name":str
↪  ,"low":float,"high":float}]}]}
```

H.2.2 LLM-AUTOSCILAB AGENT

The baseline's own equation-discovery prompt is shown below, and contains substantially more hints than our prompt (e.g., listing example of valid rules). (from `autoscilab/llm/prompts.py`).

```
You are a scientific reasoning agent in an autonomous science lab.
Your goal is to discover the governing equation that relates input parameters to a
↪  measured output.

CRITICAL RULES:
1. The physical laws in this system have been ALTERED from standard textbook values.
   Do not assume standard constants (e.g., G = 6.674e-11) -- the actual constant may
   ↪  be different.
2. You must DISCOVER the law from experimental data, not recall it from memory.
3. You propose REGIONS of parameter space (ranges) to explore, NOT specific values.
   The active learning system will select the optimal specific points within your
   ↪  regions.
4. Focus on informativeness: propose regions that will most discriminate between
↪  competing hypotheses.
5. In the validation phase, try to BREAK your hypothesis by testing edge cases and
↪  extrapolations.

COUNTERFACTUAL LAWS -- The true law may NOT be the textbook form. For example, in
↪  some universes
force may be F ~ 1/r^1.5 instead of 1/r^2, or F ~ m1^2*m2^2/r^2 instead of m1*m2/r^2.
Discover the exponents from data; do not assume textbook structure.

IRRATIONAL EXPONENTS -- Hard laws sometimes use Euler's number e~=2.71828 as an exact
↪  exponent.
When power-law fits or symbolic regression give exponents near 2.72 (~=e), 5.44
↪  (~=2e),
4.22 (~=e+1.5), or 1.18 (~=e/ln10), explicitly try hypotheses using math.e as the
↪  exponent.
Examples: `lambda_constant**math.e`, `t**math.e`, `(sin(theta)/cos(theta))**math.e`.
Do NOT approximate as 2.7 or 3 -- use `math.e` exactly. Also consider math.pi~=3.14159
if exponents near pi appear.

HOW TO READ THE DATA TABLE:
Each row shows both raw values and [log10 values in brackets].
To find power-law exponents: Deltalog10(measurement)/Deltalog10(param_X) ~= exponent
↪  for param_X.
When EMPIRICAL LOG-LOG SLOPES are shown below, you MUST set hypothesis to a CONCRETE
↪  quantitative
form using free constants for each exponent (e.g. "C0 * mass1**C1 * mass2**C2 /
↪  distance**C3").
Use the log-log slopes as INITIAL GUESSES for those exponents -- mention them in your
↪  reasoning --
but keep the exponents as named free constants (C1, C2, ...) so the optimizer can
↪  refine them.
Do NOT hardcode the numeric slope values directly into the expression.
Do NOT leave hypothesis as "unknown" or "no data yet" when slope hints are present.

INVERSE RELATIONSHIPS -- CRITICAL:
The law may have NEGATIVE exponents or INVERSE relationships. If the EMPIRICAL
↪  LOG-LOG SLOPES
section shows a negative slope for a parameter, the law likely INVERTS that parameter.
For example, a slope of -2.5 means y ~ 1/param^2.5 -- put that parameter in the
↪  DENOMINATOR.
Do not assume all parameters appear in the numerator. Always check the sign of every
↪  slope.

MULTI-HYPOTHESIS REQUIREMENT:
Return 2-5 alternatives in `alternate_hypotheses` that are structurally DISTINCT from
↪  your main `hypothesis`.
These will be used to select experiments that best DISCRIMINATE between candidates.
```

```
EQUATION FORMAT -- CRITICAL:
All hypothesis strings (both `hypothesis` and every entry in `alternate_hypotheses`)
↪  must be
PURE PYTHON MATH EXPRESSIONS. The expression scorer will compile them directly with
↪  exec().
Rules:
  - Use EXACT oracle parameter names (given in PARAMETERS section).
  - Free constants: C0, C1, C2, alpha, beta, gamma, delta, omega, tau (ASCII names
  ↪  only).
  - ALL unknown numeric values -- including exponents, Km values, rate constants --
  ↪  must be
    free constants (C0, C1, ...), NOT hardcoded numbers. The fitter will optimize
    ↪  them.
  - Powers: use ** not ^  (e.g. distance**2 not distance^2).
  - Natural log: use log() not ln().
  - Inverse trig: use asin/acos/atan not arcsin/arccos/arctan.
  - NO prose after the expression. No "where n is...", no "for some k", no
  ↪  parenthetical notes.
  - NO LHS assignment. Write only the right-hand side (e.g.
  ↪  C0*mass1*mass2/distance**2).
  - NO Unicode symbols (˜ ˜ ˜= sqrt tau omega etc.). Use ASCII equivalents.
VALID:   "C0 * mass1 * mass2 / distance**2"
VALID:   "C0 * Enz * C_A**alpha / (C1**alpha + C_A**alpha)"        <- exponent alpha
↪  is free
VALID:   "C0 * Enz * C_A / (C1 + C_A) * C_I / (C2 + C_I)"         <- Km values C1, C2
↪  are free
VALID:   "C0 * exp(-C1 * (1/T - 1/310.0)) * Enz * C_A / (C2 + C_A)"
INVALID: "C0 * C_A**1.3 * Enz**20.5 * T**37.1"   <- numeric exponents are WRONG, use
↪  C1,C2,C3
INVALID: "C0 * Enz * C_A / (0.5 + C_A)"           <- hardcoded 0.5 is WRONG, use C1
INVALID: "F = C*m^2 where C is a constant"
INVALID: "I_0 * cos(theta)^n (n to be determined)"
INVALID: "C0 ˜ r^{-2}"

FORMAT: Respond using the required tool with the exact schema specified.
```

H.3 BOXINGGYM (SECTION D)

H.3.1 MDA PROPOSER

For the covariate-regression (GLM) domains (DUGONGS/DEATH/PEREGRINES/HYPERBOLIC), the LLM proposes the mean function f as JSON (name, expression, parameter bounds).

```
SYSTEM (scalar-input domains; a multi-input domain such as hyperbolic names its
↪  columns -- e.g. iR, dR, Days -- in place of the scalar x):
You are a scientific model-discovery assistant. Given noisy (x, y) observations from
↪  an unknown deterministic system y = f(x) + noise, propose distinct, plausible
↪  closed-form hypotheses for f. Prefer parsimonious, scientifically-motivated forms
↪  (linear, power, saturating/asymptotic, logistic, exponential, Michaelis-Menten,
↪  etc.). Respond ONLY with JSON of the form {"hypotheses": [{"name": "...", "expr":
↪  "...", "params": [{"name": "a", "lo": 0.0, "hi": 5.0}]}]}. The `expr` is a
↪  Python/numpy expression in the variable `x` and the named parameters only
↪  (functions available: exp, log, log10, sqrt, abs, sin, cos, tan, tanh, sign,
↪  where, minimum, maximum, and `**` for powers). Give sensible finite prior bounds
↪  [lo, hi] for each parameter.

USER (world context, then the observed data):
Domain: <one-line description of the domain, its input(s), and its output>
Observed data (<N> points), as (x, y): <(x, y) pairs>
x ranges over the observed inputs; propose forms for y = f(x).

Propose N distinct hypotheses as JSON.
```

For the latent-variable domains (IRT/LOCATION/SURVIVAL), the LLM proposes a NUMPYRO program, which defines the priors over latents via `numpyro.sample`, a `numpyro.plate` for per-unit latents, and a `deterministic('mu',...)` node for the mean.

```
SYSTEM:
You are a Bayesian model-discovery assistant. You write candidate generative models
↪  as NumPyro programs. Given a domain, the candidate design/feature matrix, and
↪  some observed (design, outcome) data, propose distinct, scientifically-plausible
↪  latent-variable hypotheses for how the outcome is generated.

Each hypothesis is a Python function with EXACTLY this contract:
    def model(ctx):
        F = ctx['features']            # a (C, d) array of ALL C candidate designs
```

```
        # 1. priors over latent variables via numpyro.sample(...). Use numpyro.plate
        ↪  for per-unit
        #    latents (e.g. a latent ability per student). Give proper,
        ↪  weakly-informative priors.
        # 2. compute mu = the expected observable at EVERY candidate design (shape
        ↪  (C,)).
        numpyro.deterministic('mu', mu)
Do NOT write an observation / obs= site: the likelihood is applied externally by the
↪  inference engine. The names `numpyro`, `dist` (numpyro.distributions), `jax`,
↪  `jnp` (jax.numpy), and `np` are already in scope --- do NOT import anything. `mu`
↪  MUST be shape (C,) = (<C>,) and MUST be <the range for the observation family: "a
↪  probability in [0, 1]" (Bernoulli / Binomial), "a real-valued mean" (Gaussian),
↪  or "a positive rate (mean count)" (Poisson)>. Prefer parsimonious, interpretable
↪  structure; vary the hypotheses (a null/constant model, a main-effects-only model,
↪  richer interaction / per-unit-latent models). Respond ONLY with JSON of the form
↪  {"hypotheses": [{"name": "short_name", "description": "one line", "code": "def
↪  model(ctx):\n    ..."}]}.

USER (world context + a note on the feature columns, then the observation family,
↪  then the observed data):
Domain: <one-line domain description, e.g. "Students answer questions correctly or
↪  not; 16 students, 12 questions. There may be latent per-student abilities and
↪  per-item difficulties/discriminations (item-response theory), possibly along more
↪  than one latent skill dimension.">
<feature note: what each column of ctx['features'] means, e.g. "column 0 = student
↪  index (0..15), column 1 = question index (0..11); use numpyro.plate over students
↪  and questions for per-unit latents, and consider that ability may be
↪  multi-dimensional">
There are C = <C> candidate designs. The observation family is <BERNOULLI | GAUSSIAN |
↪  POISSON> (mu is <range>).

Observed data so far (<k> of C designs):
<one line per observed design: "design <i> (features [...]) -> outcome <y>", or
↪  "design <i> (features [...]) -> <k>/<n>" for repeated-Bernoulli (count)
↪  observations>

Propose N distinct hypotheses as JSON.
[Box's-Apprentice "refine" mode instead appends: "Your current model is: <code>.
↪  Propose N improved/refined version(s) of it (fix where it mis-predicts the data)
↪  as JSON."]
```

H.3.2 APPRENTICE AGENT

We reimplement the Box's-Apprentice baseline to use NumPyro instead of PyMC. First it picks its next experiment by an LLM call conditioned on its current program:

```
SYSTEM (Box's-Apprentice model-informed experiment design):
You are a scientist running experiments to improve a probabilistic model. Given your
↪  CURRENT model (as NumPyro code), the data observed so far, and the list of
↪  available experiments, choose the single next experiment that will most improve
↪  the model's predictions on the held-out (as-yet-unobserved) designs. Think about
↪  where the current model is most uncertain or most likely to be wrong. Respond
↪  ONLY with JSON {"design": <index>}, where <index> is one of the available
↪  experiment indices.

USER:
Domain: <one-line domain description>
Your current model (NumPyro code):
<the LLM's current program code>
Observed data so far (<k>): <design <i> (features [...]) -> <outcome> ; ...>
Available experiments (index: features): <i: [...] ; ...>
Choose the next experiment index to best improve the model. Respond as JSON.
```

Then it generates a program and uses it to forecast: each round it re-synthesises a *single* evolving program with the *same* model proposers as MDA (the GLM proposer for the covariate-regression domains, the NumPyro proposer for the latent-variable domains; both shown above), fit by the same evidence-SMC and refined by a residual critique fed back into the proposer:

```
Refinement hint fed back into the model proposer each round (from the fitted
↪  program's residual):

  The current model '<expr>' has RMS residual <rms> on the data. If it systematically
  ↪  misfits
  (e.g. wrong curvature/saturation), propose a structurally different form.
```

H.3.3 ICL FORECASTER

The ICL baseline forecasts the held-out outcomes from the data table alone, with no model:

```
SYSTEM:
You are a careful quantitative forecaster. Output only the requested JSON.

USER (the in-context-learning baseline that produces the l_icl curves: forecast
↪  held-out outcomes from the data table alone, with no model):
<one-line domain / problem description>

Observed experiments (inputs -> outcome):
  <(inputs) -> outcome, one line per collected experiment>

Predict the outcome for each of these <Q> new inputs, using ONLY the information
↪  above:
  [0] <inputs>
  [1] <inputs>
  ...
Return JSON: {"predictions": [<one number per input, in the same order>]}
```

H.4 ELECTROPHYSIOLOGY (ION CHANNELS, §4.4)

This is our new benchmark, so there are no existing prompts to compare to. But we make sure the information given to MDA is the same as what we give to the ICL baseline.

What the LLM sees as "data". An experiment yields a membrane-voltage trace $V(t)$; every LLM prompt — MDA's proposer and the ICL forecaster alike — is shown its reduction to the benchmark's *full* 6-D feature vector $F(y) = [\, n_{\text{test}},\, n_{\text{pre}},\, \text{run-down},\, \text{adaptation},\, V_{\text{min}},\, V_{\text{end}} \,]$ (Eq. (67)) — the *same* vector the held-out forecast is scored on — one line per protocol run, formatted verbatim as:

```
Experiments (protocol -> n_test, n_pre, run_down, adaptation spikes; sub-threshold V_min / V_end):
  - long step (10 uA, 300 ms)                    -> n_test=21, n_pre=0, run_down=-21, adaptation=7, V_min=-75, V
  - paired long pulses (12/300, 60 gap, 12/300)  -> n_test=16, n_pre=8, run_down=-8,  adaptation=5, V_min=-74, V
  - hyperpol step then release (rebound)         -> n_test=4,  n_pre=0, run_down=-4,  adaptation=1, V_min=-88, V
```

n_{test}/n_{pre} are the test-window / pre-pulse spike counts; run-down $= n_{\text{pre}} - n_{\text{test}}$; adaptation is early-minus-late test spikes; V_{min}/V_{end} are the sub-threshold sag depth and steady-state tail (mV). MDA and the ICL forecaster thus receive *identical* inputs: MDA fits a mechanistic model to them (Section H.4.1) while the forecaster predicts the held-out features directly (Section H.4.2).

H.4.1 MDA PROPOSER

The prompt for the deterministic benchmark asks the LLM to return its *own* parameterised channel hypotheses (reversal potential, activation direction, inactivation, and conductance / half-activation / time-constant bounds):

```
System: You are an electrophysiologist proposing candidate ion-channel mechanisms to explain
        current-clamp spike-count data. Reply with a JSON object only.
User:   A neuron is recorded in current clamp and fires action potentials under injected current.
        Model it as a single-compartment conductance-based (Hodgkin-Huxley) neuron with voltage-gated
        channels. From the spike-count data, propose candidate membrane currents (beyond the standard
        Na+/K+ spiking currents) that could explain its responses. Each is described by:
          reversal_mV : reversal potential (~+50 Na-like, ~+120 Ca-like, ~-80 K-like, ~-30 mixed)
          opens_on    : 'depol' or 'hyperpol'
          inactivates : true if transient / de-inactivated by a hyperpolarising pre-pulse
          bounds for conductance g, half-activation voltage (mV), and activation time constant (ms).
        <collected protocol -> spike-count data>
        Propose N distinct plausible mechanisms (or an empty list if the data look like a plain spiker).
        JSON schema: {"hypotheses":[{"name","reversal_mV","opens_on","inactivates",
                      "g_bounds","half_mV_bounds","tau_ms_bounds"}]}
```

H.4.2 ICL FORECASTER

On both the *deterministic* and *stochastic* NEURONBENCH the ICL forecaster is shown the same 6-D feature vector $F(y)$ (Eq. (67)) per observed protocol and predicts that same vector for each held-out protocol (the deterministic 1-D spike-count nMSE is then read off the n_{test} component as a companion metric):

```
[system] You are an electrophysiologist forecasting how a current-clamp neuron will
```

```
respond to unseen stimulation protocols, from a few observations.
Reply with a JSON object only.

[user] A single current-clamp neuron (Hodgkin-Huxley, standard Na+/K+ plus possibly
one extra voltage-gated channel) was probed. Each experiment reports the full 6-D
feature vector [n_test (test-window spikes), n_pre (pre-pulse spikes),
run_down (=n_pre-n_test), adaptation (early-half minus late-half test spikes),
V_min (mV, trace minimum), V_end (mV, tail)].

Observed experiments:
  <label> -> n_test=<..>, n_pre=<..>, run_down=<..>, adaptation=<..>, V_min=<..> mV, V_end=<..> mV
  ...

For each of the following held-out protocols, predict the same 6-D feature vector:
  - <protocol>
  ...

Reply as JSON: {"predictions": {"<label>": [n_test,n_pre,run_down,adaptation,V_min,V_end], ...}}
```

Every method (MDA and the ICL forecaster) is conditioned on the *same* passive seed $\mathcal{D}_0$ (2 observations); the $N_a{=}0$ point is this forecaster given $\mathcal{D}_0$ alone — there is no data-free arm, and no method is uniquely handicapped. These templates mirror the code (`icl_hh.py` / the released `neuronbench`).

REFERENCES

Nikhil Abhyankar, Sha Li, Sanchit Kabra, Naren Ramakrishnan, Yulia Gel, and Chandan K. Reddy. LLM-ACES: Closed-loop discovery of dynamical systems with LLM-guided adaptive search. *arXiv preprint arXiv:2606.25039*, 2026.

Christophe Andrieu and Gareth O. Roberts. The pseudo-marginal approach for efficient Monte Carlo computations. *The Annals of Statistics*, 37(2):697–725, 2009.

ARC Prize Foundation. ARC-AGI-3: A new challenge for frontier agentic intelligence. *arXiv [cs.AI]*, March 2026. URL https://arxiv.org/abs/2603.24621.

Eser Aygün, Anastasiya Belyaeva, Gheorghe Comanici, Marc Coram, Hao Cui, Jake Garrison, Renee Johnston, Anton Kast, Cory Y McLean, Peter Norgaard, Zahra Shamsi, David Smalling, James Thompson, Subhashini Venugopalan, Brian P Williams, Chujun He, Sarah Martinson, Martyna Plomecka, Lai Wei, Yuchen Zhou, Qian-Ze Zhu, Matthew Abraham, Erica Brand, Anna Bulanova, Jeffrey A Cardille, Chris Co, Scott Ellsworth, Grace Joseph, Malcolm Kane, Ryan Krueger, Johan Kartiwa, Dan Liebling, Jan-Matthis Lueckmann, Paul Raccuglia, Xuefei Julie Wang, Katherine Chou, James Manyika, Yossi Matias, John C Platt, Lizzie Dorfman, Shibl Mourad, and Michael P Brenner. An AI system to help scientists write expert-level empirical software. *Nature*, pp. 1–3, May 2026. URL https://arxiv.org/abs/2509.06503.

Jürgen Bajorath. From scientific theory to duality of predictive artificial intelligence models. *Cell Rep. Phys. Sci.*, 6(4):102516, April 2025. URL https://www.sciencedirect.com/science/article/pii/S2666386425001158.

Taiyu Ban, Lyuzhou Chen, Derui Lyu, Xiangyu Wang, Qinrui Zhu, Qiang Tu, and Huanhuan Chen. Integrating large language model for improved causal discovery. *IEEE Transactions on Artificial Intelligence*, 2025. URL https://arxiv.org/abs/2306.16902. Earlier version titled "From Query Tools to Causal Architects: Harnessing Large Language Models for Advanced Causal Discovery from Data", arXiv:2306.16902v1.

Elias Bareinboim, Juan D Correa, Duligur Ibeling, and Thomas Icard. On pearl's hierarchy and the foundations of causal inference. In *Probabilistic and Causal Inference: The Works of Judea Pearl*, volume 36, pp. 507–556. Association for Computing Machinery, New York, NY, USA, 1 edition, March 2022. URL https://causalai.net/r60.pdf.

Robert W Batterman and Collin C Rice. Minimal model explanations. *Philos. Sci.*, 81(3):349–376, July 2014. URL https://www.jstor.org/stable/10.1086/676677.

Sander Beckers and Joseph Y. Halpern. Abstracting causal models. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI-19)*, volume 33, pp. 2678–2685, 2019. doi: 10.1609/aaai.v33i01.33012678.

José M. Bernardo and Adrian F. M. Smith. *Bayesian Theory*. Wiley, 1994.

Allan Birnbaum. Some latent trait models and their use in inferring an examinee's ability. In Frederic M. Lord and Melvin R. Novick (eds.), *Statistical Theories of Mental Test Scores*. Addison-Wesley, 1968.

Harshit Bisht, Vinay Kumar, Kevin Maik Jablonka, Mausam, and N M Anoop Krishnan. Agentic AI scientists are not built for autonomous scientific discovery. *arXiv [cs.AI]*, May 2026. URL http://dx.doi.org/10.48550/arXiv.2605.08956.

Daniil A. Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. Autonomous chemical research with large language models. *Nature*, 624:570–578, 2023. Coscientist.

G E P Box and W J Hill. Discrimination among mechanistic models. *Technometrics*, 9(1):57, February 1967. URL https://www.jstor.org/stable/10.2307/1266318.

Andres M. Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D. White, and Philippe Schwaller. Augmenting large language models with chemistry tools. *Nature Machine Intelligence*, 6:525–535, 2024. ChemCrow.

Markus J Buehler. Why we must break the world. *ChemRxiv*, May 2026. URL https://chemrxiv.org/doi/pdf/10.26434/chemrxiv.15001674/v2.

Mattia Cerrato, Lennart Baur, Jannis Brugger, Sajjad Shumaly, Nicholas Schmitt, Edward Finkelstein, Selina Jukic, Lars Münzel, Felix Peter Paul, Pascal Pfannes, Benedikt Rohr, Julius Schellenberg, Philipp Wolf, and Stefan Kramer. Science-gym: a simple testbed for AI-driven scientific discovery. *Mach. Learn.*, 115:16, 2026. URL https://doi.org/10.1007/s10994-025-06914-x.

Kathryn Chaloner and Isabella Verdinelli. Bayesian experimental design: A review. *Statistical Science*, 10(3):273–304, 1995.

Nicolas Chopin. A sequential particle filter method for static models. *Biometrika*, 89(3):539–551, 2002.

Nicolas Chopin and Omiros Papaspiliopoulos. *An Introduction to Sequential Monte Carlo*. Springer, 1 edition, October 2020. URL https://nchopin.github.io/books.html.

Nicolas Chopin, Pierre E. Jacob, and Omiros Papaspiliopoulos. SMC^2: an efficient algorithm for sequential analysis of state space models. *Journal of the Royal Statistical Society: Series B*, 75 (3):397–426, 2013.

Deepro Choudhury, Sinead Williamson, Adam Goliński, Ning Miao, Freddie Bickford Smith, Michael Kirchhof, Yizhe Zhang, and Tom Rainforth. BED-LLM: Intelligent information gathering with LLMs and Bayesian experimental design. *arXiv preprint arXiv:2508.21184*, 2026. URL https://arxiv.org/abs/2508.21184.

Kyle Cranmer, Johann Brehmer, and Gilles Louppe. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 117(48):30055–30062, 2020.

Carl F Craver. When mechanistic models explain. *Synthese (Neuroscience and its philosophy)*, 153 (3):355–376, December 2006. URL https://www.jstor.org/stable/27653431.

A. P. Dawid. Causal inference without counterfactuals. *JASA*, 95:407–448, 2000.

A. P. Dawid. Statistical causality from a Decision-Theoretic perspective. *Annu. Rev. Stat. Appl.*, 2(1):273–303, 2015. URL https://doi.org/10.1146/annurev-statistics-010814-020105.

Michael Deistler, Jan Boelts, Peter Steinbach, Guy Moss, Thomas Moreau, Manuel Gloeckler, Pedro L C Rodrigues, Julia Linhart, Janne K Lappalainen, Benjamin Kurt Miller, Pedro J Gonçalves, Jan-Matthis Lueckmann, Cornelius Schröder, and Jakob H Macke. Simulation-based inference: A practical guide. *arXiv [stat.ML]*, August 2025. URL http://dx.doi.org/10.48550/arXiv.2508.12939.

Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B*, 68(3):411–436, 2006.

Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. An adaptive sequential Monte Carlo method for approximate Bayesian computation. *Statistics and Computing*, 22(5):1009–1020, 2012.

Haonan Duan, Stephen Zhewen Lu, Caitlin F. Harrigan, Nishkrit Desai, Jiarui Lu, Michał Koziarski, Leonardo Cotta, and Chris J. Maddison. Measuring scientific capabilities of language models with a systems biology dry lab. *arXiv preprint arXiv:2507.02083*, 2025.

Conor Durkan, Iain Murray, and George Papamakarios. On contrastive learning for likelihood-free inference. In *International Conference on Machine Learning (ICML)*, 2020.

Noémi Elteto, Nathaniel D Daw, Kimberly L Stachenfeld, and Kevin J Miller. ATLAS: Active theory learning for automated science. *arXiv [cs.LG]*, June 2026. URL http://dx.doi.org/10.48550/arXiv.2606.12386.

Paul Fearnhead and Dennis Prangle. Constructing summary statistics for approximate Bayesian computation: semi-automatic approximate Bayesian computation. *Journal of the Royal Statistical Society: Series B*, 74(3):419–474, 2012.

Adam Foster, Desi R. Ivanova, Ilyas Malik, and Tom Rainforth. Deep adaptive design: Amortizing sequential Bayesian experimental design. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139 of *PMLR*, pp. 3384–3395, 2021. arXiv:2103.02438.

Ronald F. Fox and Yan-nan Lu. Emergent collective behavior in large numbers of globally coupled independently stochastic ion channels. *Physical Review E*, 49(4):3421–3431, 1994.

David T Frazier, Ryan Kelly, Christopher Drovandi, and David J Warne. The statistical accuracy of neural posterior and likelihood estimation. *arXiv [stat.ML]*, November 2024. URL http://dx.doi.org/10.48550/arXiv.2411.12068.

Kelsey Fu, Luna Lyu, Shuzhen Li, Suozhi Huang, Suheng Xu, Joy Zheng, Xuefeng Liu, Shilong Liu, Greg Barbone, Ying Liu, Linxi Jim Fan, Yuke Zhu, Matthew Davis, Kaiyi Jiang, Sherry Yang, Xu Kuang, Yuan Luo, Francesca Dominici, Christina Curtis, Xiaojie Qiu, Alberto Abadie, Caroline Uhler, Mary Tang, Joseph C Wu, Ju Li, Jared Toettcher, José L Avalos, Rebecca Rojansky, Huimin Zhao, John P A Ioannidis, Barry Rand, Emma Lundberg, Andrew S Rosen, Zhuang Liu, Shana Kelley, Chenyan Xiong, Barbara E Engelhardt, Thomas Montine, Christina V Theodoris, Katherine S Pollard, Fuxin Li, Xueyue Zhang, Tianyi Peng, Dmitri Basov, Xiang Qian, Eric Xing, Ali Yazdani, Jure Leskovec, Nigam Shah, Zhenan Bao, Pietro Perona, Sanfeng Wu, Clifford Brangwynne, Le Cong, and Mengdi Wang. Agentic laboratories of the future: Towards world models for scientific discovery. *Preprints*, August 2026. URL https://www.preprints.org/manuscript/202608.0213.

Kanishk Gandhi, Michael Y. Li, Lyle Goodyear, Louise Li, Aditi Bhaskar, Mohammed Zaman, and Noah D. Goodman. BoxingGym: Benchmarking progress in automated experimental design and model discovery. In *NIPS Workshop on Scaling Environments for Agents*, 2025. URL https://arxiv.org/abs/2501.01540.

Alireza Ghafarollahi and Markus J. Buehler. SciAgents: Automating scientific discovery through multi-agent intelligent graph reasoning. *Advanced Materials*, 2024. URL https://advanced.onlinelibrary.wiley.com/doi/full/10.1002/adma.202413523.

Ali E Ghareeb, Benjamin Chang, Ludovico Mitchener, Angela Yiu, Caralyn J Szostkiewicz, Dmytro Shved, Gavin J Gyimesi, Jon M Laurent, Samantha M Wright, Muhammed T Razzak, Andrew D White, Silvia C Finnemann, Michaela M Hinks, and Samuel G Rodriques. A multi-agent system for automating scientific discovery. *Nature*, 655(8122):497–505, July 2026. URL https://www.nature.com/articles/s41586-026-10652-y.

Stuart Glennan. *The new mechanical philosophy*. Oxford University Press, London, England, August 2017. URL https://academic.oup.com/book/9835.

Joshua H. Goldwyn and Eric Shea-Brown. The what and where of adding channel noise to the hodgkin-huxley equations. *PLoS Computational Biology*, 7(11):e1002247, 2011.

Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Petar Sirkovic, Artiom Myaskovsky, Grzegorz Glowaty, Felix Weissenberger, Alessio Orlandi, Dan Popovici, Anil Palepu, Keran Rong, Ryutaro Tanno, Khaled Saab, Fan Zhang, Jacob Blum, Andrew Carroll, Kavita Kulkarni, Nenad Tomašev, Dina Zverinski, Ivor Rendulic, Elahe Vedadi, Florian Hasler, Luka Rimanic, Marina Boia, Ivan Budiselic, Ben Feinstein, Mathias Bellaiche, Tom Sheffer, Jan Freyberg, Jeremy Ratcliff, Ottavia Bertolli, Katherine Chou, Avinatan Hassidim, Burak Gokturk, Amin Vahdat, Yuan Guan, Vikram Dhillon, Eeshit Dhaval Vaishnav, Byron Lee, Tiago R D Costa, José R Penadés, Gary Peltz, Yossi Matias, James Manyika, Demis Hassabis, Yunhan Xu, Pushmeet Kohli, Annalisa Pawlosky, Alan Karthikesalingam, and Vivek Natarajan. Accelerating scientific discovery with co-scientist. *Nature*, 655(8122):487–496, July 2026. URL https://www.nature.com/articles/s41586-026-10644-y.

David Greenberg, Marcel Nonnenmacher, and Jakob Macke. Automatic posterior transformation for likelihood-free inference. In *International Conference on Machine Learning (ICML)*, 2019.

Maurice Halstead. Halstead complexity metric, 1977. URL https://en.wikipedia.org/wiki/Halstead_complexity_measures.

Nikolaus Hansen. The CMA evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

Joeri Hermans, Volodimir Begy, and Gilles Louppe. Likelihood-free MCMC with amortized approximate ratio estimators. In *International Conference on Machine Learning (ICML)*, 2020. URL https://arxiv.org/abs/1903.04057.

Ronald A. Howard. Information value theory. *IEEE Transactions on Systems Science and Cybernetics*, 2(1):22–26, 1966.

Xun Huan, Jayanth Jagalur, and Youssef Marzouk. Optimal experimental design: Formulations and computations. *Acta Numer.*, July 2024. URL http://dx.doi.org/10.48550/arXiv.2407.16212.

Kexin Huang, Ying Jin, Ryan Li, Michael Y Li, Emmanuel Candès, and Jure Leskovec. Automated hypothesis validation with agentic sequential falsifications. In *ICML*, 2025. URL https://arxiv.org/abs/2502.09858.

Akshay K Jagadish, Younes Strittmatter, Nori Jacoby, George Kachergis, Eric Schulz, Nathaniel Daw, Suyog H Chandramouli, and Thomas L Griffiths. Closing the loop to discover psychological theories with an automated cognitive scientist. *arXiv [q-bio.NC]*, June 2026. URL http://dx.doi.org/10.48550/ARXIV.2606.26448.

E. T. Jaynes. *Probability theory: the logic of science*. Cambridge university press, 2003.

Bai Jiang, Tung-Yu Wu, Charles Zheng, and Wing H. Wong. Learning summary statistic for approximate Bayesian computation via deep neural network. *Statistica Sinica*, 27:1595–1618, 2017.

Sanchit Kabra, Nikhil Abhyankar, Saaketh Desai, Prasad Iyer, and Chandan K. Reddy. Llm-autoscilab: Closed-loop scientific discovery via active experimentation with llms. *arXiv*, 2026. URL https://arxiv.org/abs/2605.24043.

Daniel Kasenberg, Pablo Samuel Castro, Maria K Eckstein, Nóemi Éltető, Will Dabney, Caroline Wang, Martin Engelcke, Rishika Mohanta, Aparna Dev, Matthew M Botvinick, Nenad Tomasev, Glenn C Turner, Vincent Costa, Nathaniel D Daw, Kimberly L Stachenfeld, and Kevin J Miller. AI-discovered cognitive models reveal novel insights into human and animal learning. *bioRxiv*, pp. 2026.05.18.725921, May 2026. URL https://www.biorxiv.org/content/10.64898/2026.05.18.725921v1.abstract.

Riko Kelter. Bayesian model selection in the $\mathcal{M}$-open setting — approximate posterior inference and subsampling for efficient large-scale leave-one-out cross-validation via the difference estimator. *arXiv preprint arXiv:2005.13199*, 2020.

Emre Kıcıman, Robert Ness, Amit Sharma, and Chenhao Tan. Causal reasoning and large language models: Opening a new frontier for causality. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=mqoxLkX210. Featured Certification. Preprint: arXiv:2305.00050.

Sophia Koehler, Antonia Wüst, Inga Ibs, Wasu Top Piriyakulkij, Wolfgang Stammer, Constantin Rothkopf, Kevin Ellis, and Kristian Kersting. Playing ZendoWorld: Challenging AI agents on active visual concept induction. *arXiv [cs.AI]*, July 2026. URL http://dx.doi.org/10.48550/arXiv.2607.08233.

Stefan Kramer, Mattia Cerrato, Jannis Brugger, Sašo Džeroski, and Ross D King. Automated scientific discovery: From equation discovery to autonomous discovery systems. *Mach. Learn.*, 115(5):109, May 2026. URL https://link.springer.com/article/10.1007/s10994-025-06955-2.

Mario Krenn, Robert Pollice, Si Yue Guo, Matteo Aldeghi, Alba Cervera-Lierta, Pascal Friederich, Gabriel Dos Passos Gomes, Florian Häse, Adrian Jinich, Akshatkumar Nigam, Zhenpeng Yao, and Alán Aspuru-Guzik. On scientific understanding with artificial intelligence. *Nat. Rev. Phys.*, 4(12):761–769, October 2022. URL https://pmc.ncbi.nlm.nih.gov/articles/PMC9552145/.

Michael Y. Li, Emily B. Fox, and Noah D. Goodman. Automated statistical model discovery with language models. In *International Conference on Machine Learning (ICML)*, 2024a. URL https://arxiv.org/abs/2402.17879.

Wen-Ding Li, Keya Hu, Carter Larsen, Yuqing Wu, Simon Alford, Caleb Woo, Spencer M Dunn, Hao Tang, Michelangelo Naim, Dat Nguyen, Wei-Long Zheng, Zenna Tavares, Yewen Pu, and Kevin Ellis. Combining induction and transduction for abstract reasoning. *arXiv [cs.LG]*, November 2024b. URL http://arxiv.org/abs/2411.02272.

D. V. Lindley. On a measure of the information provided by an experiment. *The Annals of Mathematical Statistics*, 27(4):986–1005, 1956. doi: 10.1214/aoms/1177728069.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The AI scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024.

Peter Machamer, Lindley Darden, and Carl F Craver. Thinking about mechanisms. *Philos. Sci.*, 67 (1):1–25, March 2000. URL https://www.jstor.org/stable/188611.

David J. C. MacKay. Bayesian model comparison and backprop nets. In *NIPS*, pp. 839–846, December 1991. URL https://proceedings.neurips.cc/paper/1991/file/c3c59e5f8b3e9753913f4d435b53c308-Paper.pdf.

Dan MacKinlay. Bayes inference in an open world, May 2016. URL https://danmackinlay.name/notebook/bayes_misspecified.html.

Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, 1943.

Lisa Messeri and M J Crockett. Artificial intelligence and illusions of understanding in scientific research. *Nature*, 627(8002):49–58, March 2024. URL https://www.nature.com/articles/s41586-024-07146-0.

Bruno Mlodozeniec, David Krueger, and Richard E Turner. Probabilistic modelling is sufficient for causal inference. In *ICML*, December 2025. URL http://dx.doi.org/10.48550/arXiv.2512.23408.

Christian A Naesseth, Fredrik Lindsten, and Thomas B Schön. Elements of sequential monte carlo. *Foundations and Trends in Machine Learning*, 2019. URL http://arxiv.org/abs/1903.04797.

George Papamakarios, David Sterratt, and Iain Murray. Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019.

Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2nd edition, 2009.

Wasu Top Piriyakulkij, Cassidy Langenfeld, Tuan Anh Le, and Kevin Ellis. Doing experiments and revising rules with natural language and probabilistic reasoning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Ingmar Posner, Anson Lei, and Bernhard Schölkopf. From observation to insight: Mechanistic world models and the quest for autonomous discovery. *arXiv [cs.AI]*, July 2026. URL http://dx.doi.org/10.48550/arXiv.2607.12474.

Ben Prystawski, Kushin Mukherjee, Daniel Wurgaft, Linas Nasvytis, Michael Y Li, Noah D Goodman, and Michael C Frank. auto-psych: Automating the science of mind using agent-driven theory discovery and experimentation. *arXiv [cs.AI]*, June 2026. URL http://dx.doi.org/10.48550/ARXIV.2606.26460.

Friedrich Pukelsheim. *Optimal Design of Experiments*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics (SIAM), 2006. Reprint of the 1993 Wiley edition.

Stefan T. Radev, Ulf K. Mertens, Andreas Voss, Lynton Ardizzone, and Ullrich Köthe. BayesFlow: Learning complex stochastic models with invertible neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 33(4):1452–1466, 2022.

Tom Rainforth, Adam Foster, Desi R. Ivanova, and Freddie Bickford Smith. Modern Bayesian experimental design. *Statistical Science*, 39(1), 2024.

Georg Rasch. *Probabilistic Models for Some Intelligence and Attainment Tests*. Danmarks Pædagogiske Institut, Copenhagen, 1960.

Mark D. Reckase. *Multidimensional Item Response Theory*. Statistics for Social and Behavioral Sciences. Springer, 2009.

Jonathan Richens and Tom Everitt. Robust agents learn causal world models. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=pOoKI3ouv1. Oral; honorable mention outstanding paper. arXiv:2402.10877.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625:468–475, 2024.

Tom Rossa, Angus Phillips, and Tom Rainforth. Action-BED: Task-driven Bayesian experimental design with singly intractable objectives. *arXiv preprint arXiv:2606.23662*, 2026. URL https://arxiv.org/abs/2606.23662.

Paul K. Rubenstein, Sebastian Weichwald, Stephan Bongers, Joris M. Mooij, Dominik Janzing, Moritz Grosse-Wentrup, and Bernhard Schölkopf. Causal consistency of structural equation models. In *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence (UAI)*. AUAI Press, 2017. arXiv:1707.00819.

Wesley C Salmon. *Scientific explanation and the causal structure of the world*. Princeton University Press, Princeton, NJ, December 1984. URL https://press.princeton.edu/books/paperback/9780691101705/scientific-explanation-and-the-causal-structure-of-the-world.

Matthew Self and Peter Cheeseman. Bayesian prediction for artificial intelligence. In *Proc. UAI*, 1987. URL http://dx.doi.org/10.48550/arXiv.1304.2717.

Thomas Serre and Ellie Pavlick. From prediction to understanding: Will AI foundation models transform brain science? *Neuron*, 2025. URL http://dx.doi.org/10.48550/arXiv.2509.17280.

Galit Shmueli. To explain or to predict? *Stat. Sci.*, 25(3):289–310, August 2010. URL http://projecteuclid.org/euclid.ss/1294167961.

Freddie Bickford Smith, Andreas Kirsch, Sebastian Farquhar, Yarin Gal, Adam Foster, and Tom Rainforth. Prediction-oriented bayesian active learning. In *Proc. 26th Int. Conf. on Artificial Intelligence and Statistics (AISTATS)*, 2023. arXiv:2304.08151.

Zhangde Song, Jieyu Lu, Yuanqi Du, Botao Yu, et al. Evaluating large language models in scientific discovery. *arXiv preprint arXiv:2512.15567*, 2025. SDE framework; https://github.com/HowieHwong/sde-harness.

Kyle Swanson, Wesley Wu, Nash L. Bulaong, John E. Pak, and James Zou. The virtual lab: AI agents design new SARS-CoV-2 nanobodies with experimental validation. *bioRxiv*, 2024. URL https://www.biorxiv.org/content/10.1101/2024.11.11.623004v1.

Silviu-Marian Udrescu and Max Tegmark. AI feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631, 2020.

Stefan Wahl, Raphaela Schenk, Ali Farnoud, Jakob H. Macke, and Daniel Gedon. A probabilistic framework for LLM-based model discovery. *arXiv preprint arXiv:2602.18266*, 2026. URL https://arxiv.org/abs/2602.18266. Introduces ModelSMC.

Marcel Weber. Causes without mechanisms: Experimental regularities, physical laws, and neuroscientific explanation. *Philos. Sci.*, 75(5):995–1007, December 2008. URL https://www.cambridge.org/core/journals/philosophy-of-science/article/abs/causes-without-mechanisms-experimental-regularities-physical-laws-and-neuroscienti 33F39A74D28FCF3C8BF11C3CFAE8AEDF.

Matt L. Wiemann, Lindsay M. Smith, Peter Melchior, Siddharth Mishra-Sharma, Andrew Gordon Wilson, Pavel Izmailov, and Carolina Cuesta-Lázaro. DiscoverPhysics: Benchmarking LLMs for out-of-the-box scientific thinking. *arXiv*, 2026. URL https://arxiv.org/abs/2605.26087.

Simon N. Wood. Statistical inference for noisy nonlinear ecological dynamic systems. *Nature*, 466 (7310):1102–1104, 2010.

James Woodward. *Making Things Happen: A Theory of Causal Explanation*. Oxford University Press, Oxford, England, 2004. URL https://www.amazon.ca/Making-Things-Happen-Theory-Explanation/dp/0195189531.

Hanbo Xie and Robert C Wilson. Successful automatic model discovery can produce false mechanisms. *PsyArXiv*, July 2026. URL https://osf.io/preprints/psyarxiv/r46ux_v1.

Tianshi Zheng, Kelvin Kiu-Wai Tam, Newt Hue-Nam K. Nguyen, Baixuan Xu, Zhaowei Wang, Jiayang Cheng, Hong Ting Tsang, Weiqi Wang, Jiaxin Bai, Tianqing Fang, Yangqiu Song, Ginny Y. Wong, and Simon See. NewtonBench: Benchmarking generalizable scientific law discovery in LLM agents. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2510.07172.