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

Anonymous authors
Paper under double-blind review

ABSTRACT

A primary goal of science is to learn mechanistic world models from limited experimental data, both to explain observations and to predict novel interventions. We introduce the Model Discovery Agent (MDA), which combines LLM proposals for $\mathcal{M}$-open model discovery, experiment design based on Value of Information, and approximate Bayesian inference over model structures, parameters, and stochastic latent trajectories. We apply MDA to learn symbolic reaction rate laws for ChemBench (Kabra et al., 2026), partially observed ODE models for GlucoseBench (Xie, 2018; Kovatchev et al., 2009), and partially observed SDE models for a new stochastic single-neuron simulator we create. In the appendix, we also show results on various other domains from BoxingGym (Gandhi et al., 2025). We show that MDA has improved sample efficiency compared to various baseline methods, and the learned models are good predictors but also provide interpretable abstractions of each domain.

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 can support scientific explanation and prediction (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 action a?" (e.g., predicting the effect of administering a new drug to a patient).

Unfortunately, a model with latent variables/ mechanisms is typically *unidentifiable* from observation alone: the passive data underdetermines it, and interventions — perturbing the system and watching how it responds — can distinguish explanations that passive data cannot. But experiments are expensive (a lab assay, a clinical trial), which makes the operative problem *data efficiency*: identify the model, well enough to answer future queries, in as few experiments as possible. This is the classical remit of *Bayesian experimental design* (BED) — choose the intervention whose outcome is most informative (Lindley, 1956; Chaloner & Verdinelli, 1995; Rainforth et al., 2024) — increasingly combined with hypothesis generation in recent active theory-learning systems (Piriyakulkij et al., 2024; Elteto et al., 2026; Prystawski et al., 2026). Here we study this combination across explicit algebraic models and partially observed deterministic and stochastic dynamics.

We define this problem formally in Section 2. Then in Section 3 we present a solution, which we call the **Model Discovery Agent (MDA)**, which combines three ingredients: approximate Bayesian

[1] 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 (Pearl, 2009). In this paper, we restrict ourselves to "level 2" causality (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). Note that the philosophy of science literature makes a distinction between causal models and mechanistic models (see e.g., (Machamer et al., 2000; Craver, 2006; Woodward, 2004; Weber, 2008; Glennan, 2017; Batterman & Rice, 2014)); we use the term mechanistic broadly here.

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

inference over a hierarchy of models, parameters, and latent trajectories, implemented using particle methods or MAP estimation; 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 (see e.g., (Rainforth et al., 2024)). The loop fits and weights candidate models, selects an experiment, and uses predictive discrepancies to guide further proposals.

In Section 4, we empirically evaluate MDA on problems of increasing complexity, and compare to various baselines. On CHEMBENCH (Kabra et al., 2026), MDA discovers static symbolic rate laws for chemical reactions. On GLUCOSEBENCH (Xie, 2018; Kovatchev et al., 2009), MDA learns partially observed deterministic ODEs, including latent compartments that encode delayed input responses. Finally, on NEURONBENCH, a novel single-neuron electrophysiology benchmark we introduce based on a stochastic version of the Hodgkin-Huxley (HH) model, MDA learns a partially observed SDE. In the appendix (Section E), we also report results of applying MDA to BOXING-GYM (Gandhi et al., 2025), which contains a variety of simple problems from psychology to ecology, which we model using Generalized Linear Models and probabilistic programs. Across these benchmarks, MDA discovers interpretable models which provide strong predictive performance on novel inputs and interventions.

2 PROBLEM STATEMENT

Agent-environment interface. We consider an agent interacting with an unknown "blackbox" dynamical system (c.f., (Ljung, 1998)). The outputs or observations are $y_{1:T}$, for $y_t \in \mathcal{Y} \subset \mathbb{R}^{d_y}$. The inputs or control signals $a_{1:T}$ are chosen by the agent using $a_t \sim \pi(\cdot \mid a_{1:t-1}, y_{1:t-1}, t)$, with $a_t \in \mathcal{A} \subset \mathbb{R}^{d_a}$, where π is the policy.[2] The agent can also optionally specify the initial condition of the system, z_0; if it is not specified, its distribution must be inferred from the training data and context. Finally, the agent can optionally apply a perturbation or intervention δ that modifies the system parameters, yielding $\theta' = \mathrm{do}(\delta, \theta)$. (An omitted perturbation leaves the system unchanged.) We define an experiment design as the tuple $\chi = (\delta, z_0, \pi)$.

Experimental protocol. The agent is presented with some background context C, and an optional initial dataset $\mathcal{D}_0 = \{(\chi^i, y^i_{1:T}) : i = 1 : N_0\}$, where each sample is drawn from the system using $y^i_{1:T} \sim p^*(\cdot|\chi^i)$. (We also record the sampled inputs $a^i_{1:T}$, but omit this from notation, for brevity.) The agent is then given a budget of K turns to interact with the system. At each step, it designs an experiment $\chi^k = (\delta^k, z^k_0, \pi^k)$, records one trajectory (episode) $y^k_{1:T}$ from the environment, and updates $\mathcal{D}_{0:k} = \mathcal{D}_0 \cup \{(\chi^i, y^i_{1:T}) : i = 1 : k\}$. The agent can use this data to update its beliefs about the underlying model, $p_k(m) = p(m \mid C, \mathcal{D}_{0:k}, \mathcal{M}_k)$, where the candidate support $\mathcal{M}_k$ may grow or shrink, and this belief can be used to design the next experiment, and to make predictions about the future. The objective is low held-out forecasting loss within the experiment budget; MDA uses model-discrimination information gain as a tractable acquisition surrogate (Eq. (59)), not the held-out outcomes themselves. See Algorithm 1 for the pseudocode for this experimental protocol, that is agnostic to the form of the agent.

Conditional forecasting and evaluation. We evaluate predictions on held-out experiments, not recovery of a unique true mechanism, since distinct latent models can have similar interventional predictions (Beckers & Halpern, 2019; Dyer et al., 2023; Cornish et al., 2026). Given a new design, $\chi \sim \mathcal{Q}$, the oracle rolls out the policy to collect the ground truth trajectory $(a_{1:T}, y_{1:T})$. Let t_c be a cut point, through which the agent observes the initial history $\mathcal{H}_{t_c} = (a_{1:t_c}, y^{\mathrm{obs}}_{1:t_c})$, including observation masks when entries are missing. (If $t_c = 0$, the agent does not see any such prefix.) The agent must then predict a distribution over (functions of) future observables, given this partial history $\mathcal{H}_{t_c}$, the training data $\mathcal{D} = \mathcal{D}_{0:k}$, and any relevant context or metadata C, by computing the

[2] For interventional prediction, we assume actions depend only on recorded history and context (including state estimates computed from them), with policy randomization independent of system noise. This excludes unmeasured confounding of action selection; causal identification additionally requires adequate action coverage and suitable model/identification assumptions (Cornish et al., 2026). Without these, same-policy observational prediction may still be accurate, but need not transfer to a new intervention or policy.

posterior predictive distribution:[3]

$$p(F(Y_{t_c+1:T}) \mid \chi, \mathcal{H}_{t_c}, \mathcal{D}, C)\,. \tag{1}$$

The target function F maps the future observation sequence to a scalar, feature vector, or trajectory, depending on the benchmark. For CHEMBENCH it is the scalar log-rate $F(y) = \log(1+y) \in \mathbb{R}$, matching nMSLE; for GLUCOSEBENCH it is the future observed trajectory $F(y_{t_c+1:T}) = y_{t_c+1:T} \in \mathbb{R}^{(T-t_c)d_y}$, where $d_y = 2$ is the number of observed variables per step; and for NEURONBENCH, it is a vector of summary features, $F(y_{1:T}) \in \mathbb{R}^6$ (e.g., neuron spike count — see Eq. (90) for details). We condition on the observed history so far $\mathcal{H}_{t_c}$ for this unit (if $t_c > 0$), plus the training episodes $\mathcal{D}_{0:k}$ (collected by the agent), plus any supplied context or metadata C.

We mostly focus on predicting the mean of Eq. (1), and measuring performance with squared error, although we do consider distributional forecasting metrics in Section D.2. In the simplest setting, where the prediction target is the raw observation sequence, $F(y_{1:T}) = y_{1:T}$, there is no missing data, and $t_c = 0$ (so there is no history prefix), the normalized mean squared error metric we use is

$$L_{\text{nMSE}} = \frac{1}{d_y} \sum_{i=1}^{d_y} \frac{\sum_{q=1}^{Q} \sum_{t=1}^{T} (\hat{y}_{qti} - y_{qti})^2}{N s_i^2} \tag{2}$$

Here q indexes held-out queries (novel experiments/designs), i observation channels, t future time points, y_{qti} is the observation. $\hat{y}_{qti}$ is the agent's prediction, s_i is a scaling factor that is benchmark specific, and $N = QT$ is the number of observed time points in the test set. We give the general case in Section A.2.

3 METHOD

Overview. The MDA method is visualized in Fig. 5; see Algorithm 2 for detailed pseudocode. At each step, the agent updates its belief state $p_k = p(m|\mathcal{D}_{0:k})$, which is a posterior distribution over models or hypotheses m, represented as a weighted finite set. MDA then chooses the next experiment by maximizing the value of information, $\chi_{k+1} = \arg\max_{\chi \in \mathcal{X}} \text{VoI}(\chi, p_k)$. It runs the experiment and updates its dataset by appending $\mathcal{D}_{k+1}$. After K rounds, the agent is asked to forecast the outcomes to some novel experimental conditions. We describe this in more detail below, but for full details, see Section A.

The three-level hierarchy. In the general case, MDA represents the unknown world by the 3 level hierarchy corresponding to the State Space Model shown in Fig. 1. In equations, this corresponds to

$$m \sim p(m), \qquad \theta \sim p(\theta \mid m), \qquad \theta' = \text{do}(\delta, \theta) \tag{3}$$

$$z_t \sim p_m(z_t \mid z_{t-1}, a_t; \theta')\,, \qquad y_t \sim p_m(y_t \mid z_t; \theta'), \tag{4}$$

for $t = 1{:}T$, with z_0 specified by χ when available (otherwise assigned an initial-state distribution), and a_t selected by π. The discrete structure m specifies what the latent variables z_t are, and how they depend on each other, as well as the functional forms for all the distributions; θ specifies the numerical parameters; $z_{1:T}$ is the latent trajectory; and $y_{1:T}$ is what is measured. MDA must perform inference over structures, parameters, and —when dynamics or initial states are uncertain—latent paths.

Computing the posterior over models. The belief state over models, $p_k = p(m|\mathcal{D}_{0:k}) \propto p(m)p(\mathcal{D}_{0:k} \mid m)$, is represented as a discrete distribution over a finite candidate pool. This is computed using Bayes' rule over the models in the pool, and then keeping the top N_m, if a cap on pool size is imposed.

[3] If χ specifies an open-loop policy π, then it defines the whole input sequence $a_{1:T}$ up front (so the policy is just a function of time, not the past observations or actions). If π is a closed-loop policy, then future inputs $a_{t_c+1:T}$ must be generated jointly with future observations $y_{t_c+1:T}$, when forecasting the future. (This paper focuses on learning action-conditioned world models, not on learning the policy of another agent from observational data.)

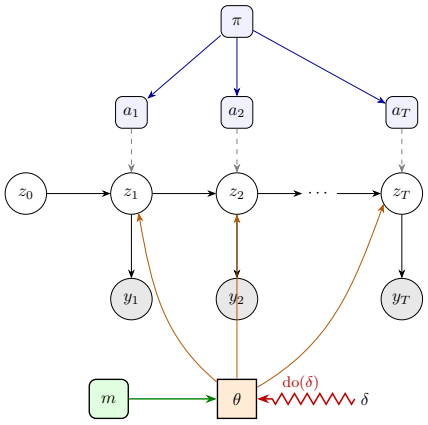


Figure 1: **The three-level hierarchy** (Eq. (4)). A model structure m (green) selects a parameterized mechanism θ (orange), which governs the latent trajectory $z_{1:T}$ (white); the trajectory emits the noisy observations $y_{1:T}$ (grey). The experiment may specify z_0 directly (otherwise it is inferred) and specifies a policy π that selects inputs $a_{1:T}$ (blue). For closed-loop control, each a_t also depends on past actions and observations (feedback edges omitted); our experiments use open-loop policies. An intervention $\mathrm{do}(\delta)$ (lightning bolt) changes the parameters from θ to θ'.

Expanding the hypothesis space. Bayes rule 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 acknowledge that our initial hypothesis class may be insufficient to capture the truth, so we need a way to grow it on demand. To do this, we 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 one or more novel models conditioned on the observations and context. Where available, the prompt also includes the current model pool and residual diagnostics. Note that this hypothesis expansion step is the only part of MDA that uses an LLM. After expanding the pool of models, we compute the evidence for each new model and renormalize, keeping the top N_m models. Crucially, we do not require the LLM to propose the right model in a single shot; instead, it can suggest local mutations to previous hypotheses, and thus can iteratively refine the candidate library by performing R_m iterations of expansion, scoring and pruning. See Algorithm 2 for details.

Shrinking the hypothesis space. If the highest model weight is sufficiently high, and if its error is sufficiently low, then we can optionally shrink the pool down to a minimal size, by adapting the value of N_m. This limits computational cost and candidate proliferation. Shrinkage is gated by confidence and fit quality, but can still discard a model that becomes useful later. Maintaining a diverse library of hypotheses is important, even if the posterior is very concentrated on just a single model, since it improves the search process, similar to other evolutionary optimization algorithms such as FunSearch (Romera-Paredes et al., 2024).

Computing the marginal likelihood (evidence). To compute the posterior over models, we need to estimate the marginal likelihood or evidence for each model, $Z_m^k = p(\mathcal{D}_{0:k}|m) = \int p(\mathcal{D}_{0:k}|m,\theta)\, p(\theta|m)\, d\theta$, where the likelihood factorizes over the trajectories: $p(\mathcal{D}_{0:k}|m,\theta) = \prod_{i\in\mathcal{D}_{0:k}} p(y^i_{1:T}|\chi^i, m, \theta)$. We estimate Z_m^k using tempered SMC in Section A.6.1 or the BIC approximation in Section A.6.3. The Bayesian evidence averages the likelihood over the declared parameter prior (MacKay, 1991), which provides an automatic Occam penalty that depends on both model flexibility and the parameter prior. However, for some problems we also add an explicit complexity penalty by using the model prior $p(m) \propto e^{-\lambda C_m}$, where C_m is the number of free parameters in m and λ is tuned on a validation set. (By default we use $\lambda = 0$, corresponding to a uniform prior.)

Computing the likelihood. For latent variable models, computing the likelihood of a single trajectory, $p(y_{1:T} \mid \chi, m, \theta) = \int \prod_{t=1}^{T} p(y_t \mid z_t, m, \theta)\, p(z_t \mid z_{t-1}, a_t, m, \theta)\, dz_{1:T}$, may be intractable.

For this we mostly use another layer of SMC, namely the bootstrap particle filter in Section A.5.1, although we also compare to the Bayesian Synthetic Likelihood method in Section A.5.2.

Experiment design. We choose the experiment whose outcome is most informative about which hypothesis is true: $\chi^\star = \arg\max_{\chi \in \mathcal{X}} I(M; Y_\chi \mid \mathcal{D})$ (Lindley, 1956; Box & Hill, 1967; Chaloner & Verdinelli, 1995; Rainforth et al., 2024). This is called the Expected Information Gain (EIG), which is a special case of Value of Information (VoI). In practice we use the model-disagreement surrogate in Eq. (59), which is an approximation to the mutual information. We optimize the VoI surrogate using CMA-ES (Hansen, 2016) for CHEMBENCH, which has 7 real-valued inputs, and exact enumeration of finite action menus for GLUCOSEBENCH and NEURONBENCH. See Section A.9 for details.

Prediction. After each step, we convert the belief state, which is a distribution over models, $p_k = p(m|\mathcal{D}_{0:k})$, to a predictive distribution in observation space, $\rho_k(\chi) = p(Y|\chi, \mathcal{D}_{0:k})$, 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. (2):

$$\hat{F}_k(\chi) = \mathbb{E}[F(Y)|\chi, \mathcal{D}_{0:k}] = \sum_m \int \Big[\int F(y)\, p(y|m, \theta, \chi)\, dy \Big]\, p(m, \theta|\mathcal{D}_{0:k})\, d\theta \tag{5}$$

For conditional forecasting (where $t_c > 0$), the inner predictive distribution also conditions on the query prefix in Eq. (1); this can be implemented by estimating the latent state at the forecast cut point t_c. We also consider simplifications of this, where we just plugin the MAP model $\hat{m}_k$ and its corresponding parameters, $\hat{\theta}_k$, instead of averaging over them. For $F(y) = y$, the prediction becomes the plugin posterior predictive mean $\hat{F}_k(\chi) = \mathbb{E}[Y \mid \hat{m}_k, \hat{\theta}_k, \chi]$. For general target functions, we can sample from the posterior predictive, and then approximate $E[F(Y)]$ by Monte Carlo.

4 EXPERIMENTAL RESULTS

We organize the results by order of increasing model complexity. CHEMBENCH requires fitting static symbolic laws; glucose forecasting requires learning partially observed deterministic ODEs; and NEURONBENCH requires prediction from partially observed SDEs. In each case we evaluate held-out interventional prediction. We plot test performance vs the total number of experiments at each step k, which is $B_k = N_0 + k$, where N_0 are the number of initial experiments in $\mathcal{D}_0$. The final MDA budgets are $3 + 22$, $2 + 4$, and $2 + 8$ in chemistry, glucose, and HH, respectively. Additional results, on BOXINGGYM, can be found in Section E.

Baselines. We compare MDA to two main classes of baseline. The first class are methods based on *induction*, which fit a model $(\hat{m}, \hat{\theta})$ given $\mathcal{D}$, and then predict using this model. For CHEMBENCH, we use the method from their paper, and for GLUCOSEBENCH, we fit ARX and sparse polynomial dynamics (SINDYc). We also consider two baseline approaches based on *transduction*, which use an LLM to directly predict the output Y given the test input χ and all the training data $\mathcal{D}$, i.e., they ask the LLM to generate $\arg\max_y p(y \mid \chi, \mathcal{D})$ using in-context learning (ICL). Surprisingly, (Li et al., 2024b) showed that the transductive approach can outperform the inductive approach on some ARC AGI problems, so this is not a trivial baseline.[4] to The experiments used to create the data $\mathcal{D}$ can either be chosen by the LLM itself (which we call "ICL (+LLM design)", c.f., (Gupta et al., 2025)), or by the MDA agent (which we call "ICL (+MDA design)"); this distinction isolates the effects of experiment design from prediction.

LLMs. We use GPT-5.6-Luna at medium reasoning effort as our LLM for all the agents used in the main paper (except the LLM-free ARX baseline for GLUCOSEBENCH). However, some older

[4]Induction and transduction use compute in different ways. For induction, most of the time is spent in numerical model fitting, and the LLM calls (for proposing new models) are only a small fraction of the overall time. (The model fitting uses CPUs, which we run locally on a laptop or sometimes on Modal; the LLM calls (which we access via OpenRouter) use hardware accelerators of various forms, such as GPUs.) For transduction, all of the time is spent inside the LLM. It is therefore hard to match compute budgets. However, our focus in this paper is on sample efficiency, not compute effiency.

	Michaelis–Menten	MM + competitive inhibition + Arrhenius	Fractal kinetics
Truth	$\frac{kEC_A}{K_m + C_A}$	$\frac{kEC_A e^{E_a(1/310-1/T)}}{K_m(1+C_I/K_i)+C_A}$	kEC_A^{α}
MDA	$\frac{kEC_A}{K_m + C_A}$ (.002)	$f^\star s_B(C_B) q_P(C_P) q_{\mathrm{pH}}(\mathrm{pH})$ (.033)	$\frac{kEC_A^n}{K_h^n + C_A^n}$ (.115)
LLM-AUTOSCILAB	$\frac{5.114EC_A}{.102 + C_A}$ (.011)	$1.49\times10^{-69} C_A^{.673} C_I^{-.481} C_P^{-.007}$ $E^{.065} T^{25.804} \mathrm{pH}^{4.059}$ (.750)	$1.90 C_A^{.353} C_I^{.002} C_B^{-.001} C_P^{-.002}$ $E^{.998} T^{.275} \mathrm{pH}^{-.055}$ (.018)

Table 1: **Illustrative CHEMBENCH laws at $B = 15$.** Parentheses: RMSLE (red exceeds the 0.05 pass threshold). $f^\star$ denotes the true law above; $s_B, q_P, q_{\mathrm{pH}}$ are extra fitted factors.

results in the appendix (in particular, the BOXINGGYM results in Section E), used Claude-Opus-4.7. We have not systematically studied the effects of LLM choice due to cost, and since this is orthogonal to the concerns of this paper. None of the agents have internet access, or the ability to read source code or data files. Prompts are in Section G.

4.1 CHEMBENCH: DISCOVERING ENZYME-KINETIC RATE LAWS

Benchmark. CHEMBENCH is derived from (Kabra et al., 2026). The task is to learn a static algebraic law mapping seven controllable inputs—substrate, inhibitor, second-substrate and product concentrations, enzyme loading, temperature, and pH—to the initial rate of a chemical reaction:

$$r = f(C_A, C_I, C_B, C_P, E, T, \mathrm{pH}; \theta).$$

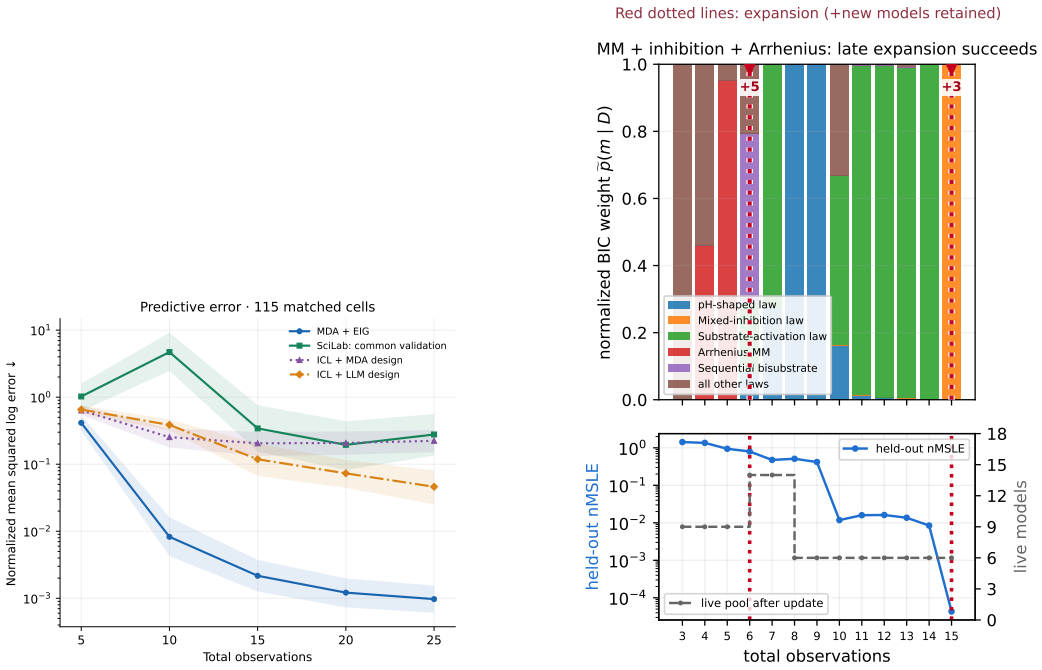
The complete benchmark contains 57 worlds: nine canonical mechanisms and 48 compound mechanisms. We use the medium difficulty level for all worlds. We evaluate held-out prediction error using RMSLE, as defined in Eq. (67), and its normalized nMSLE version, as defined in Eq. (68). An agent is said to "pass" a given world if $\mathbb{1}[\mathrm{RMSLE} < 0.05]$. See Section B for further details.

Methods. MDA is initialized with three scalar rate observations in $\mathcal{D}_0$, which are created by varying $C_A \in \{0.03, 0.3, 3\}$ and holding the other terms constant. It then actively selects $K = 22$ additional experiments. We initialize the hypothesis space with nine canonical mechanisms plus up to 12 valid, nonduplicate LLM proposals; later triggered calls request six candidates, with a live-pool cap of 24. We fit each model by bounded, multistart nonlinear least squares, and approximate the marginal likelihood using the BIC approximation in Eq. (45). We approximate the model posterior using $p(m \mid \mathcal{D}) \propto e^{-\mathrm{BIC}_m/2}$. The predictor uses the highest-weight model and its point estimate $\hat{\theta}_m$. Experiment design uses the derivative-free CMA-ES algorithm of (Hansen, 2016) to maximize the weighted between-model disagreement in Eq. (69), which is an approximation to EIG. (See Fig. 7 for some evidence that these EIG designs outperform random designs.) For the main baseline, we use the LLM-AUTOSCILAB agent from (Kabra et al., 2026), which combines an LLM with the evolutionary symbolic regression method PySR (Cranmer, 2023), independently fitted at each budget $B \in \{5, 10, 15, 20, 25\}$. For the ICL+MDA baseline, we reuse MDA's data, and for ICL+LLM, we start with $\mathcal{D}_0$ but then use the LLM to choose the next 22 experiments.

Results. Figure 2a shows that MDA is substantially more sample efficient than the other methods. (The small error spike at $B = 10$ for LLM-AUTOSCILAB is explained in Section B.3.) Figure 2b visualizes the posterior over models over time created by MDA on a specific example, along with the corresponding test error. At steps 6 and 15, an inadequate fit triggers an expansion of the hypothesis space; the second such expansion adds the orange model, and its selection sharply reduces error. Illustrative laws from MDA and LLM-AUTOSCILAB are shown in Table 1.

4.2 GLUCOSEBENCH: DISCOVERING LATENT ODE MODELS

Benchmark. GLUCOSEBENCH modifies `simglucose` (Xie, 2018), a Python implementation of the UVA/Padova type-1 diabetes model (Kovatchev et al., 2009). The original simulator was FDA-accepted for preclinical insulin-control testing in place of animal trials (Cobelli & Kovatchev, 2023). Its 30 virtual patients (ten per age group) have hidden 13-dimensional nonlinear ODE states. Agents observe noisy glucose every five minutes and insulin every 15 minutes over six hours; simulator variables and parameters remain hidden. After two initial episodes, agents select four more from 48 meal amount/time and bolus amount/delay templates. We evaluate twelve unseen interventions for the *same patient*, conditioned on a 45-minute prefix; other cuts and details appear in Section C.



(a) Held-out predictive error.

(b) One $\mathcal{M}$-open discovery trajectory.

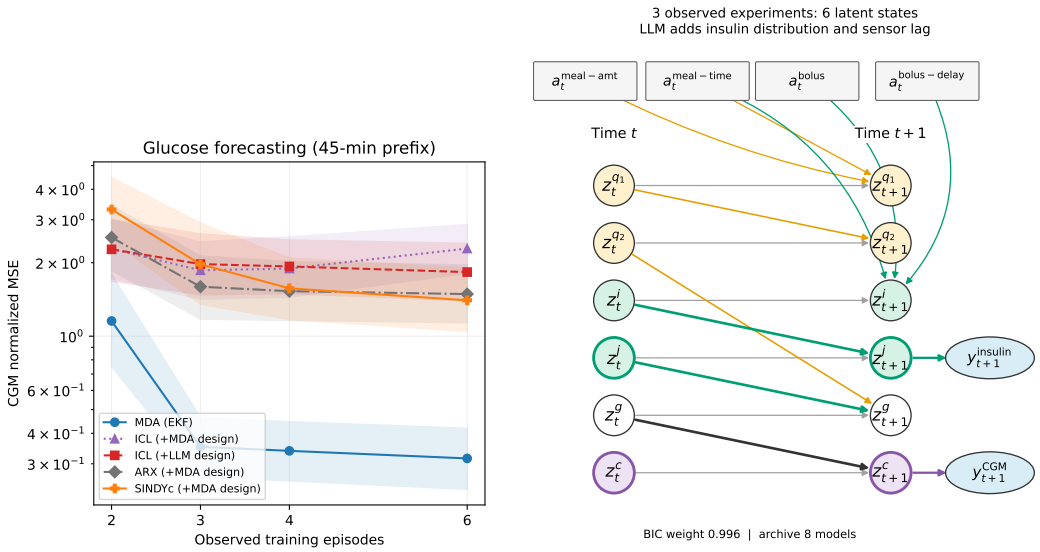
Figure 2: **CHEMBENCH: data efficiency and model discovery.** (a) Held-out predictive nMSLE. Errors are averaged arithmetically over seeds within world, then geometrically over worlds; bands are 95% world-bootstrap intervals. All methods use the same 115 world–seed cells spanning 54 worlds where the common-validation LLM-AUTOSCILAB variant is evaluable at every budget (of 171 attempted; Section B.3). (b) A representative MM + competitive-inhibition + Arrhenius world. Stacked bars show normalized BIC model weights, colors track the five laws attaining the most posterior mass (whether seeded or proposed), red dotted lines mark $\mathcal{M}$-open expansion (labels count new models retained after pruning, not net pool growth). The lower panel shows held-out nMSLE and live-pool size after each update, including adaptive shrinkage at step $B = 8$, when the pool reduced from 14 to 6, and at $B = 15$, when 3 new laws replace 3 old ones, leaving 6 in total.

Methods. MDA starts with four compartment ODEs and can add mechanisms and latent states (up to six per model). It fits parameters numerically, weights models using BIC (Eq. (45)), and selects experiments with noise-normalized disagreement (Eq. (77)). An EKF conditions states on the query prefix before BIC-weighted prediction (Eq. (76)). Baselines are the two ICL methods, an ARX model for each observation channel (Section C.3), and SINDYc sparse polynomial dynamics (Brunton et al., 2016a;b). ARX and SINDYc are fitted to MDA's data, since they cannot desgin their own experiments. Each patient is fitted separately (a single shared model is future work).

Results. Figure 3(a) shows that MDA is much more sample efficient than the other methods in terms of predicting future GCM given a prefix of $t_c = 45$ minutes (see Fig. 8 for the effect of changing the prefix length). Figure 3(b) illustrates a 6-state model learned by MDA after 3 trajectories. It shows the sparse latent connectivity and added memory nodes, which retain the effects of transient inputs.

4.3 NEURONBENCH: DISCOVERING MECHANISMS WITH STOCHASTIC LATENT DYNAMICS

Benchmark. NEURONBENCH is a new benchmark we created based on generalized stochastic Hodgkin–Huxley models. Each world uses one of six "mystery neurons", five with nonstandard mechanisms and one with a textbook M-current comparator. Finite populations of ion channels introduce intrinsic gating noise, turning the hidden voltage-and-gate dynamics into an SDE. The agent observes 20 noisy voltage samples per experiment. The design space consists of 90 structured electrical pulse protocols which specify the input current over time. We score held-out predictions



(a) Held-out CGM prediction.

(b) A learned sparse SSM.

Figure 3: **GLUCOSEBENCH: prediction and learned latent structure.** Left: initial-training-variance-normalized CGM MSE after a 45-minute prefix, geometric means across all 30 patients with 95% patient-bootstrap intervals; GPT-5.6 Luna (medium). Right: a six-state model proposed and selected after three episodes. Bold nodes/edges mark added insulin-memory and sensor-lag states.

using a six-dimensional functional $F(y_{1:T})$ containing spike-count, adaptation, run-down, and sub-threshold voltage features, defined in Eq. (90). See Section D for further details.

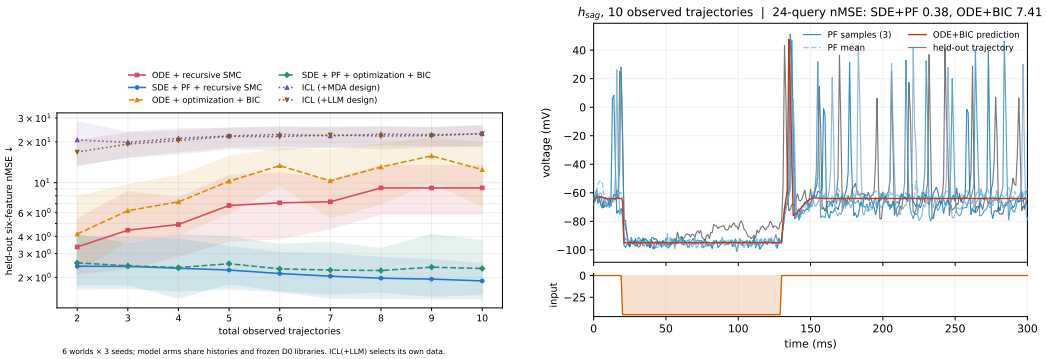
Methods. A model m specifies which ion-channel mechanisms are present, parameters θ include their conductances and kinetics, and the latent path $z_{1:T}$ contains voltage and unobserved channel gates. We consider 4 MDA variants: stochastic dynamics (SDE) vs deterministic dynamics (ODE), crossed with recursive SMC estimation of the parameter posterior and evidence (Section A.6.2) vs MAP parameter estimation plus the BIC evidence approximation (Section A.6.3). The ODE arms use the Gaussian likelihood in Eq. (24), and the SDE arms estimate the likelihood with a bootstrap particle filter (Section A.5.1); the likelihood is then used as input to SMC parameter posterior estimation or parameter point estimation. We also compare to the two ICL baselines.

Results. Figure 4a shows that MDA with stochastic dynamics (and either SMC or BIC estimate for marginal likelihood) works much better than deterministic ODE approximations, and both work better than the two ICL baselines. Figure 4b shows that stochastic sample paths retain spiking variability that the fitted ODE misses. These predictions do not exactly match the precise spike times of the single sample ground truth rollout, but they suffice to get low nMSE score of 0.38 for the feature summary (aggregated over 24 queries) for the SDE+SMC method, compared to 7.41 for the ODE+BIC method.

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) established a theoretical connection between robust prediction over a rich intervention class and causal models; however, our finite held-out tests do not establish causal identification. There is also much recent work on causality and LLMs (see, e.g., Kıcıman et al., 2024; Ban et al., 2025). We choose to use an LLM to explicitly represent causal mechanisms, rather than performing implicit reasoning, since a key goal is to discover an interpretable model. Our model-based pipelines outperform the tested ICL forecasters, but also use numerical fitting and simulation; this is not a compute-matched isolation of induction versus transduction.

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



(a) Controlled inference comparison.

(b) Posterior-predictive trajectories.

Figure 4: **NEURONBENCH: stochastic inference and its predictive consequences.** (a) Held-out six-feature nMSE for four MDA variants (ODE/SDE × recursive SMC/optimization+BIC) and two ICL baselines. To isolate inference, all four MDA variants use the same LLM-generated model archive and experimental history collected by the SDE–SMC parent. ICL (+MDA design) uses the same history; ICL (+LLM design) chooses its own actions. All curves use the same six worlds and three seeds; lines are geometric means and shading gives pointwise 95% hierarchical-bootstrap intervals. SMC uses 128 parameter particles; SDE likelihoods use 128 PF particles. Predictions sample the full model–parameter mixture, without top-component truncation. (b) An illustrative trajectory from the I_h-sag world for a particular input stimulus, along with predictions after observing 10 trajectories: PF sample paths (blue), PF mean (dashed), ODE+BIC prediction (red), held-out trajectory (gray), and injected current below. The title reports aggregate error across 24 queries, not the error of this selected illustrative trajectory.

- **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, 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). MDA builds on SMC-S's pool-conditioned symbolic proposals (Piriyakulkij et al., 2024) and shares mechanistic population scoring with ModelSMC (Wahl et al., 2026). ATLAS (Elteto et al., 2026) similarly uses active discrimination among RNN hypotheses to recover agents in bandit tasks. Our emphasis is a common executable-model interface across algebraic, ODE, and stochastic latent-dynamics tasks, with controlled inference and acquisition comparisons (see Section F for SINDy and other numerical approaches).
- **Benchmarks for interactive scientific discovery.** Various benchmarks evaluate agents that learn scientific laws by *interactive experimentation*: we build on ACTIVESCIBENCH-CHEM (Kabra et al., 2026), and BOXINGGYM (Gandhi et al., 2025), and add a glucose study and our own NEURONBENCH. Other relevant benchmarks include DISCOVERPHYSICS (Wiemann et al., 2026), NEWTONBENCH (Zheng et al., 2026), SCIENCE-GYM (Cerrato et al., 2026), and SCIGYM (Duan et al., 2025).

6 DISCUSSION

MDA combines LLM-proposed executable libraries with approximate Bayesian inference and experimental design, to learn models with predictively useful abstractions. In the future we would like to improve inference efficiency, using methods such as amortized SBI (Cranmer et al., 2020; Radev et al., 2022; Deistler et al., 2025), as well as tackle harder benchmarks by giving the LLM more tools for analysing data (rather than putting everything in the prompt) and for proposing a library of reusable model mechanisms.

7 AI USE STATEMENT

In this work, we used generative AI tools to help write the code, launch and analyse experiments, and help with writing and figure preparation.

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.

Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016a. doi: 10.1073/pnas.1517384113.

Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Sparse identification of nonlinear dynamics with control (SINDYc). *arXiv preprint arXiv:1605.06682*, 2016b.

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.

Tingting Chen, Srinivas Anumasa, Beibei Lin, Vedant Shah, Anirudh Goyal, and Dianbo Liu. Auto-Bench: An automated benchmark for scientific discovery in LLMs. *arXiv preprint arXiv:2502.15224*, 2025. URL https://arxiv.org/abs/2502.15224.

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.

Claudio Cobelli and Boris Kovatchev. Developing the UVA/Padova type 1 diabetes simulator: Modeling, validation, refinements, and utility. *J. Diabetes Sci. Technol.*, 17(6):1493–1505, November 2023. URL https://journals.sagepub.com/doi/10.1177/19322968231195081.

Rob Cornish, Muhammad Faaiz Taufiq, Arnaud Doucet, and Chris Holmes. Causal falsification of digital twins. *J. Mach. Learn. Res.*, 2026. URL http://dx.doi.org/10.48550/arXiv.2301.07210.

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.

Miles Cranmer. Interpretable machine learning for science with PySR and SymbolicRegression.jl. *arXiv preprint arXiv:2305.01582*, 2023.

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.

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.

Joel Dyer, Nicholas Bishop, Yorgos Felekis, Fabio Massimo Zennaro, Anisoara Calinescu, Theodoros Damoulas, and Michael Wooldridge. Interventionally consistent surrogates for agent-based simulators. *arXiv [cs.MA]*, December 2023. URL http://dx.doi.org/10.48550/arXiv.2312.11158.

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.

Rushil Gupta, Jason Hartford, and Bang Liu. LLMs for experiment design in scientific domains: Are we there yet? In *ICML 2025 Generative AI and Biology (GenBio) Workshop*, July 2025. URL https://openreview.net/forum?id=dIEeOwrmOe.

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

Jakob Hartmann, James Harvey, Jhonathan Navott, Erik Y. Wang, Luckeciano C. Melo, Flaviu Cipcigan, Cheng Zhang, and Alessandro Abate. Amortising bayesian experimental design for sequential information gathering in LLMs. *arXiv preprint arXiv:2607.03426*, 2026. URL https://arxiv.org/abs/2607.03426.

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.

Alan L. Hodgkin and Andrew F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, 117(4):500–544, 1952. doi: 10.1113/jphysiol.1952.sp004764.

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.

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.

Boris P. Kovatchev, Marc Breton, Chiara Dalla Man, and Claudio Cobelli. In silico preclinical trials: A proof of concept in closed-loop control of type 1 diabetes. *Journal of Diabetes Science and Technology*, 3(1):44–55, 2009. doi: 10.1177/193229680900300106.

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.

Lennart Ljung. *System identification: Theory for the user*. Prentice Hall information and system sciences series. Prentice Hall, Philadelphia, PA, 2 edition, December 1998. URL https://www.amazon.com/dp/0136566952.

F. Llorente, L. Martino, E. Curbelo, J. Lopez-Santiago, and D. Delgado. On the safe use of prior densities for Bayesian model selection. *WIREs Computational Statistics*, 2022. doi: 10.1002/wics.1595.

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.

Helmut Lütkepohl. *New Introduction to Multiple Time Series Analysis*. Springer, 2005. doi: 10.1007/978-3-540-27752-1.

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.

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.

Simo Särkkä and Lennart Svensson. *Bayesian Filtering and Smoothing*. Cambridge University Press, 2 edition, 2023. doi: 10.1017/9781108917407.

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.

Parshin Shojaee, Ngoc-Hieu Nguyen, Kazem Meidani, Amir Barati Farimani, Khoa D Doan, and Chandan K. Reddy. LLM-SRBench: A new benchmark for scientific equation discovery with large language models. In *Proceedings of the International Conference on Machine Learning*, pp. 55325–55359, 2025. URL https://proceedings.mlr.press/v267/shojaee25a.html.

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.

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-neuroscientific-explanation/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.

Jinyu Xie. Simglucose v0.2.1, 2018. URL https://github.com/jxx123/simglucose. Python implementation of the 2008 UVA/Padova simulator; accessed September 2026.

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.

A METHOD: FURTHER DETAILS

A.1 THE AGENT-ENVIRONMENT INTERFACE

Algorithm 1 sketches the agent/environment interface at a high level.[6]

Algorithm 1 The agent/environment interface.

env ← ENV.INIT()
C ← env.DESCRIBE()
$\mathcal{D}_0$ ← env.INIT-DATA()
F ← env.TARGET() ▷ the evaluation functional
agent ← AGENT.INIT(LLM, $C, \mathcal{D}_0, F$)
$\ell_{\text{pred}}(0)$ ← env.EVAL-PREDICTOR(agent.GET-PREDICTOR()) ▷ hold-out predictive loss
for $k = 1 : K$ **do**
    χ_k ← agent.DESIGN()
    obs_k ← env.STEP(χ_k)
    agent.UPDATE-STATE(χ_k, obs_k)
    $\ell_{\text{pred}}(k)$ ← env.EVAL-PREDICTOR(agent.GET-PREDICTOR()) ▷ Algorithm 2
end for

function EVAL-PREDICTOR($\hat{F}$) ▷ score a predictor on the held-out query set $\mathcal{Q}$
    $\hat{f}_q \leftarrow \hat{F}(\chi_q, \mathcal{H}_{t_{c,q}})$ for each held-out query q
    return normalized loss against independent references f_q^{ref} ▷ Eq. (7)
end function

A.2 EVALUATION

Let $\hat{P}_q$ be the agent's predictive distribution for query q, conditional on χ_q, its observed prefix $\mathcal{H}_{t_{c,q}}$, training data $\mathcal{D}$, and context C. The squared-error point prediction is

$$\hat{f}_q = \mathbb{E}_{\hat{P}_q}[F(Y_{t_{c,q}+1:T})]. \tag{6}$$

For nonlinear features, this is $\mathbb{E}[F(Y)]$, not $F(\mathbb{E}[Y])$. Write its entries as $\hat{f}_{qji}$, where $i = 1, \ldots, d_F$ indexes feature components, or observed channels for trajectory targets (then $d_F = d_y$); j indexes future times and is a singleton for a scalar or fixed-dimensional feature vector. In general we use the following normalized mean squared error metric:

$$L_{\text{nMSE}} = \frac{1}{d_F} \sum_{i=1}^{d_F} \frac{\sum_{q,j} o_{qji}(\hat{f}_{qji} - f_{qji}^{\text{ref}})^2}{N_i s_i^2}, \qquad N_i = \sum_{q,j} o_{qji}. \tag{7}$$

Here q indexes held-out queries (novel experiments/designs), i observation channels or feature dimensions, j future time points (a singleton for scalar or feature targets), o is an observation mask (all ones when fully observed), with $N_i > 0$ for each scored component, and s_i is a scaling factor that is benchmark specific. (Chemistry specifies s_i in Eq. (68); Glucose specifies s_i in Eq. (73); HH specifies s_i in Eq. (91).) $\hat{f}_{qji}$ is the agent's prediction, and f_{qji}^{ref} is a reference, corresponding to an observed outcome or an independently estimated oracle mean, as specified by the benchmark. For a fully observed fixed-dimensional feature vector, this becomes

$$L_{\text{features}} = \frac{1}{d_F} \sum_{i=1}^{d_F} \frac{\sum_q (\hat{f}_{qi} - f_{qi}^{\text{ref}})^2}{Q s_i^2} \tag{8}$$

For trajectory targets, this becomes

$$L_{\text{traj}} = \frac{1}{d_y} \sum_{i=1}^{d_y} \frac{\sum_q \sum_{t>t_{c,q}} o_{qti}(\hat{y}_{qti} - y_{qti})^2}{s_i^2 \sum_q \sum_{t>t_{c,q}} o_{qti}} \tag{9}$$

[6] Our API is loosely based on BOXINGGYM's Fig. 2 (Gandhi et al., 2025). However, we make several changes: (i) we evaluate after every step, to get a learning curve, rather than a single number; (ii) we add an update-state method to each agent, since MDA sequentially updates its belief state.

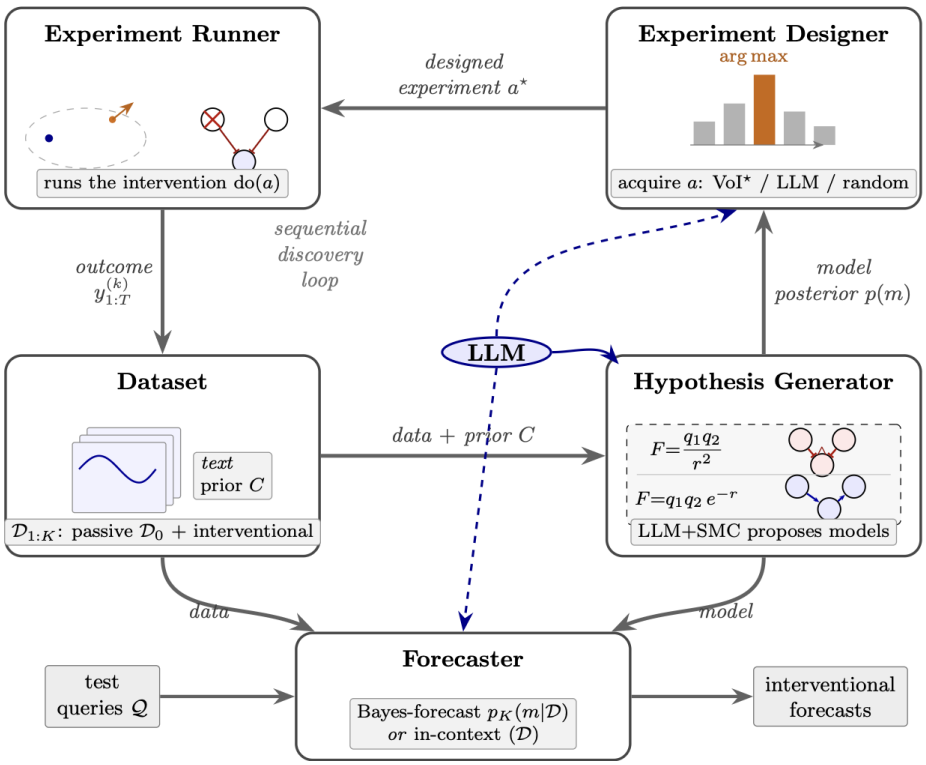


Figure 5: **The MDA discovery loop.** See text for details. (Figure based on (Elteto et al., 2026, Fig.2).)

For univariate fully observed time series of common length T, with $t_c = 0$ and $s_1 = 1$, this simplifies further to the familiar expression

$$L_{\text{traj}} = \frac{1}{QT} \sum_{q=1}^{Q} \sum_{t=1}^{T} (\hat{y}_{qt} - y_{qt})^2 \tag{10}$$

For probabilistic forecasts, we can go beyond evaluating the mean by scoring the full predictive distribution $\hat{P}_q$, which may integrate uncertainty over states, parameters, and model structures. The reference distribution $P_q^\star$ is the oracle's distribution for the same target, design, and observed prefix. CRPS scores a univariate forecast against a held-out realization (or is averaged over independent oracle draws), which we discuss in Section D.2.

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 numerical fitting, inference, and acquisition routines, never by the LLM. See Algorithm 2 for the high level pseudocode; we will describe the individual functions below.

SSM. The agent assumes the unknown environment can be represented by the state-space model defined in Eq. (4) and Fig. 1.[7] An experiment *design* χ is the choice of initial state z_0, the policy π (a fixed input sequence $a_{1:T}$ in our open-loop experiments) and the optional perturbations δ. We use the following notation: $\theta' = \text{do}(\delta, \theta)$ are the parameters of the system after applying intervention δ, z_t are the hidden states, a_t are the optional inputs, and y_t are the observed outputs. For simpler

[7]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.

Algorithm 2 The MDA agent (member functions). Schematic prequential-check variant; domain-specific checks are described in the benchmark appendices. 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). Optional ADAPT-AND-PRUNE selects a smaller cap when confidence and fit pass the domain-specific gates, retains the highest-scoring models, and refits if the pool changes (ChemBench: Section B.2); it is the identity when disabled.

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 11)
  choose LOGLIK $\in$ {LOGLIK-DET, LOGLIK-PF, LOGLIK-BSL}
  choose MARGLIK $\in$ {MARGLIK-SMC, MARGLIK-PF, MARGLIK-BIC}
  store optimizer OPT
  $p(m \mid \mathcal{D}) \leftarrow$ MODEL-POSTERIOR($\mathcal{D}, \mathcal{M}$; N_m, LOGLIK, MARGLIK) ▷ Algorithm 9
end function

function DESIGN(self)
  return OPT. $\arg\max_{\chi \in \mathcal{X}}$ VoI(χ; $p(m \mid \mathcal{D})$) ▷ Algorithm 14
end function

function UPDATE-STATE(self, χ, obs)
  $\mathcal{D}^- \leftarrow \mathcal{D}$
  $e^* \leftarrow$ PREDICTIVE-CHECK(χ, obs, $\mathcal{D}^-, p(m \mid \mathcal{D}^-)$) ▷ pre-update forecast
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\chi, \text{obs})\}$ ▷ absorb datum: $\mathcal{D}$ is now $\mathcal{D}_{0:k}$
  $p(m \mid \mathcal{D}) \leftarrow$ MODEL-POSTERIOR($\mathcal{D}, \mathcal{M}$)
  for $i = 1$ **to** R_m **do** ▷ model-search rounds (ModelSMC-style)
    if $e^* \le \tau_e \ \vee \ n_e \ge N_e^{\max}$ **then break** ▷ stop if adequate (n_e gate)
    $\mathcal{M} \leftarrow$ EXPAND-HYP-SPACE(LLM, $\mathcal{M}, \mathcal{D}, C$; N_{new}) ▷ propose local mutations
    $p(m \mid \mathcal{D}) \leftarrow$ MODEL-POSTERIOR($\mathcal{D}, \mathcal{M}$) ▷ full-batch re-judge
    n_e += 1
    $e^* \leftarrow$ PREDICTIVE-CHECK(χ, obs, $\mathcal{D}^-, p(m \mid \mathcal{D})$) ▷ refitted, in-sample adequacy
  end for
  $(\mathcal{M}, p(m \mid \mathcal{D}), N_m) \leftarrow$ ADAPT-AND-PRUNE($\mathcal{M}, p(m \mid \mathcal{D}), \mathcal{D}$) ▷ optional pool shrinkage
end function

function GET-MODEL(self)
  return $\hat{m} \leftarrow \arg\max_{m \in \mathcal{M}} \ p(m \mid \mathcal{D})$ ▷ current-data MAP, not maximum over rounds
end function

function GET-PREDICTOR(self, flavour=MAP) ▷ return a predictor $\hat{F}(\cdot)$
  if flavour = MAP **then**
    return $\hat{F}\colon (\chi, \mathcal{H}) \mapsto \mathbb{E}[F(Y) \mid \chi, \mathcal{H}, \hat{m}, \hat{\theta}]$ ▷ plug-in parameters; filter using $\mathcal{H}$
  else
    return $\hat{F}\colon (\chi, \mathcal{H}) \mapsto \mathbb{E}[F(Y) \mid \chi, \mathcal{H}, \mathcal{D}]$ ▷ posterior predictive, Eq. (1)
  end if
end function

models, there might not be any latent variables — equivalent to $z = \text{const}$ — or there might not be any temporal dynamics — equivalent to $T = 1$.

Latent dynamics. For a fixed open-loop input sequence, the distribution over latent paths is given by

$$p(z_{1:T}|\chi, \theta, m) = \prod_{t=1}^{T} p(z_t|z_{t-1}, \chi, \theta', m) \tag{11}$$

Here the formula conditions on z_0 when specified by χ; otherwise integrate over its declared initial-state distribution. The same convention applies to the compact likelihood algorithms below. Query prediction additionally conditions on its observed prefix, with $\mathcal{H} = \varnothing$ when $t_c = 0$. Note that $p(z_t|z_{t-1}, a_t, \theta)$ may be an implicit distribution, that we can sample from but cannot evaluate pointwise. For some cases, the latent dynamics are deterministic, and are given by an ODE:

$$p(z_t \mid \chi, \theta, m, z_0) = \delta(z_t - z_t(\chi, m, \theta)) \tag{12}$$

$$z_t(\chi, m, \theta) = \text{ODE-solve}(t, m, \theta', z_0, a_{1:t}) \tag{13}$$

where θ' are the (optionally perturbed) parameters, z_0 is the initial condition specified by χ, and $a_{1:T}$ are the inputs specified by its open-loop policy.

Observation model. We usually assume the emission distribution is

$$p(y_t|z_t, m, \theta) = \mathcal{N}(y_t|f_o(z_t), \sigma^2) \tag{14}$$

where f_o is the observation model. For scalar latent state space, this is often the identity function; for multivariate latent state space (e.g., as in HH), f_o often is a linear projection, which selects the relevant observable components.

Three-level inference. 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

$$p(m \mid \mathcal{D}, \mathcal{M}) = \frac{p(\mathcal{D} \mid m)p(m)}{p(\mathcal{D})} = \frac{p(\mathcal{D} \mid m)p(m)}{\sum_{m' \in \mathcal{M}} p(\mathcal{D} \mid m')p(m')} \tag{15}$$

This is computed by the MODEL-POSTERIOR function in Algorithm 9. When we add the hypothesis space expansion procedure of Algorithm 12, we get a method that is similar to the SMC-S procedure discussed in Section A.8.

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

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

By default, this is computed using tempered SMC in Algorithm 6. See Section A.6 for details.

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

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

By default, this is computed by bootstrap particle filtering (a special case of SMC) in Algorithm 4. See Section A.5 for details.

Using PF to estimate $p(\mathcal{D} \mid m, \theta)$ inside of an outer SMC algorithm to estimate $p(\theta \mid \mathcal{D}, m)$ follows the nested architecture of SMC^2 (Chopin et al., 2013). Our outer level enumerates and weights candidate models rather than running another SMC sampler. The three-level hierarchy thus combines particle inference within models with approximate finite-library selection; LLM proposals and pruning search for useful structures.

Simpler inference backends. 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. For the finite candidate pool, model weights are computed by enumeration; there is no additional structure-level SMC layer. We also consider other approximations, such as replacing particle filtering for computing the likelihood with Bayesian synthetic likelihood (see Section A.5.2), and replacing tempered SMC for computing the marginal likelihood with a BIC approximation (see Section A.6.3).

From beliefs to actions. Given a posterior $p(m, \theta|\mathcal{D})$, we can choose the next experiment χ by maximizing the value of information provided by χ, as explained in Section A.9.

Arm	Likelihood / parameter inference	Model scoring / prediction	Model library / experiment design
Chemistry MDA	Weighted residual fit / bounded multistart optimization	BIC weights / MAP plug-in prediction	9 canonical models plus up to 12 initial proposals, adaptive cap 6–24, up to 6 proposals per expansion / CMA-ES over 7d space
Glucose MDA	ODE observation likelihood with AR(1) CGM noise / bounded optimization	BIC weights / model-averaged plug-in prediction	4 initial models, up to 2 proposals per expansion, cap 14 / enumeration over 48 nominal templates (42 distinct)
HH SDE+PF	Bootstrap PF, $N_z = 128$ / SMC tempering with $N_p = 64$, then recursive updates	Estimated evidence / BMA prediction	3 model proposals plus plain Na/K/leak, online expansion / enumeration over 90 designs
HH ODE+BIC	Deterministic Gaussian likelihood / bounded optimization	BIC weights / model-averaged deterministic predictions at fitted parameters	Same initial archive / enumeration over 90 designs

Table 2: **Main experimental settings.** We specify the key parameter/algorithm configurations for the experiments in the main text. All LLMs use GPT-5.6-Luna, medium effort (except in Section E). The two ICL baselines perform direct prediction without these inference backends. Chemistry BIC, glucose BIC, and HH ODE+BIC use no additional parameter-count prior; HH SDE+PF weights additionally use $p(m) \propto \exp(-2C_m)$. Experiment design maximizes a between-model disagreement surrogate (Eq. (59), with domain-specific scaling) using CMA-ES or enumeration. More specific details can be found in the appendix for each experimental domain.

From beliefs to predictions. Given a posterior $p(m, \theta|\mathcal{D})$, we can predict the outcome of a novel experiment using the posterior mean:

$$\hat{F}_{\text{BMA}}(\chi) = \int \int \sum_m F(Y) p(Y|\chi, m, \theta) p(m, \theta|\mathcal{D}) dY \, d\theta \tag{18}$$

We also consider a simplified prediction, where we use a plugin estimate with the MAP model and its parameters,

$$\hat{F}_{\text{plugin}}(\chi) = \int F(Y) p(Y|\chi, \hat{m}, \hat{\theta}_m) dY \tag{19}$$

where $\hat{\theta}_m$ is the posterior mean or mode of the parameters for model $\hat{m}$.

Experimental settings. Table 2 summarizes the different variants of MDA that we use in our main experiments. Further details can be found in the following appendices.

A.4 Priors

Candidate structures may be seeded or LLM-proposed; each has its own free parameters θ. The joint prior factorises as

$$p(m, \theta) = p(m) \prod_{j=1}^{C_m} p(\theta_j \mid m), \tag{20}$$

with a *uniform* prior on each coefficient over its declared bounds[8], and a *structure* prior

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

[8]When the proposer chooses bounds after seeing data, those bounds define a data-dependent prior. Conditioning on that choice does not remove data reuse: narrowing bounds around a fit can inflate evidence. These are approximate predictive model weights, not calibrated Bayes factors under a prespecified prior.

Algorithm 3 LOGLIK-DET — the exact log-likelihood for *deterministic* latent dynamics (Eq. (24)), summed over the dataset. Each experiment requires one noise-free rollout from its own known initial state; emissions are independent Gaussians.

def LOGLIK-DET$(\mathcal{D}, m, \theta) \to \ell$
$\ell \leftarrow 0$
for each $(\chi_i, y_{i,1:T_i}) \in \mathcal{D}$ **do**
  $z_{i,1:T_i} \leftarrow \mathrm{ODE-solve}(m, \theta, \chi_i)$ ▷ experiment-specific initialization and inputs
  $\ell \leftarrow \ell + \sum_{t=1}^{T_i} \log \mathcal{N}(y_{i,t}; f_o(z_{i,t}), \sigma^2)$
end for
return ℓ

where C_m is the number of free parameters of m and λ is a per-parameter Occam penalty (in nats; the active per-domain settings and BIC exceptions are in Table 2). 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 6. 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 distribution on the visited finite support.

Prior choice and predictive weighting. Uniform priors are convenient, but not noninformative: their bounds and parameterization can substantially affect marginal likelihoods and hence model weights (Llorente et al., 2022). A stronger justification would combine physical parameter scales with prior-predictive checks that simulated responses are plausible across relevant interventions. Sensitivity of model weights need not imply sensitivity of predictions when different mechanisms behave similarly; our empirical claims concern held-out prediction rather than calibrated probabilities of recovering the true mechanism. For this predictive objective, an alternative is to choose ensemble weights using cross-validated or genuinely pre-update predictive scores, rather than marginal likelihoods. Such weights would represent predictive utility, not posterior model probabilities; the refitted residuals used in our adequacy checks are not held-out validation scores. A prospective safeguard is to fix parameter ranges by mechanism and physical units before seeing outcomes, or to evaluate newly proposed bounds only on subsequent data. Finite-bound validation alone does not prevent evidence inflation. We leave a systematic investigation of prior choice, predictive robustness, and predictive weighting to future work.

A.5 Computing the likelihood

All three likelihood routines take $(\mathcal{D}, m, \theta)$ and sum the log-likelihoods over experiments $(\chi_i, y_{i,1:T_i}) \in \mathcal{D}$, assumed conditionally independent given (m, θ) and their designs. Each experiment has its own initial state and input sequence; trajectories are not concatenated. The recursive evidence update passes a singleton dataset for each new experiment.

We assume the complete-data conditional likelihood for a single experiment is given by

$$p(y_{1:T}|z_{1:T}, \chi, m, \theta) = \prod_{t=1}^{T} \mathcal{N}(y_t|f_o(z_t), \sigma^2) \tag{22}$$

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

$$p(y_{1:T}|\chi, m, \theta) = \int p(y_{1:T}|z_{1:T}, \chi, m, \theta) p(z_{1:T}|\chi, m, \theta) dz_{1:T} \tag{23}$$

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

$$p(y_{1:T}|\chi, m, \theta) = \prod_{t=1}^{T} \mathcal{N}(y_t|f_o(z_t(\chi, m, \theta)), \sigma^2) \tag{24}$$

where $z_t(\chi, m, \theta)$ is defined in Eq. (13). See Algorithm 3. By contrast, for the stochastic models, the path integral over $z_{1:T}$ is generally intractable, so we must use approximations, which we discuss below.

Algorithm 4 LOGLIK-PF: sum of independently estimated trajectory log-likelihoods. Each experiment starts a fresh bootstrap filter with N_z particles. The product of the independent PF likelihood estimates is unbiased for the dataset likelihood under the discretized model; its logarithm is not unbiased. Transitions use the experiment's inputs and may require solver substeps (Fox–Lu SDE for HH, Eq. (88)). Cost is $O(N_z \sum_i T_i)$ at fixed solver resolution.

def LOGLIK-PF$(\mathcal{D}, m, \theta) \to \hat{\ell}$
$\hat{\ell} \leftarrow 0$
for each $(\chi_i, y_{i,1:T_i}) \in \mathcal{D}$ **do**
  $z^j_{i,0} \leftarrow z_0(\chi_i)$ for $j = 1, \dots, N_z$ ▷ reset for this experiment
  for $t = 1, \dots, T_i$ **do**
    $z^j_{i,t} \sim p(z_t \mid z^j_{i,t-1}, \chi_i, m, \theta)$ for every j
    $w^j_{i,t} \leftarrow \mathcal{N}(y_{i,t}; f_o(z^j_{i,t}), \sigma^2)$ for every j
    $\hat{\ell} \leftarrow \hat{\ell} + \log\left(N_z^{-1} \sum_j w^j_{i,t}\right)$ ▷ evaluate in log space
    systematically resample $\{z^j_{i,t}\}$ with probabilities proportional to $\{w^j_{i,t}\}$
  end for
end for
return $\hat{\ell}$

Algorithm 5 LOGLIK-BSL: Gaussian synthetic log-likelihood of independent experiment summaries, not of raw trajectories. The summary map, simulation count R, shrinkage λ, and ridge $\varepsilon > 0$ are fixed across candidate models.

def LOGLIK-BSL$(\mathcal{D}, m, \theta) \to \ell$
$\ell \leftarrow 0$
for each $(\chi, y) \in \mathcal{D}$ **do**
  simulate independent $(z^{(r)}, y^{(r)}) \sim p(z, y \mid \chi, m, \theta)$ for $r = 1, \dots, R$
  $s_r \leftarrow s(y^{(r)})$; $\hat{\mu} \leftarrow R^{-1} \sum_r s_r$
  $S \leftarrow (R-1)^{-1} \sum_r (s_r - \hat{\mu})(s_r - \hat{\mu})^\top$
  $\hat{\Sigma} \leftarrow (1-\lambda) S + \lambda \operatorname{diag}(S) + \varepsilon I$
  $\ell \leftarrow \ell + \log \mathcal{N}(s(y); \hat{\mu}, \hat{\Sigma})$
end for
return ℓ

A.5.1 PARTICLE FILTERING

Our default method is to use the bootstrap particle filter algorithm shown in Algorithm 4. It propagates weighted filtering particles for $p(z_t \mid y_{1:t}, m, \theta)$; sampling posterior paths additionally requires retained ancestry or smoothing. The displayed routine returns an unbiased approximation of the likelihood, given by

$$\hat{p}(y_{1:T} \mid m, \theta) = \prod_t \left(\tfrac{1}{N_z} \sum_{j=1}^{N_z} w_t^{(j)}\right) \tag{25}$$

where w_t^j is the weight of particle j. For details, see e.g. (Chopin & Papaspiliopoulos, 2020; Naesseth et al., 2019).

A.5.2 BAYESIAN SYNTHETIC LIKELIHOOD

An alternative to PF is to replace the likelihood $p(y_{1:T}|m, \theta)$ with an approximation

$$p_s(s(y_{1:T})|m, \theta) \approx \mathcal{N}(s(y_{1:T})|\mu_{m,\theta}, \Sigma_{m,\theta}) \tag{26}$$

where $s(y_{1:T})$ is some summary feature vector derived from the trajectory, and μ and Σ are computed as described below. This approximates the density of the summary, not the raw-data density. This is useful when a per-time step observation model, $p(y_t|z_t, m, \theta)$, is not available (since this is a requirement of PF). In addition, using the summary features can provide robustness to model misspecification.

To estimate μ and Σ, we can use the "Bayesian Synthetic Likelihood" method of (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 then estimate the likelihood using

$$\mu_{m,\theta} = \frac{1}{R} \sum_r s(y^{(r)}) \tag{27}$$

$$\Sigma_{m,\theta} = \widehat{\mathrm{Cov}}_r\big[s(y^{(r)})\big] + \varepsilon I, \tag{28}$$

with a small ridge ε for conditioning. See Algorithm 5 for the pseudocode. We can also replace the Gaussian with a normalizing flow model, a technique called *neural likelihood estimation* (Papamakarios et al., 2019).[9]

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 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:

$$(\hat{m}, \hat{\theta}) = (f^m_w(s_\phi(y_{1:T})), f^\theta_w(s_\phi(y_{1:T}))) \tag{29}$$

We then pass $s_\phi(y_{1:T})$, corresponding to the final layer of the network, into the above BSL method to compute the likelihood, which is passed to SMC to compute the evidence Z_m.

There are two problems with BSL. (1) It is not much faster than PF (see e.g., Fig. 19), since it still requires rolling out R stochastic trajectories for every candidate (m, θ). (2) The above approach to learning $s_\phi(y)$ with a neural network requires predicting m and θ on the output, which is challenging if θ has variable size and/or m is a complex structure (such as a graph or program) and not just a discrete index. We can avoid both problems by using *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*

$$r_\phi(y, m, \theta) \;=\; \frac{d_\phi}{1 - d_\phi} \;\approx\; \frac{p(y_{1:T} \mid m, \theta)}{p(y_{1:T})}, \tag{30}$$

Once trained, we can use $r_\phi(y, m, \theta)$ in lieu of $p(y|m, \theta)$ inside of tempered SMC, without needing rollouts at run time, solving problem (1). Also, this is a discriminative model that conditions on m and θ and emits a scalar, avoiding variable-dimensional outputs. It still requires an encoding of (m, θ) and simulation training coverage; generalization to newly proposed structures is not automatic. We leave exploration of this approach to future work.

A.6 Computing the marginal likelihood

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

$$Z^k_m = p(\mathcal{D}_{0:k} \mid m) = \int p(\mathcal{D}_0 \mid m, \theta) \left[\prod_{j=1}^{k} p(y_{j,1:T} \mid \chi_j, m, \theta) \right] p(\theta \mid m)\, d\theta \tag{31}$$

where $p(y \mid m, \theta)$ is the observed data likelihood computed above.

A.6.1 Tempered SMC

Our default method to compute Z^k_m is to use tempered SMC with an adaptive tempering schedule, as shown in Algorithm 6. This also returns the posterior over the parameters for each model:

$$p(\theta|m, \mathcal{D}_{0:k}) \approx \sum_{i=1}^{N_p} W^k_i \delta(\theta - \theta_i) \tag{32}$$

[9] 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). See (Cranmer et al., 2020; Frazier et al., 2024) for more discussion.

Algorithm 6 Adaptive-tempering SMC over N_p parameter particles for each model m. Returns particles, weights, and $\log Z_m$. The effective sample size is given by $\mathrm{ESS} = (\sum W_i)^2 / \sum W_i^2$; 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 (LLM-free, but potentially expensive). ℓ_i is a *log*-likelihood; $\textsc{loglik}(\mathcal{D}, m, \theta)$ 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 for a symmetric proposal, with proposals outside the support rejected. The NumPy fitter instead clips proposals at the bounds, so it violates detailed balance for this target and can bias posterior and evidence estimates; its legacy results should not be read as a test of exact Bayesian inference. Its diagonal proposal scale is half the current per-coordinate particle standard deviation (floored at 10^{-3} before scaling).

def MARGLIK-SMC$(\mathcal{D}, m;$ LOGLIK$) \to (\{\theta_i\}, \{W_i\}, \log Z_m)$
sample $\theta_i \sim p(\cdot \mid m)$ for $i = 1..N_p$; $W_i \leftarrow 1/N_p$
$\beta \leftarrow 0$; $\log Z_m \leftarrow 0$
$\ell_i \leftarrow \textsc{loglik}(\mathcal{D}, m, \theta_i)$
while $\beta < 1$ **do** ▷ $\le J_p$ annealing rungs
  take $\Delta\beta = 1 - \beta$ if its ESS meets the target or the rung cap is reached; otherwise bisect to target ESS ηN_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 $s = 1 \dots R_p$ **do** ▷ RW-Metropolis at temperature β
    propose $\theta_i' \sim q(\cdot \mid \theta_i)$; reject immediately if outside the prior bounds
    $\ell_i' \leftarrow \textsc{loglik}(\mathcal{D}, m, \theta_i')$
    $(\theta_i, \ell_i) \leftarrow (\theta_i', \ell_i')$ w.p. $\min\{1, \, e^{\beta(\ell_i' - \ell_i)}\}$
  end for
end while
return $(\{\theta_i\}, \{W_i\}, \log Z_m)$

Refitting on all accumulated data at every experiment makes the total cost $O(K^2 T N_p)$ per model, holding the tempering and rejuvenation budgets fixed and assuming an $O(T)$ trajectory likelihood. With PF likelihoods, this additionally scales with N_z (and the number of solver substeps per observation). The recursive alternative below avoids repeatedly processing old trajectories.

With noisy PF likelihoods, generally $\mathbb{E}[\widehat{L}^\beta] \neq L^\beta$ for $0 < \beta < 1$. Unbiased PF estimates alone therefore do not justify the intermediate tempered targets; exact pseudo-marginal inference requires an appropriate extended-state construction (Andrieu & Roberts, 2009). We treat the implemented particle evidence as a numerical approximation.

A.6.2 Recursive parameter and evidence updates

For many experiments or long trajectories, full-history refitting is expensive. For a fixed model m, define the new-experiment likelihood $L_{m,k}(\theta) = p(y_{k,1:T} \mid \chi_k, m, \theta)$. Conditioning on the chosen designs and assuming experiments are independent given (m, θ), Bayes' rule gives the exact recursion

$$c_{m,k} = \int L_{m,k}(\theta) \, p(\theta \mid m, \mathcal{D}_{0:k-1}) \, d\theta, \qquad Z_m^k = Z_m^{k-1} c_{m,k}, \tag{33}$$

$$p(\theta \mid m, \mathcal{D}_{0:k}) = \frac{L_{m,k}(\theta) \, p(\theta \mid m, \mathcal{D}_{0:k-1})}{c_{m,k}}. \tag{34}$$

Algorithm 7 MARGLIK-PF: particle filtering over static parameters across experiments (Eqs. (35) and (36)). This outer recursion is distinct from the latent-state filter LOGLIK-PF; any chosen trajectory likelihood can supply its increments. Cached state is maintained separately for each model.

def MARGLIK-PF($\mathcal{D}_{0:k}, m$; LOGLIK) $\to (\{\theta_i\}, \{W_i\}, \log \hat{Z}_m)$
if no valid cached state exists **then return** MARGLIK-SMC($\mathcal{D}_{0:k}, m$; LOGLIK) and cache its result
for each experiment (χ_j, y_j) not yet assimilated, in order **do**
  $a_i \leftarrow \log W_i + \text{LOGLIK}(\{(\chi_j, y_j)\}, m, \theta_i)$ for every i
  $c \leftarrow \text{logsumexp}_i(a_i)$
  $\log \hat{Z}_m \leftarrow \log \hat{Z}_m + c$
  $W_i \leftarrow \exp(a_i - c)$
  if $1/\sum_i W_i^2 < N_p/2$ **then**
    systematically resample $\{\theta_i\}$ with probabilities $\{W_i\}$; $W_i \leftarrow 1/N_p$
  end if
end for
cache particles, weights, evidence, and assimilated experiment IDs
return $(\{\theta_i\}, \{W_i\}, \log \hat{Z}_m)$

Initialize $\widehat{Z}_m^0$ and the parameter particles by tempered SMC on $\mathcal{D}_0$. Given normalized retained weights $W_{m,i}^{k-1}$, evaluate only the new trajectory at each retained parameter value:

$$\widehat{c}_{m,k} = \sum_{i=1}^{N_p} W_{m,i}^{k-1} \widehat{L}_{m,k}(\theta_{m,i}^{k-1}), \qquad \log \widehat{Z}_m^k = \log \widehat{Z}_m^{k-1} + \log \widehat{c}_{m,k}, \tag{35}$$

$$\widetilde{W}_{m,i}^k = \frac{W_{m,i}^{k-1} \widehat{L}_{m,k}(\theta_{m,i}^{k-1})}{\widehat{c}_{m,k}}. \tag{36}$$

Here $\widehat{L}$ is the exact likelihood when tractable, or its PF estimate for stochastic dynamics; the implementation computes the sums in log space. The model weights are then proportional to $p(m)\widehat{Z}_m^k$ over the current archive. If $1/\sum_i (\widetilde{W}_{m,i}^k)^2$ falls below $N_p/2$, we systematically resample parameter particles and reset their weights to $1/N_p$, without changing the accumulated evidence. Otherwise we retain the weighted particles. The HH SDE+PF experiment in Section D.3 uses this procedure, with $N_p = 64$ and $N_z = 128$, and no subsequent parameter rejuvenation.

Cost and limitations. Excluding initialization, for fixed particle counts and no full-history rejuvenation, the recursive cost per model is $O(N_p \sum_{k=1}^K C(T_k))$, where $C(T)$ is the cost of one trajectory likelihood. For equal lengths and a tractable $O(T)$ likelihood, this is $O(KTN_p)$ rather than $O(K^2TN_p)$; suppressing likelihood cost gives the $O(KN_p)$ versus $O(K^2N_p)$ comparison. For PF, $C(T) = O(TN_z)$ at fixed solver resolution. Thus recursion avoids revisiting old experiments but does not eliminate the cost of processing a long new trajectory. For a single growing time series, analogous within-trajectory updates require retaining each parameter particle's latent filtering state; the implemented HH outer update instead assimilates whole, independently initialized experiments.

Repeated resampling can collapse parameter diversity: it duplicates existing values rather than discovering new ones. The recursion is therefore a finite-particle approximation, not a guarantee of accurate long-run evidence. Weight ESS after resampling does not measure distinct parameter support; particle impoverishment can understate parameter uncertainty and distort both prediction intervals and acquisition scores. (See (Chopin, 2002; Chopin et al., 2013) for details.) Posterior-invariant MCMC rejuvenation or occasional batch refits can help, but may require evaluating the full history and invalidate the simple linear-cost bound. A newly proposed structure has no inherited parameter posterior or evidence and must be fitted on all accumulated data; the current HH implementation refits when the archive changes.

A.6.3 BIC APPROXIMATION

As a faster alternative to tempered SMC, we use the Bayesian Information Criterion (BIC), which we now explain.

Algorithm 8 MARGLIK-BIC: optimization and BIC evidence approximation. The Gaussian profiled-scale branch matches Eq. (45); the direct-likelihood branch covers PF/BSL optimization. N is the declared observation count, not the number of Monte Carlo particles. Comparisons require a common observation space and likelihood convention.

def MARGLIK-BIC$(\mathcal{D}, m;$ LOGLIK$) \to (\{\hat{\theta}_m\}, \{1\}, \log \hat{Z}_m)$
if using Gaussian residuals with a profiled common variance **then**
  $\hat{\theta}_m \leftarrow \arg\min_{\theta \in \Theta_m} \mathrm{RSS}_m(\theta)$ ▷ bounded multi-start optimization
  $b_m \leftarrow N \log(\mathrm{RSS}_m(\hat{\theta}_m)/N) + C_m \log N$
else
  $\hat{\theta}_m \leftarrow \arg\max_{\theta \in \Theta_m}$ LOGLIK$(\mathcal{D}, m, \theta)$ ▷ declared simulation budget for noisy likelihoods
  $b_m \leftarrow -2\,$LOGLIK$(\mathcal{D}, m, \hat{\theta}_m) + C_m \log N$
end if
return $(\{\hat{\theta}_m\}, \{1\}, -b_m/2)$ ▷ log evidence up to a common constant

Write $\ell_m(\theta) = \log p(\mathcal{D} \mid m, \theta)$ and $C_m = \dim(\theta_m)$. For an interior maximum and a smooth prior, the large-sample Laplace approximation is

$$\log Z_m \approx \ell_m(\hat{\theta}_m) + \log p(\hat{\theta}_m \mid m) + \frac{C_m}{2} \log(2\pi) - \frac{1}{2} \log |H_m|, \tag{37}$$

where $H_m = -\nabla^2 \ell_m(\hat{\theta}_m)$. Under regularity conditions, H_m grows proportionally to N, so retaining the leading terms gives

$$-2 \log Z_m \approx \mathrm{BIC}_m = -2\ell_m(\hat{\theta}_m) + C_m \log N. \tag{38}$$

BIC drops bounded, generally model-dependent prior-density and curvature terms. We normalize $p(m) \exp(-\mathrm{BIC}_m/2)$ over the candidate pool. Below we discuss how to compute BIC, with different noise assumptions. See Algorithm 8 for the pseudocode.

Fixed residual scale ($\tau = 1$). For N scalar observations, assume fixed positive whitening scales σ_n shared across candidates, and define

$$\mathrm{RSS}_m(\theta) = \sum_{n=1}^{N} \left(\frac{y_n - \hat{y}_n(\theta)}{\sigma_n} \right)^2. \tag{39}$$

With independent Gaussian errors and $\hat{\theta}_m$ minimizing this RSS,

$$-2 \log p(\mathcal{D} \mid m, \hat{\theta}_m) = \mathrm{RSS}_m + \sum_{n=1}^{N} \log(2\pi\sigma_n^2). \tag{40}$$

For a common $\sigma_n = \sigma$, the last term is $N \log(2\pi\sigma^2)$. This is the negative twice-log-likelihood, not yet BIC: adding the parameter penalty and dropping the shared normalization gives

$$\mathrm{BIC}_m^{\mathrm{fixed}} = \mathrm{RSS}_m + C_m \log N. \tag{41}$$

Here $\mathrm{RSS}_m = \mathrm{RSS}_m(\hat{\theta}_m)$ is already noise-whitened.

Fitted (profiled) residual scale. Alternatively, retain the same fixed whitening scales σ_n, shared across candidate models, and an unknown common residual-scale multiplier τ, giving the log likelihood

$$y_n \mid m, \theta_m, \tau \sim \mathcal{N}\big(\hat{y}_n(\theta_m), \tau^2 \sigma_n^2\big), \tag{42}$$

$$-2 \log p(\mathcal{D} \mid m, \theta_m, \tau) = \frac{\mathrm{RSS}_m(\theta_m)}{\tau^2} + N \log \tau^2 + \sum_{n=1}^{N} \log(2\pi\sigma_n^2). \tag{43}$$

The maximum-likelihood estimate $\hat{\theta}_m$ minimizes this weighted RSS. Writing $\mathrm{RSS}_m = \mathrm{RSS}_m(\hat{\theta}_m)$, maximizing over τ gives $\hat{\tau}_m^2 = \mathrm{RSS}_m/N$. Substitution yields

$$-2 \log p(\mathcal{D} \mid m, \hat{\theta}_m, \hat{\tau}_m) = N \log(\mathrm{RSS}_m/N) + N + \sum_{n=1}^{N} \log(2\pi\sigma_n^2). \tag{44}$$

Algorithm 9 MODEL-POSTERIOR: normalize estimated log evidences plus log priors over the current pool (no LLM). Each model uses the selected evidence backend, including its cached state for recursive updates. The routine updates the stored pool and parameter states in place and returns approximate, pool-conditional model weights.

def MODEL-POSTERIOR($\mathcal{D}$, $\mathcal{M}$; N_m, LOGLIK, MARGLIK) $\to p(m \mid \mathcal{D})$
for $m_i \in \mathcal{M}$ **do**
  $(\{\theta_{ij}\}, \{W_{ij}\}, \log \hat{Z}_i) \leftarrow$ MARGLIK($\mathcal{D}$, m_i; LOGLIK)
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
renormalize $p(m \mid \mathcal{D})$ on the retained pool
store $\hat{p}(\mathcal{D}) = \sum_{m_i \in \mathcal{M}} \pi(m_i)\hat{Z}_i$ ▷ finite-pool normalizer only
return $p(m \mid \mathcal{D})$

The full BIC adds $(C_m + 1)\log N$, since the fitted scale τ is one additional parameter. The terms N and $\sum_n \log(2\pi\sigma_n^2)$ and this additional $\log N$ are identical for every candidate on the same dataset, so we drop them. This gives the comparison score

$$\mathrm{BIC}_m := N \log(\mathrm{RSS}_m/N) + C_m \log N. \tag{45}$$

where $C_m = \dim(\theta_m)$ is the number of fitted model parameters (excluding the common fitted residual scale). Lower is better. We use $\log \widehat{Z}_m = -\mathrm{BIC}_m/2$ as an approximate evidence score, and normalize $p(m)\exp(-\mathrm{BIC}_m/2)$ over the pool.

Noise terms σ_n. The term σ_n is a heteroscedastic whitening scale:

$$\sigma_n = \max(\rho|y_n|, \sigma_{\mathrm{floor}}) \tag{46}$$

In CHEMBENCH we use $\rho = 0.03$ and $\sigma_{\mathrm{floor}} = 0.01$. For the stochastic HH experiment, the deterministic ODE+BIC control instead uses a constant $\sigma_n = 2\,\mathrm{mV}$ (equivalently, $\rho = 0$ and $\sigma_{\mathrm{floor}} = 2\,\mathrm{mV}$), matching the declared voltage observation noise. Each trajectory contributes 20 voltage observations, so $N = 20(N_0 + k)$ after k active experiments and N_0 initial trajectories.

Comparison of BIC estimators. In the end-to-end CHEMBENCH control in Fig. 7, fixed-scale BIC gives better early-budget predictions than profiled-scale BIC under CMA-EIG, although their performance is similar at 15 observations.

A.7 PRIOR SENSITIVITY

The model evidence averages the likelihood over the parameter prior, rather than only rewarding its maximum. A diffuse prior can therefore strongly penalize a model even when its fitted predictions barely change (Llorente et al., 2022). For example, enlarging a uniform prior's volume by a factor c, while leaving the likelihood integral essentially unchanged, reduces $\log Z_m$ by $\log c$; an improper prior leaves ordinary Bayes factors undefined.

In Eq. (37), BIC retains the leading terms $\ell_m(\hat{\theta}_m) - \frac{1}{2}C_m \log N$, dropping prior-density and curvature terms that can matter at small N. It avoids explicit prior-volume normalization, but fitting bounds still matter, and unidentified parameters violate its regularity assumptions. Thus a predictive advantage for BIC need not mean that it estimates evidence more accurately. Figure 7 compares predictive performance of complete inference configurations; it does not isolate evidence-estimation accuracy.

A.8 COMPUTING THE MODEL POSTERIOR

The main inference procedure over models, given the current hypothesis class $\mathcal{M}$, is shown in Algorithm 9. 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 10 ADAPT-AND-PRUNE: optional score-ranked pool shrinkage. The fit statistic and thresholds are domain-specific; ChemBench uses maximum weight > 0.9 and median relative fit residual ≤ 0.05 to select caps 6 versus 24 (Section B.2). This does not merge predictively similar models or restore previously discarded models.

def ADAPT-AND-PRUNE$(\mathcal{M}, p, \mathcal{D}) \to (\mathcal{M}, p, N_m)$
if adaptation disabled **then return** $(\mathcal{M}, p, N_m)$
$m^* \leftarrow \arg\max_m p(m)$
$c \leftarrow N_{\max}$
if $p(m^*) > \tau_c$ and fit error of m^* on $\mathcal{D}$ is $\leq \tau_{\text{fit}}$ **then** $c \leftarrow N_{\min}$
if $|\mathcal{M}| > c$ **then**
    retain the c models with largest $p(m)$ in $\mathcal{M}$
    invalidate inference caches if a full refit is required by the backend
    $p \leftarrow$ MODEL-POSTERIOR$(\mathcal{D}, \mathcal{M}; c, \text{LOGLIK}, \text{MARGLIK})$
end if
return $(\mathcal{M}, p, c)$

$\mathcal{M}$-open extension. When the current best (MAP) model's residual (error) e^* exceeds a threshold τ_e (Algorithm 13), we invoke EXPAND-HYP-SPACE (Algorithm 12): the LLM proposes a batch of N_{new} new structures from the observations and available fit diagnostics. ChemBench provides residuals for the current pool, inspired by the pool-conditioned proposals of SMC-S (Piriyakulkij et al., 2024); glucose supplies observed trajectories and candidate fit diagnostics. The new structures are scored by evidence and the pool is evidence-pruned in the following MODEL-POSTERIOR call. If $e^* \leq \tau_e$ the pool is left unchanged and MODEL-POSTERIOR merely re-scores the existing structures on the new datum — no LLM call.

The predictive check. The predictive check measures how well the current model explains observations; a large discrepancy triggers $\mathcal{M}$-open expansion. Algorithm 13 gives a prequential variant: score the new observation before fitting it, optionally combining this with residuals on stored data. The executed ChemBench and DP agents instead gate their first expansion on residuals cached from the preceding fit. They then refit on all available data and, if expanding again, recheck that fit. A newly surprising observation can therefore affect expansion one update later. Neither version retains a running archive of historical forecasts. Relative residuals balance component scales, and taking their median limits the influence of outliers, but can also delay expansion after a single informative failure. More precisely, let the relative error on experiment j be the median relative error over target components i:

$$r_j(m, \theta) = \underset{i}{\text{median}} \frac{|F_i(y_j) - \mathbb{E}[F_i(Y) \mid \chi_j, m, \theta]|}{\max(|F_i(y_j)|, c_i)} \tag{47}$$

Then the estimated error at step k is

$$e^* = \text{median}\left(\{r_k(m^*, \hat{\theta}^{m^*})\} \cup \{r_j(m^*, \hat{\theta}^{m^*}) : j < k\}_{\texttt{backtest}}\right), \tag{48}$$

where the first set is the prequential error on the new observation and the optional second set contains the back-test errors. Here m^* and $\hat{\theta}^{m^*}$ are fixed at their pre-observation values when the new datum is scored. Expansion is triggered when $e^* > \tau_e$.

Finite-pool interpretation and limitations. The outer (model inference) loop, at step k, is approximating the following target distribution

$$p_k(m) \;\propto\; p(m)\, Z_m^{(k)} \tag{49}$$

On a deduplicated finite pool $\mathcal{M}$, exact evidences would give the restricted posterior $w_i \propto p(m_i) Z_{m_i}^{(k)}$. In practice the evidences, or their BIC substitutes, are approximate. When enabled, residual-conditioned LLM proposals enlarge this support (Piriyakulkij et al., 2024); top-N_m pruning is deterministic hypothesis search, not a posterior-invariant resampling kernel. Normalizing over enumerated unique candidates does not require treating them as importance samples from the LLM, but it also does not account for unproposed model mass.

Algorithm 11 INIT-HYP-SPACE: domain-configured initialization. ChemBench hybrid mode combines canonical laws with LLM proposals; truth-blind initialization supplies no privileged true model. Validation checks executability and parameter bounds, not held-out performance.

def INIT-HYP-SPACE(LLM, $C, \mathcal{D}_0) \to \mathcal{M}$
$\mathcal{M} \leftarrow$ configured seed library (possibly empty)
if LLM initialization enabled **then**
  propose a batch of N_{init} models from LLM$(\cdot \mid C, \mathcal{D}_0)$
  validate candidates; merge valid candidates into $\mathcal{M}$, removing exact duplicates
end if
if $\mathcal{M} = \varnothing$ **then** retry within the declared limit or report initialization failure
return $\mathcal{M}$ ▷ scoring and capping follow in MODEL-POSTERIOR

Algorithm 12 EXPAND-HYP-SPACE — grow the pool with a batch of N_{new} LLM-proposed structures. Pool residuals are optional context (used in ChemBench; DP supplies observation summaries). Scoring and pruning are deferred to the next MODEL-POSTERIOR (Algorithm 9); this is the sole LLM call.

def EXPAND-HYP-SPACE(LLM, $\mathcal{M}$, $\mathcal{D}$, C; $N_{\text{new}}) \to \mathcal{M}'$
optionally compute $\rho_j \leftarrow$ RESIDUALS$(m_j, \mathcal{D})$ for $m_j \in \mathcal{M}$ ▷ backend-specific context
propose N_{new} candidates using C, observation summaries, and any available pool diagnostics
return $\mathcal{M} \cup \{m_l'\}$

Algorithm 13 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 $\mathcal{D}_{0:k-1}$ are also re-scored against m^*, and the check returns the median error over the combined set. The per-datum residual $\tilde{\ell}(a,b) = \text{median}_i\, |a_i - b_i| / \max(|a_i|, c_i)$ is per-component *relative* error, where $a = F(y)$ is the observed target and $b = \hat{F}(\chi)$ its forecast, with a small floor of c_i 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 $t = 1 : T$. The first call uses the pre-observation fit; subsequent calls following expansion use a refitted model and are in-sample adequacy checks, not new prequential evidence. The HH stochastic gate uses a different statistic.

def PREDICTIVE-CHECK$\big(\chi,\ y,\ \mathcal{D}_{0:k-1},\ p(m \mid \mathcal{D}),\ \text{BACKTEST} = T\big) \to e$
$m^* \leftarrow \arg\max_m p(m \mid \mathcal{D})$ ▷ MAP under the supplied fit
$\hat{F}(\chi) \leftarrow \mathbb{E}[F(Y) \mid \chi,\ m^*, \hat{\theta}^{m^*}]$ ▷ MAP model's target forecast
$E \leftarrow \big\{\, \tilde{\ell}\big(F(y),\ \hat{F}(\chi)\big) \,\big\}$ ▷ prequential only if the supplied fit excludes y
if BACKTEST **then**
  $E \leftarrow E \cup \big\{\, \tilde{\ell}\big(F(y_j),\ \hat{F}(\chi_j)\big) \ :\ (\chi_j, y_j) \in \mathcal{D}_{0:k-1} \,\big\}$ ▷ in-sample residuals
end if
return median(E)

A.9 ALGORITHMS FOR EXPERIMENT DESIGN

VoI API. The design step maximises a VOI score over the design space. Algorithm 14 defines the interface, which reads off the pooled two-level posterior $p(m \mid \mathcal{D})\, W_i^m$. The objectives only differ in *which* uncertainty they reduce — the model index (model-discrimination EIG, the default), or the whole latent (m, θ) (joint EIG). 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 χ is the expected gain in attainable utility from observing its outcome *before* deciding,

$$\text{VoI}_U(\chi) = \mathbb{E}_{y \sim p(\cdot \mid \chi, \mathcal{D})} \Big[\max_{d \in \Delta} \mathbb{E}_{p(h \mid \mathcal{D}, y, \chi)}\, U(d,h) \Big] \ - \ \max_{d \in \Delta} \mathbb{E}_{p(h \mid \mathcal{D})}\, U(d,h), \tag{50}$$

Algorithm 14 VoI — the experiment-design score at χ under the two-level posterior, with a FLAVOUR selector. All flavours use only the per-particle predictions $\mu_i(\chi) = \mathbb{E}[Y_\chi \mid m_i, \theta_i, \chi]$ (weights W_i^m within structure m), noise-whitened by the observation covariance Σ_ε. MODEL is Lindley's model-disagreement surrogate (default); JOINT keeps within-structure parameter uncertainty;

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

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

return
$$\begin{cases} \sum_m p(m \mid \mathcal{D}) \left\|\bar{\mu}_m(\chi) - \bar{\mu}(\chi)\right\|^2_{\Sigma_\varepsilon^{-1}} & \text{MODEL (Eq. (59))} \\ \sum_{m,i} p(m \mid \mathcal{D})\, W_i^m \left\|\mu_i(\chi) - \bar{\mu}(\chi)\right\|^2_{\Sigma_\varepsilon^{-1}} & \text{JOINT (Eq. (63))} \end{cases}$$

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, $q^\star = p(M \mid \mathcal{D}, y, \chi)$, and $\max_q \mathbb{E}_{p(M|\cdot)} \log q(M) = -H[p(M \mid \cdot)]$, so Eq. (50) becomes

$$\mathrm{VoI}_{\log}(\chi) = \Big(-\mathbb{E}_y\, H[p(M \mid \mathcal{D}, y, \chi)]\Big) - \Big(-H[p(M \mid \mathcal{D})]\Big) = I(M; Y_\chi \mid \mathcal{D}) = \mathrm{EIG}(\chi). \quad (51)$$

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. (51) in its *outcome-entropy* form, $I(M; Y_\chi \mid \mathcal{D}) = H[Y_\chi \mid \mathcal{D}] - H[Y_\chi \mid M, \mathcal{D}]$, in terms of each structure's *posterior predictive* of the outcome,

$$P_m(\cdot) \;:=\; p\big(Y_\chi \mid m, \mathcal{D}, \chi\big) \;=\; \int p\big(Y_\chi \mid m, \theta, \chi\big)\, p(\theta \mid m, \mathcal{D})\, d\theta, \quad (52)$$

i.e. the distribution of the outcome we would observe under design χ 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 $p(Y_\chi \mid \mathcal{D}) = \sum_m p(m)\, P_m$, so $H[Y_\chi \mid \mathcal{D}] = H\big(\sum_m p(m) P_m\big)$; and conditioning on $M{=}m$ makes $Y_\chi \sim P_m$, so $H[Y_\chi \mid M, \mathcal{D}] = \sum_m p(m)\, H(P_m)$. Hence

$$\mathrm{EIG}(\chi) \;=\; I(M; Y_\chi \mid \mathcal{D}) \;=\; H\Big(\sum_m p(m)\, P_m\Big) \;-\; \sum_m p(m)\, H(P_m). \quad (53)$$

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 $I(M; Y_\chi \mid \mathcal{D}) = \sum_m p(m)\ \mathrm{KL}\big(P_m \,\big\|\, \sum_{m'} p(m') P_{m'}\big)$, the mean KL from each structure's predictive to the mixture (a generalised Jensen–Shannon divergence), which is manifestly maximised by designs χ 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 $Y_\chi \in \mathbb{R}^d$ 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 $Y_\chi^m = \bar{\mu}_m(\chi) + \varepsilon$, where

$$\bar{\mu}_m(\chi) = \mathbb{E}_{\theta|m,\mathcal{D}}[\mathbb{E}[Y_\chi \mid m, \theta, \chi]] \quad (54)$$

and $\varepsilon \sim \mathcal{N}(0, \Sigma_\varepsilon)$ is an assumed common within-model noise covariance ($\sigma^2 I$ in the simplest case). Parameter and process uncertainty generally make the actual within-model predictive covariance model-dependent; the following bound applies to this common-covariance surrogate model, not automatically to the original mixture. The unconditional distribution is thus the Gaussian mixture

$$Y_\chi \sim \sum_m p(m \mid \mathcal{D})\, \mathcal{N}(\bar{\mu}_m(\chi), \Sigma_\varepsilon) \quad (55)$$

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

$$I(M; Y_\chi \mid \mathcal{D}) = H[Y_\chi] - H[Y_\chi \mid M] \approx \tfrac{1}{2} \ln \det\Big(I + \Sigma_\varepsilon^{-1} \Sigma_\mu(\chi)\Big), \tag{56}$$

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

$$I(M; Y_\chi \mid \mathcal{D}) \approx \tfrac{1}{2} \ln(1 + \mathrm{Var}[\mu(\chi)]/\sigma^2) \tag{57}$$

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 $\Sigma_\mu(\chi)$. 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.

Weighted model disagreement. A cheaper surrogate replaces the log-determinant by the trace of $\Sigma_\varepsilon^{-1}\Sigma_\mu(\chi)$. This resembles information-matrix criteria in optimal design (Chaloner & Verdinelli, 1995; Pukelsheim, 2006), but is not itself a posterior ellipsoid volume or exact mutual information for a categorical model index. Unlike EIG, which is bounded by $H(M \mid \mathcal{D})$, this trace can be arbitrarily large. In practice, we therefore use the following

$$\chi^\star = \arg\max_{\chi \in \mathcal{X}} \ \mathrm{tr}\big(\Sigma_\varepsilon^{-1} \Sigma_\mu(\chi)\big) \tag{58}$$

$$= \arg\max_{\chi} \ \textstyle\sum_m p(m \mid \mathcal{D}) \, \big\|\bar{\mu}_m(\chi) - \bar{\mu}(\chi)\big\|^2_{\Sigma_\varepsilon^{-1}}, \tag{59}$$

$$\bar{\mu}_m(\chi) = \mathbb{E}_{\theta|m,\mathcal{D}}[\mathbb{E}[Y_\chi \mid m, \theta, \chi]] \approx \sum_i W_i^m \mu_{m,\theta_m^i}(\chi) \tag{60}$$

$$\bar{\mu}(\chi) = \sum_m p(m \mid \mathcal{D}) \, \bar{\mu}_m(\chi) \tag{61}$$

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* $\bar{\mu}_m(\chi)$ 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. With a finite model pool and exact conditional predictive densities, an unbiased outer Monte Carlo estimator of EIG is:

$$\widehat{\mathrm{EIG}}(\chi) = \frac{1}{N} \sum_{n=1}^{N} \log \frac{p(y_n \mid m_n, \chi)}{\sum_{m'} p(m' \mid \mathcal{D}) \, p(y_n \mid m', \chi)}, \qquad m_n \sim p(m \mid \mathcal{D}), \ \ y_n \sim p(Y_\chi \mid m_n, \mathcal{D}), \tag{62}$$

where the inner sum is the marginal predictive $p(y_n \mid \chi, \mathcal{D})$. (A further inner loop over the particles θ appears when parameters are not marginalised analytically). This is the estimator BOXINGGYM's `info_gain` evaluates. The displayed estimator is unbiased under those assumptions. Approximating the conditional densities inside the logarithm introduces additional finite-inner-sample bias; that nested approximation is distinct from the outer Monte Carlo error. The finite-model sum 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. (59) 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 vari-

[10] The *Loewner (partial) order* on symmetric matrices: $A \succeq B$ iff $A - B$ is positive semidefinite (Pukelsheim, 2006). With the same noise covariance at both designs, $\Sigma_\mu(\chi) \succeq \Sigma_\mu(\chi')$ implies Eq. (56) is at least as large at χ, so a design that raises the between-class covariance in *every* direction has a no-smaller surrogate score. For design-dependent noise, compare noise-whitened covariances instead.

ance splits into the between-class term of Eq. (59) plus a within-class one:

$$\operatorname{tr}\Big(\Sigma_\varepsilon^{-1}\operatorname{Var}_{p(m,\theta\mid\mathcal{D})}\big[\,\mathbb{E}(Y_\chi \mid m,\theta,\mathrm{do}(\chi))\,\big]\Big) = \underbrace{\sum_m p(m\mid\mathcal{D})\big\|\bar\mu_m(\chi)-\bar\mu(\chi)\big\|^2_{\Sigma_\varepsilon^{-1}}}_{\text{between classes (mechanism disagreement)}} + \underbrace{\sum_m p(m\mid\mathcal{D})\sum_i W_i^m\big\|\mu_{m,i}(\chi)-\bar\mu_m(\chi)\big\|^2_{\Sigma_\varepsilon^{-1}}}_{\text{within class (parameter uncertainty)}}, \tag{63}$$

estimated empirically by the pooled particle variance $\sum_{m,i} w_{m,i}\big\|\mu_{m,i}(\chi)-\bar\mu(\chi)\big\|^2_{\Sigma_\varepsilon^{-1}}$ 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. (59). However, optimizing Eq. (63) can be useful if the parameter values contain more useful information than just the model index. (We tried this version in Section E.6, but found it to be less numerically stable than Eq. (59).)

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).

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 $I(M;Y_\chi)$, 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. (50) with the terminal action set to *forecasting the target* rather than identifying the model: at a future experimental query $\chi_q\sim\mathcal{Q}$, nature returns observation Y_{χ_q}, from which we get feature $F_q = F(Y_{\chi_q})$; 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

$$\mathcal{R}_\ell(\chi_q\mid\mathcal{D}) \;=\; \min_{\hat F_q}\,\mathbb{E}_{p(Y_q\mid\chi_q,\mathcal{D})}\,\ell\big(F(Y_q),\hat F_q\big), \tag{64}$$

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

$$\mathrm{VoI}_\ell^{\mathcal{Q}}(\chi) \;=\; \mathbb{E}_{\chi_q\sim\mathcal{Q}}\Big[\,\mathcal{R}_\ell(\chi_q\mid\mathcal{D}) \;-\; \mathbb{E}_{y_\chi\sim p(\cdot\mid\chi,\mathcal{D})}\,\mathcal{R}_\ell(\chi_q\mid\mathcal{D},y_\chi,\chi)\Big]. \tag{65}$$

This is exactly Eq. (50) with decision $d=\hat F$ and utility $U=-\ell$ evaluated on the target: it designs χ 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). We leave systematic invesigation of this objective to future work.

B CHEMISTRY

B.1 BENCHMARK AND METRICS

ActiveSciBench-Chem (Kabra et al., 2026), which we call CHEMBENCH, asks an agent to discover a static rate law

$$y = f(C_A, C_I, C_B, C_P, \mathrm{Enz}, T, \mathrm{pH}; \theta). \tag{66}$$

The seven inputs control substrate, inhibitor, second-substrate and product concentrations, enzyme loading, temperature, and pH. The 57 supported worlds comprise nine canonical mechanisms and 48 compound mechanisms. We evaluate every world at medium difficulty, and 1% multiplicative observation noise. We use three replicate IDs per world; these are algorithmic seeds, not three redraws of the observation noise at a fixed design. The oracle deterministically hashes the world and seven-dimensional condition to its noise draw, so repeating exactly the same condition returns the same measured rate. The seed changes MDA's particle fitting and design search, LLM-AUTOSCILAB's controller trajectory, and the held-out query conditions; different selected conditions therefore generally encounter different deterministic noise values.

The benchmark evaluates a submitted law on novel conditions using the root mean square log error:

$$\mathrm{RMSLE} = \left[\frac{1}{N}\sum_{i=1}^{N}\left(\log(1+\hat{y}_i) - \log(1+y_i)\right)^2\right]^{1/2}. \tag{67}$$

Our held-out predictive score specializes Eq. (2) to a scalar log-rate target, with s_1^2 equal to its test-set variance:

$$\mathrm{nMSLE} = \frac{N^{-1}\sum_i \left(\log(1+\hat{y}_i) - \log(1+y_i)\right)^2}{\mathrm{Var}[\log(1+y_i)]}. \tag{68}$$

We report the *benchmark pass rate*, the fraction of runs satisfying $\mathbb{1}[\mathrm{RMSLE} < 0.05]$. Although the paper text of Kabra et al. (2026) mentions 0.01, its code uses 0.05, which we follow. A law that cannot be evaluated over the declared benchmark test range counts as a failure.

B.2 MDA MODEL AND EXPERIMENTAL PROTOCOL

Each LLM-proposed law has free constants θ with positive bounds. The primary ChemBench implementation uses bounded parameter point estimation and BIC weights in Eq. (45); its predictor selects the highest-weight structure and uses $f_m(a; \hat{\theta}_m)$. The SMC control instead uses independent bounded priors for each parameter, an explicit complexity prior $p(m) \propto \exp(-2C_m)$, and a particle approximation to $p(\theta \mid m, \mathcal{D})$ and the marginal evidence. Both fit residuals in log-rate space.

The primary hybrid $\mathcal{M}$-open arm begins with nine canonical executable laws and makes one LLM initialization call requesting 12 additional laws, giving a nominal initial pool of up to 21 before pruning; screening and deduplication can reduce this number. When the resulting pool fails the predictive-adequacy check, each subsequent expansion call requests a batch of six residual-directed proposals; the live pool is score-pruned to at most 24 models, and can shrink to six after a confident, adequate fit as described below. The primary arm uses BIC weights and a point estimate, so the expectation in Eq. (69) is evaluated at $\hat{\theta}_m$.

Initialization. The MDA agent is initialized with three fixed passive observations followed by a sequence of active experiments of batch size 1. The passive set holds $(C_I, C_B, C_P, \mathrm{Enz}, T, \mathrm{pH}) = (0, 1, 0, 1, 310, 7)$ fixed and uses $C_A \in \{0.03, 0.3, 3\}$. It is a law-agnostic substrate sweep, shared in form across every MDA cell.

Experiment design. After fitting the hypothesis pool to these three points, each active round scores a condition by the posterior between-model variance

$$V(a) = \sum_m p(m \mid \mathcal{D})\left(\bar{\mu}_m(a) - \sum_{m'} p(m' \mid \mathcal{D})\bar{\mu}_{m'}(a)\right)^2, \qquad \bar{\mu}_m(a) = \mathbb{E}_{p(\theta \mid m, \mathcal{D})}[f_m(a; \theta)]. \tag{69}$$

This is the same as Eq. (59) without the $\Sigma_\epsilon(a)^{-1}$ factor. Under the benchmark's multiplicative noise this makes it a model-disagreement proxy rather than the exact implemented EIG objective.

CMA-ES maximizes the objective in Eq. (69) over the continuous normalized seven-dimensional box, using 48 objective evaluations per round; concentration and enzyme axes are mapped logarithmically and temperature and pH linearly. The oracle evaluates the maximizing condition, then MDA refits before choosing the next one. Thus the active designs are adaptive and generally differ across arms, worlds, and replicate IDs.

Adaptive pool size and pruning. After fitting each update, MDA checks whether the largest normalized model weight exceeds 0.9 and its residual ratio is at most 1 (median relative fit residual at most 0.05). If both hold, it retains the six highest-scoring models and refits; otherwise the cap is 24. For the BIC arm, ranking uses BIC weights. Raising the cap does not itself restore discarded candidates: growth requires new proposals. Exact-expression matching prevents re-adding an expression already in the pool, but this is not a similarity-aware merge of predictions or a guarantee against splitting weight among near-equivalent laws. Pruning can discard a law that would become useful later; the live pool is not an indefinitely retained archive of every proposal.

The pool sizes in Fig. 2b are measured after proposal, fitting, and pruning. At $B = 6$, five new retained expressions take the pool from nine to 14; at $B = 8$, adaptive shrinkage removes eight, leaving six. At $B = 15$, the proposal transcript requests six candidates, and three new expressions survive while three old ones leave: the final size remains six. The red markers indicate calls that enlarged the candidate set before pruning, not a positive net change in the final live pool.

B.3 Baseline

We run the complete LLM-AutoSciLab pipeline independently at each budget $B \in \{5, 10, 15, 20, 25\}$, using GPT-5.6-Luna with medium reasoning effort. Each run collects exactly B observations and receives one final 800-iteration PySR fit; intermediate symbolic refinement is disabled, including at $B = 20$ where the upstream default would otherwise add another fit.

In our initial experiments we noticed a very large error spike at $B = 10$ (a smaller spike is still visible in Fig. 2a). The cause was LLM-AutoSciLab's model selector, which favored nearly saturated seven-variable power laws because it compared their *training scores* against other candidates' *validation scores*, allowing severe overfitting. (This effect does not appear with the larger experimental budgets used in their paper.) By slightly modifying their code to rank all candidates on the same validation split, we reduced the error by about $9\times$ on the matched cohort (although a smaller spike remains), without changing results at other sample sizes.

B.4 Results

Sample efficiency. The primary sample efficiency results are shown in Fig. 2a. Although there are 171 (world, seed) combinations, the baseline LLM-AutoSciLab did not submit a valid law for each combination at all budgets. We therefore restrict all four methods to the same 115 (world, seed) cells evaluable by corrected LLM-AutoSciLab at every budget. These span 54 worlds, with unequal numbers of retained replicates. We average seed errors arithmetically within each world, then take the geometric mean across worlds; 95% intervals use 5,000 world-bootstrap replicates. The curve measures prediction quality among evaluable runs; invalid submissions count as failures in the benchmark pass rate. At $B = 20$ and 25, MDA's paired nMSLE ratios to corrected LLM-AutoSciLab are 0.00624 (95% CI 0.00263–0.0158) and 0.00351 (0.00152–0.00830), respectively. The all-cell pass rates are 74.9% and 78.4% for MDA versus 15.8% at both budgets for LLM-AutoSciLab, retaining all 171 cells per budget. We see that MDA is significantly more sample efficient than LLM-AutoSciLab and the ICL baselines. ICL with its own LLM-selected designs slightly outperforms ICL on MDA's history: designs that distinguish executable models need not be the most useful examples for an in-context predictor. This does not contradict the active-versus-random benefit for MDA itself.

Benefits of $\mathcal{M}$-open. Figure 6 shows the benefits of the $\mathcal{M}$-open setting: we compare using a fixed set of 9 possible mechanisms vs allowing the agent to grow the hypothesis space using the LLM. On the left we plot test nMSE, and on the right, we plot pass rate. We see the $\mathcal{M}$-open setting

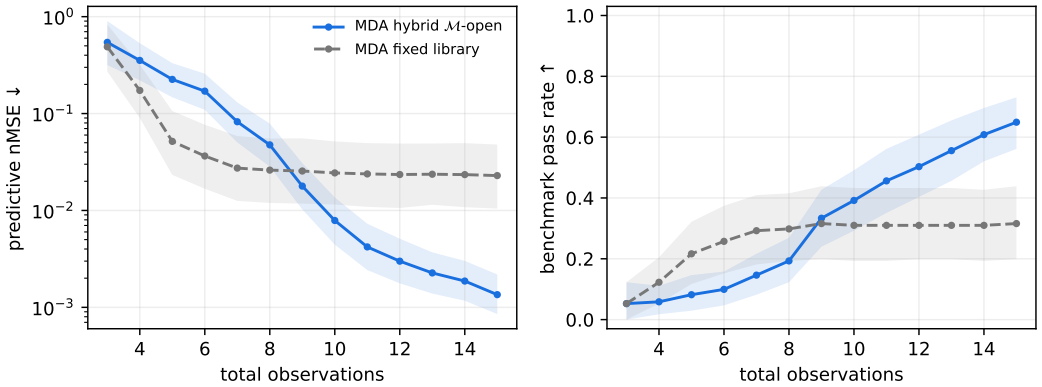


Figure 6: **Benefits of LLM-augmented model search.** Both arms use the BIC backend, CMA-ES maximization of $V(a)$, identical three-point passive design, and the same 57 worlds and three replicate IDs; the grey arm is restricted to the nine executable seed laws and makes no LLM calls, whereas the blue arm permits LLM initialization and residual-directed expansion. At $B = 15$, hybrid minus fixed pass rate is $+33.3$ points over all worlds (95% CI $+22.8$ to $+43.9$) and $+39.6$ points over the 48 out-of-library worlds ($+27.8$ to $+51.4$). The corresponding hybrid/fixed nMSE ratios are 0.059 (0.032–0.109) and 0.0356 (0.0185–0.0681). On the nine in-library worlds both pass every cell. Intervals average the three replicates within world and bootstrap worlds.

is much better. This measures the combined benefit of LLM initialization and later expansion, rather than isolating later expansion alone.

Benefits of EIG vs random designs. Figure 7 shows the benefits of active experiment design (left panel, optimizing $V(a)$ using CMA) vs using random experiment design (right panel). We also compare using tempered SMC to estimate the evidence (blue lines) vs BIC with a fitted residual scale (red) or fixed scale (green). These configurations also differ in parameter treatment and model prior: the SMC arm uses $p(m) \propto e^{-2C_m}$. Thus the curves do not isolate evidence approximation alone. The BIC configurations predict better here, with the fixed τ variant in Eq. (41) slightly better (in the EIG case) than the profiled τ variant in Eq. (45) in this cohort.

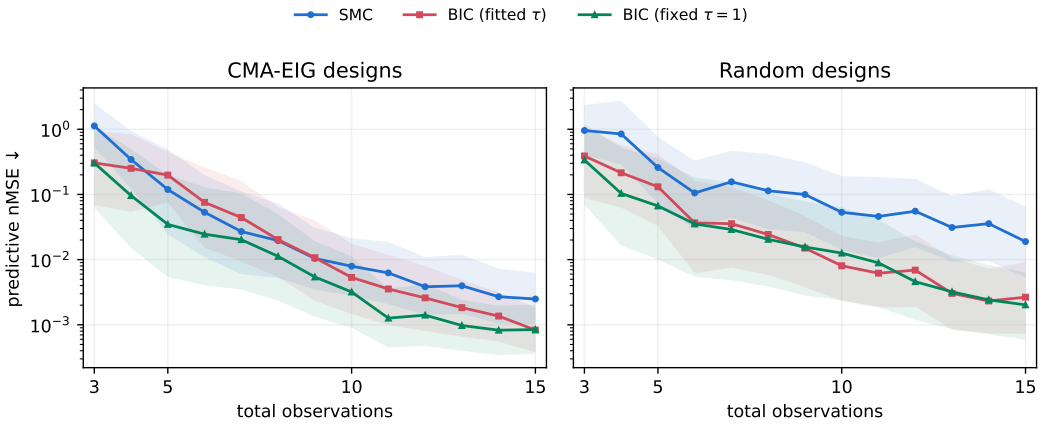


Figure 7: **ChemBench inference configurations × experiment design.** Twelve predeclared worlds and three seeds per world; all six arms receive the same three passive observations and then collect 12 active observations. Both panels show held-out predictive nMSE: left, designs optimize $V(a)$ with CMA-ES; right, random designs. Colors identify the same three inference methods in both panels, with shared axes. BIC denotes bounded multi-start point fitting with weights proportional to $e^{-\mathrm{BIC}/2}$; SMC uses 100 parameter particles per structure. Lines are geometric means over worlds and replicates; bands are 95% world-bootstrap intervals. At 15 observations, CMA-ES versus random reduces nMSE by factors 0.314 $(0.127–0.698)$ under BIC and 0.132 $(0.061–0.287)$ under SMC; the BIC+CMA versus SMC+CMA nMSE ratio is 0.334 $(0.163–0.694)$. These BIC contrasts use the original fitted-scale arms. The added green arms fix $\tau = 1$, replay their paired fitted-scale arm's initial LLM proposal, and subsequently run their own inference, expansion, and design loops. Random arms share the same design histories. The budget-geometric comparison integrates log nMSE over budgets with the trapezoidal rule before forming paired world-bootstrap ratios.

C GLUCOSEBENCH: A PARTIALLY OBSERVED ODE DOMAIN

C.1 DOMAIN AND EXPERIMENTAL PROTOCOL

GLUCOSEBENCH is based on a modified version of `simglucose`[11], a Python implementation of the UVA/Padova type-1 diabetes model from (Kovatchev et al., 2009), whose original simulator was accepted by the FDA for preclinical testing of insulin- control algorithms in place of animal trials (Cobelli & Kovatchev, 2023). We modified the model to allow partial noisy observations of both glucose (every 5 minutes) and insulin (every 15 minutes), for up to 6 hours. Meals and injected insulin drive hidden physiological compartments, while a continuous glucose monitor (CGM) gives noisy glucose measurements. The 30 virtual profiles comprise ten children, ten adolescents, and ten adults. Physiology is deterministic conditional on the profile and inputs; uncertainty comes from observation noise and model misspecification, not stochastic physiological transitions.

Patient-specific training and held-out interventions. Each profile has its own training episodes, candidate archive, and fitted parameters. Its test set contains new intervention episodes from that same profile, not different patients. No data are pooled across patients. Learners receive observations, masks, measurement units, and delivered inputs, including basal insulin, but not age, patient ID, body weight, simulator parameters, or latent states. Thus the task measures personalized intervention generalization, not unseen-patient transfer.

Each episode lasts six hours. CGM is sampled every five minutes (73 readings, including time zero), with temporally correlated native sensor noise. We add a plasma-insulin assay every 15 minutes (25 readings): the oracle converts its private insulin mass to concentration and adds independent Gaussian noise with standard deviation 10 pmol/L. This is an added observation model, not a native sensor. Missing assays have an explicit mask and contribute nothing to fitting or evaluation.

True data-generating process. The generator uses a private 13-dimensional patient state: $x = (Q_{sto1}, Q_{sto2}, Q_{gut}, G_p, G_t, I_p, X, I_1, I_d, I_l, I_{sc1}, I_{sc2}, G_{sc})$. Conditional on a virtual-patient parameter vector and the known meal and insulin inputs, its continuous-time dynamics are deterministic (the equations below describe the nonnegative physiological domain):

$$\begin{aligned}
\dot{Q}_{sto1} &= -k_{\max} Q_{sto1} + D, & \dot{Q}_{sto2} &= k_{\max} Q_{sto1} - k_{gut} Q_{sto2},\\
\dot{Q}_{gut} &= k_{gut} Q_{sto2} - k_{abs} Q_{gut}, & \dot{G}_p &= EGP + R_a - F_{snc} - E - k_1 G_p + k_2 G_t,\\
\dot{G}_t &= -U_{id} + k_1 G_p - k_2 G_t, & \dot{I}_p &= -(m_2 + m_4) I_p + m_1 I_l + k_{a1} I_{sc1} + k_{a2} I_{sc2},\\
\dot{X} &= -p_{2U} X + p_{2U}(I - I_b), & \dot{I}_1 &= -k_i (I_1 - I),\\
\dot{I}_d &= -k_i (I_d - I_1), & \dot{I}_l &= -(m_1 + m_{30}) I_l + m_2 I_p,\\
\dot{I}_{sc1} &= U_I - (k_{a1} + k_d) I_{sc1}, & \dot{I}_{sc2} &= k_d I_{sc1} - k_{a2} I_{sc2},\\
\dot{G}_{sc} &= -k_{sc} G_{sc} + k_{sc} G_p.
\end{aligned} \tag{70}$$

All coefficients in Eq. (70) are fixed by the private virtual-patient profile. The nonlinear fluxes are

$$\begin{aligned}
Q_{sto} &= Q_{sto1} + Q_{sto2},\\
k_{gut} &= k_{\min} + \frac{k_{\max} - k_{\min}}{2}\left[\tanh\{\alpha(Q_{sto} - b\bar{D})\} - \tanh\{\gamma(Q_{sto} - d\bar{D})\} + 2\right],\\
\alpha &= \frac{5}{2\bar{D}(1-b)}, \qquad \gamma = \frac{5}{2\bar{D}d}, \qquad R_a = \frac{f k_{abs} Q_{gut}}{BW},\\
EGP &= [k_{p1} - k_{p2} G_p - k_{p3} I_d]_+, \qquad E = k_{e1}[G_p - k_{e2}]_+,\\
I &= I_p / V_i, \qquad U_{id} = \frac{(V_{m0} + V_{mx} X) G_t}{K_{m0} + G_t}.
\end{aligned} \tag{71}$$

When no meal is active and $\bar{D} = 0$, the implementation uses $k_{gut} = k_{\max}$ rather than evaluating the displayed ratios. Here $D(\tau) = 1000 m(\tau)$ converts grams of ingested carbohydrate to mg, $\bar{D}$ is the current meal's cumulative consumed load plus pre-meal stomach content, and $U_I(\tau) = 6000\{u_b(\tau) + a(\tau)\}/BW$ converts basal and bolus insulin to the simulator's per-kilogram units.

[11] https://github.com/jxx123/simglucose; simulator revision `2e897f1`.

The corpus adapter delivers a meal at at most 5 g/min and spreads each bolus over five minutes. The pinned implementation uses adaptive Dormand–Prince integration with inputs updated every minute, and we sample every $\Delta = 5$ minutes.

Our extension: the two-channel observation model. At sample time $t_j = j\Delta$, our public observation process is

$$\begin{aligned} y_j^C &= \mathrm{clip}_{[L,U]}\left(G_{sc}(t_j)/V_g + \epsilon_j^C\right), \quad M_j^C = 1, \\ y_j^I &= I_p(t_j)/V_i + \epsilon_j^I, \qquad\qquad M_j^I = \mathbb{I}\{t_j \bmod 15\ \text{min} = 0\}, \\ \epsilon_j^I &\overset{\text{iid}}{\sim} \mathcal{N}(0, 10^2) \qquad\qquad \text{pmol/L}. \end{aligned} \tag{72}$$

The GuardianRT CGM error ϵ_j^C is the simulator's colored, Johnson–SU-transformed process: it evolves at 15-minute knots and is cubically interpolated to the five-minute grid before sensor-range clipping. Plasma insulin is our added noisy assay, without rounding, and unobserved between 15-minute assay times.

Input sequences. An experiment selects a template for an entire input sequence, rather than adapting inputs within the episode. Background basal insulin is fixed at the profile's prescribed simulator rate and is public to all methods. The four design controls $a_t^{\text{meal-amt}}$, $a_t^{\text{meal-time}}$, a_t^{bolus}, and $a_t^{\text{bolus-delay}}$ specify a schedule chosen once per experiment. A scheduler converts them into delivery rates. The menu crosses

$$\begin{aligned} \text{meal amount} &\in \{20, 40, 60\}\ \text{g}, & \text{meal time} &\in \{60, 120\}\ \text{min}, \\ \text{bolus} &\in \{0, 0.5, 1, 1.5\}\ \text{U}, & \text{bolus delay} &\in \{0, 30\}\ \text{min}. \end{aligned}$$

U denotes units of insulin activity; U/min is a delivery rate. Delay is relative to meal time. Meals are consumed at 5 g/min and boluses delivered over five minutes; we record actual delivery after pump discretization. There are 48 nominal templates but 42 distinct interventions, since delay does not matter for zero bolus. Sampling is without replacement over indices, so equivalent zero-bolus interventions can recur.

Training set. All methods start with the same two passive episodes: a 30-g meal at minute 60 with a 1-U bolus at minute 75, and a 45-g meal at minute 90 with a 1.5-U bolus at minute 90. Four additional experiments are selected; evaluation budgets are $B_k \in \{2, 3, 4, 6\}$ total episodes.

Test set. Each patient has twelve frozen held-out interventions, generated by Latin hypercube sampling over meal amount 20–60 g, time 60–120 min, bolus 0–1.5 U, and delay 0–30 min (times rounded to five minutes). These include inputs between training-menu levels.

Evaluation. For each patient, we use the trajectory nMSE in Eq. (9), with one scale per channel, s_c, defined by the empirical variance of observed values in the two shared initial training episodes $\mathcal{D}_0$:

$$\begin{aligned} L_c &= \frac{1}{N_c s_c^2} \sum_q \sum_{t > t_c} o_{qtc} (\widehat{y}_{qtc} - y_{qtc})^2 \\ N_c &= \sum_q \sum_{t > t_c} o_{qtc}, \qquad s_c^2 = \max\left\{ \frac{1}{|\mathcal{Y}_{0c}|} \sum_{y \in \mathcal{Y}_{0c}} (y - \bar{y}_{0c})^2, 1 \right\}. \end{aligned} \tag{73}$$

Here $\mathcal{Y}_{0c}$ contains all observed channel-c values in $\mathcal{D}_0$, $\bar{y}_{0c}$ is their mean, and o_{qtc} is the observation mask. The scale floor is one squared native unit per observation. We normalize channels separately rather than adding errors in incompatible units, and take geometric means over profiles. This scale is fixed across methods, budgets and forecast cuts and uses no test outcomes. Our main figure is the CGM, $t_c = 45$ slice of the multi-cut comparison below, with the same fixed paired-query set.

C.2 MDA

In this section, we describe how we applied MDA to this domain.

Models. The backend supplies four reduced compartmental ODEs before seeing $\mathcal{D}_0$; their parameters are then fitted to $\mathcal{D}_0$. These comprise one-stage meal/insulin absorption with linear or glucose-dependent insulin action, two-stage meal absorption, and a delayed-CGM-readout model. The four-state, two-stage meal model is shown on the left of Fig. 10, and corresponds to these equations:

$$\begin{aligned} \dot{z}^{q_1} &= u_{\text{meal}} - z^{q_1}/\tau_{q_1}, \qquad \dot{z}^{q_2} = (z^{q_1} - z^{q_2})/\tau_{q_2}, \\ \dot{z}^{i} &= u_{\text{bolus}} - z^{i}/\tau_{i}, \qquad \dot{z}^{g} = k_g(g_b c_0 - z^g) + \alpha z^{q_2} - \beta z^i z^g/g_b. \end{aligned} \tag{74}$$

Here c_0 is a fitted equilibrium multiplier, while z^{q_1}, z^{q_2}, z^i are effective input-memory compartments. These seed ODEs provide mechanistic inductive bias not supplied as executable models to ICL; the system comparison does not isolate the contribution of LLM births. I_b is the average insulin concentration over the observed 45-minute prefixes of the training episodes, and g_b is the average CGM value. The initial glucose mean is g_b, and input-memory means are zero; the hidden states are then updated by Bayesian filtering (see below).

The conditional observation means are $\widehat{y}_t^{\text{CGM}} = z_t^g$ and $\widehat{y}_t^{\text{insulin}} = I_b + \lambda z_t^i$, with fitted positive gain λ. We fit an AR(1) model to CGM residuals. For EKF prediction we represent it using an extra real-valued state e_t:

$$e_t = \rho e_{t-1} + \eta_t, \quad \eta_t \sim \mathcal{N}(0, \sigma_\eta^2), \qquad y_t^{\text{CGM}} = h_{\text{CGM}}(z_t) + e_t. \tag{75}$$

The full latent state is thus (z_t, e_t). Physiological coordinates follow the deterministic ODE with no added process noise; only the sensor-error coordinate receives these Gaussian innovations. Insulin assays retain their separate observation noise.

The LLM edits dynamics and observation maps using restricted arithmetic expressions, including adding latent compartments. Initial models have three or four states; proposals may have six states and twelve fitted parameters. State names sometimes describe a role (e.g., `qfast`, `qslow`, `cgm`), but do not establish physiological identifiability.

Parameter estimation. Parameters are fitted by multistart L-BFGS-B using the joint working likelihood: correlated Gaussian CGM residuals and independent Gaussian insulin-assay errors. The CGM innovation variance is profiled out; insulin noise is fixed at its declared value. BIC counts the ODE parameters plus the fitted CGM correlation and innovation scale, using the number of observed scalar targets as its sample size.

Predictive uncertainty. The fully Bayesian prediction integrates uncertainty in states, parameters, and model structure, while retaining observation noise in the predictive likelihood, as follows (where we drop conditioning on the known design and context for brevity):

$$\begin{aligned} p(Y_{t_c+1:T} \mid \mathcal{H}, \mathcal{D}) = \sum_m \int\int & p(\widetilde{z}_{t_c} \mid \mathcal{H}, m, \theta, \mathcal{D})\, p(Y_{t_c+1:T} \mid \widetilde{z}_{t_c}, m, \theta) \\ & \cdot p(\theta \mid m, \mathcal{H}, \mathcal{D})\, p(m \mid \mathcal{H}, \mathcal{D})\, d\widetilde{z}_{t_c}\, d\theta. \end{aligned} \tag{76}$$

Here $\mathcal{H} = \mathcal{H}_{t_c}$ is the query prefix and $\widetilde{z}_t = (z_t, e_t)$ includes the sensor-error state. For the displayed forecasts, the first factor, $p(\widetilde{z}_{t_c} \mid \mathcal{H}, m, \theta, \mathcal{D})$, is obtained by approximate Bayesian filtering, based on the Extended Kalman Filter (Särkkä & Svensson, 2023). The second factor, $p(Y_{t_c+1:T} \mid \widetilde{z}_{t_c}, m, \theta)$, propagates the filtered state into the future (using the EKF without any evidence updates), and converts this into predicted observations. We approximate the third factor, $p(\theta \mid m, \mathcal{H}, \mathcal{D})$ by $p(\theta \mid m, \mathcal{D})$, and then further approximate this by the point estimate $\delta(\theta - \hat{\theta}_m)$. Finally we approximate the fourth factor, $p(m \mid \mathcal{H}, \mathcal{D})$, by $p(m \mid \mathcal{D})$, which is represented by a weighted histogram, where the weights are derived from the BIC scores of each model in the library. We show some sample forecasts in Fig. 9.

Experiment design. MDA maximizes BIC-weighted between-model mean-prediction disagreement, normalized by observation-noise variance and averaged over the two channels at their observation times. Specifically, for a candidate input schedule a, let $\mu_{m,t,c}(a; \hat{\theta}_m)$ denote model m's ODE prediction for channel c at time t, using its fitted parameters, and let $w_m \propto \exp(-\text{BIC}_m/2)$

be normalized model weights. We score

$$V(a) = \frac{1}{2} \sum_{c \in \{\text{CGM,insulin}\}} \frac{1}{|\mathcal{T}_c|} \sum_{t \in \mathcal{T}_c} \frac{\text{Var}_{m \sim w}[\mu_{m,t,c}(a; \hat{\theta}_m)]}{\nu_c^2}, \tag{77}$$

where $\mathcal{T}_c$ contains scheduled observation times after initialization. The CGM scale is the model-weighted stationary variance of the fitted AR(1) residuals, $\nu^2_{\text{CGM}} = \max\{1, \sum_m w_m \sigma_m^2/(1-\rho_m^2)\}$; the insulin scale is the declared assay-noise variance. These acquisition scales differ from the evaluation normalization s_c^2. This is a noise-normalized, trajectory-valued counterpart of Eq. (69), giving equal weight to the two channels. Parameters are held at point estimates rather than integrated out, and we maximize over unused menu indices rather than by CMA-ES (ties are broken randomly). Note that this score is a model-disagreement surrogate for EIG, not exact EIG; within-model predictive variance is not included in its numerator.

C.3 Baselines

ARX. As a baseline, we consider fitting a linear autoregressive model with exogenous inputs (Lütkepohl, 2005) for each observation channel separately:

$$\widetilde{y}_t = b + \sum_{\ell=1}^{P} \alpha_\ell \widetilde{y}_{t-\ell} + \sum_{j=0}^{L-1} \beta_j^\top u_{t-1-j} + \epsilon_t. \tag{78}$$

Here u_t contains delivered meal and bolus amounts in sampling interval t, and $\widetilde{y}_t$ subtracts an episode baseline. The plotted control fits separate CGM and insulin regressions on their five- and fifteen-minute grids, centering by their respective 45-minute prefix means. There are no cross-channel lags.

VARX. We also consider a joint VARX (vector autoregressive) model which predicts $\widetilde{\mathbf{y}}_t = (\widetilde{y}_t^{\text{CGM}}, \widetilde{y}_t^{\text{insulin}})^\top$ using

$$\widetilde{\mathbf{y}}_t = \mathbf{b} + \sum_{\ell=1}^{P} A_\ell \widetilde{\mathbf{y}}_{t-\ell} + \sum_{j=0}^{L-1} B_j u_{t-1-j} + \boldsymbol{\epsilon}_t, \qquad A_\ell, B_j \in \mathbb{R}^{2\times 2}. \tag{79}$$

Off-diagonal A_ℓ entries allow cross-channel prediction. Our implementation uses the five-minute grid, centering each channel by its initial observation. Missing insulin predictors are carried forward from the last observed assay, never interpolated from future assays; only observed targets enter the loss:

$$\sum_{e,t,c} o_{etc}(\widetilde{y}_{etc} - w_c^\top x_{et})^2 + \lambda \sum_c \|S w_c\|_2^2. \tag{80}$$

The vector x_{et} contains an intercept, channel lags and input lags; S is diagonal training-feature RMS scaling floored at one. Ridge includes the intercept.

In both cases, lag order and regularization use whole-training-episode cross-validation. Forecasts recurse on predicted outputs after the cut.

SINDy. SINDy (Sparse identification of nonlinear dynamics) (Brunton et al., 2016a) identifies sparse combinations of candidate functions; we use its controlled form, SINDYc (Brunton et al., 2016b), implemented with PySINDy's sequentially thresholded least squares. The two state coordinates are observed CGM and insulin, with known meal and bolus inputs; an optional pair of fixed 30-minute exponential input memories represents delayed effects. No simulator states or fitted MDA mechanisms are provided. This control reuses MDA's training episodes, not its acquisition rule.

We select degree-one or degree-two polynomial libraries, smoothing windows of five or nine samples, input memories on/off, and thresholds $\{0, 0.005, 0.02\}$ (ridge penalty 0.1): 24 configurations. Missing training insulin is linearly interpolated between assays; quadratic Savitzky–Golay smoothing estimates derivatives on the five-minute grid. Scaling and leave-one-episode-out selection use training data only, scoring the two-channel suffix after a 45-minute prefix. Before query evaluation, full-data refits are screened on training inputs at all four cuts and prefixes perturbed by one training standard deviation. We reject divergent predictions, excursions beyond the observed training range plus six standard deviations, or scaled RMS changes above 0.02 when halving the RK4 step from

one minute. The first passing CV-ranked model is retained. This is a finite-horizon safeguard, not a global stability proof.

At prediction time, available prefix observations reset their corresponding coordinates; missing assays are propagated, not filled using future observations. Cut zero uses training-only initial values. After the cut, forecasts are fully recursive with no parameter refitting, clipping, or fallback. All 120 fits and 5,760 forecasts completed; independent replay reproduced the fits and predictions. Screening changed seven CV winners; 117 selected models are linear and three quadratic.

C.4 Results

Sample efficiency curves. In Fig. 3a we show the prediction loss for CGM (after $t_c = 45$) vs number of training experiments for the different methods. We see that MDA is substantially more sample efficient.

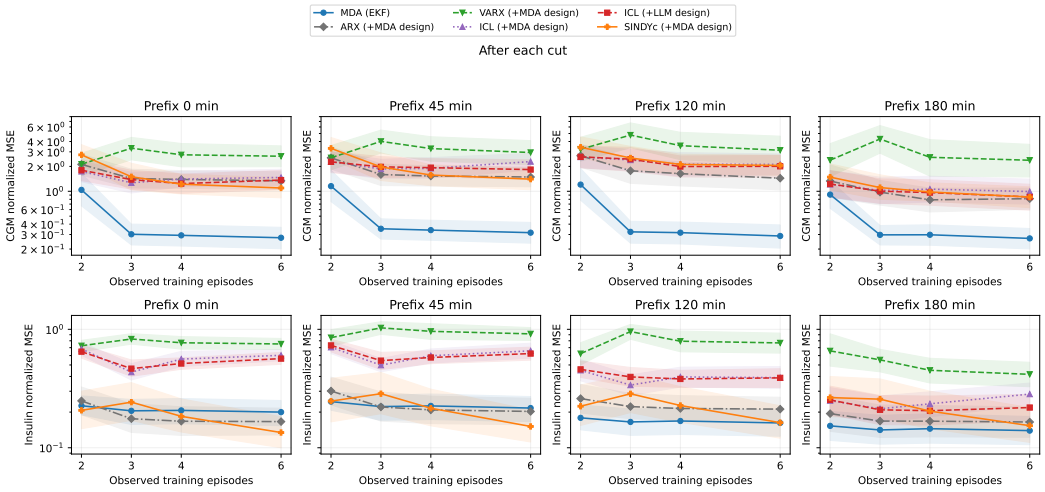


Figure 8: **Conditional forecasting at four prefix lengths.** CGM (top) and insulin (bottom), across all 30 patient profiles. (Each patient has one algorithmic replicate; bootstrap intervals reflect variation across patients, not repeated LLM runs.) MDA uses EKF state estimation and BIC model averaging at fixed fitted parameters; at cut zero it propagates training-derived initial states without query measurements. ARX, VARX, SINDYc and ICL (+MDA design) share MDA's histories; ICL (+LLM design) uses its own archived histories. Curves are patient-level geometric means of training-variance-normalized MSE; bands are 95% patient-bootstrap intervals. All curves use the same 273/360 patient–query pairs valid for every method, budget and cut. Each column scores only observations after its cut, so columns change both conditioning information and forecast horizon.

In Fig. 8 we extend the analysis by varying the cut point, using $t_c = 0, 45, 120, 180$ minutes, and showing results for GCM prediction and insulin prediction. We use the same training data across all runs, and just change the initial belief state, $p(z_{t_c} \mid \mathcal{H}_{t_c}, m, \theta, \mathcal{D})$. We plot prediction error for CGM in the top row and insulin in the bottom row, and vary t_c along the columns. We see two main effects: (1) for predicting GCM, MDA has a big lead over all other methods, whereas for predicting insulin levels, MDA ties with the simpler ARX and SINDYc baselines, because this signal is easier to predict; (2) the amount of history conditioning does not make much difference to GCM prediction, but for insulin prediction, conditioning on ≥ 120 minute prefix for this patient helps most methods to various degrees.

Forecast ability. Figure 9 gives an example of probabilistic forecasting, using an approximation of Eq. (76), for a single patient and input sequence (lower rows), using MDA's model mixture trained with 2 (left column) or 6 (right column) trajectories. The example illustrates changes in the forecast mean and uncertainty; it does not establish improved variance calibration. Model disagreement accounts for 38% of the summed CGM predictive variance at two episodes, but less than 0.001% at six episodes, when one model has 99.81% weight. The remaining variance is within-model state and observation uncertainty. We do not report calibration or proper-score comparisons here; Section D.2 discusses probabilistic scoring.

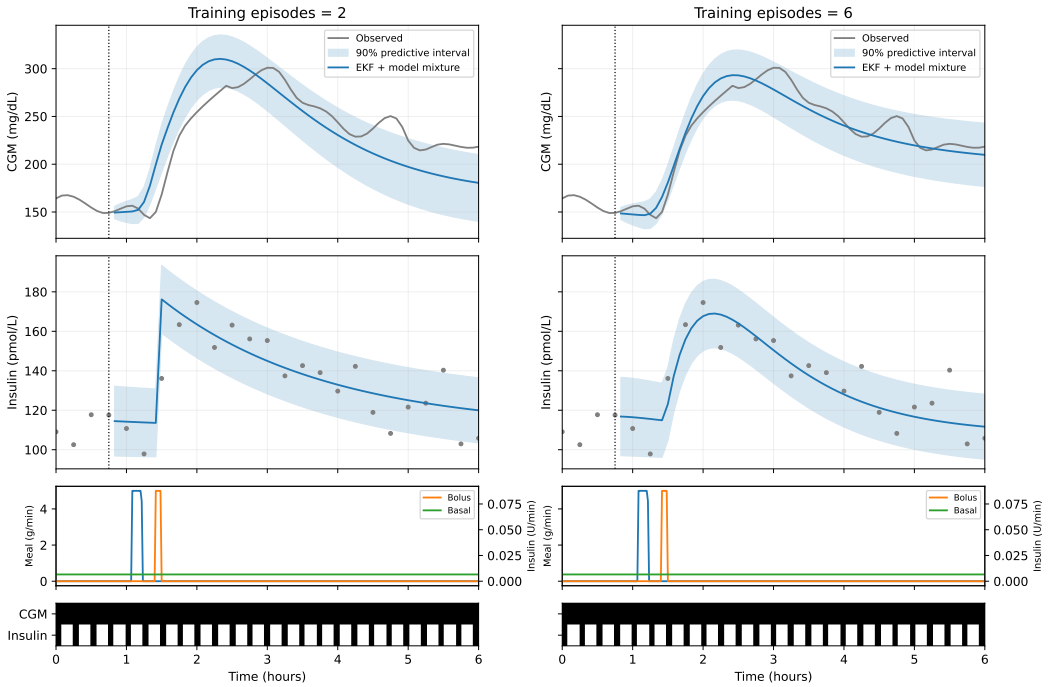


Figure 9: **Two-channel predictions before and after learning.** Sample forecasts for a single patient after two versus six training episodes. Blue curves are EKF/BIC-mixture predictive means; gray traces/points are held-out CGM/insulin observations. The dotted line marks the 45-minute forecast cut. Lower rows show delivered inputs and the observation mask. Shading shows marginal 90% predictive intervals, computed as quantiles of the Gaussian-component mixture, not a Gaussian fit to its moments. Intervals include filtered-state, observation-noise and between-model uncertainty, but *not parameter uncertainty*. The EKF's approximate covariance initialization and positivity safeguard apply; nominal 90% coverage is not a calibration claim.

Model structure learning. Figure 10 shows an example of model expansion. After two episodes the MAP model has four states $(z^g, z^{q_1}, z^{q_2}, z^i)$. A newly proposed six-state model selected after episode three adds insulin distribution and sensor lag:

$$\dot{z}^j = k_j(z^i - z^j), \qquad \dot{z}^c = (z^g - z^c)/\tau_c, \qquad \widehat{y}^{\mathrm{CGM}} = z^c, \qquad \widehat{y}^{\mathrm{insulin}} = I_b + \lambda z^j.$$

Insulin affects glucose through z^j instead of directly through z^i.

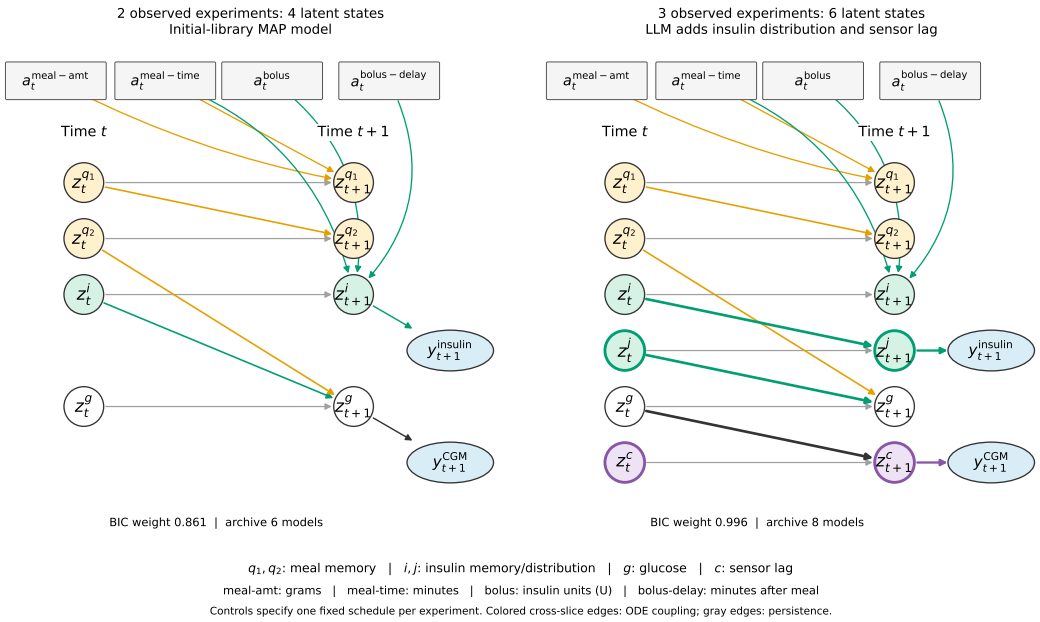


Figure 10: **A glucose model grows new latent mechanisms.** Illustration of the MAP model for a single patient after varying number of training trajectories. Left: four-state initial-library model at two episodes. Right: six-state LLM proposal born and selected at episode three. Bold outlines/edges mark added states and dependencies. Only next-slice observations are drawn. Colored cross-slice edges show ODE couplings in a local Euler rendering; gray edges show persistence. Exact finite-time integration can induce indirect dependencies. Orange denotes meal memory, green insulin memory, and purple sensor lag. Basal delivery is fixed and omitted.

D NEURONBENCH: FURTHER DETAILS

In this section, we introduce our new NEURONBENCH benchmark. We also give a brief primer on neuron electrophysiology and Hodgkin–Huxley models, to make this section comprehensible to a machine learning audience.

D.1 BACKGROUND

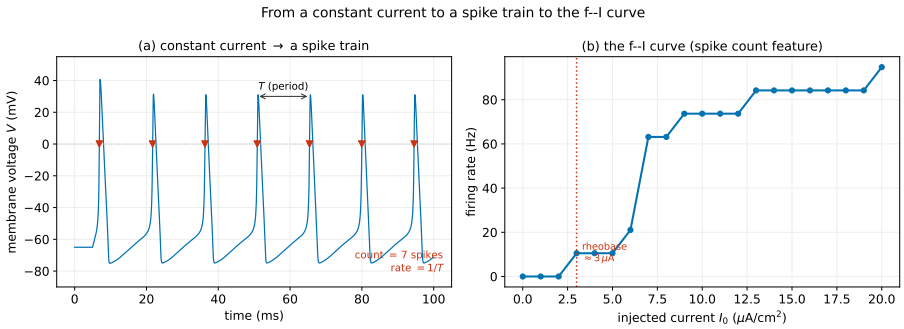


Figure 11: **The f–I curve, and why we count spikes.** *(a)* A constant supra-threshold current makes the model fire a periodic *spike train*; the readout is simply the *spike count* (red markers) — or, per unit time, the firing rate $1/T$. *(b)* Sweeping the injected current traces the *f–I curve* (firing 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 $-65\,\text{mV}$. 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 $\sim+40\,\text{mV}$ 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. 11.

Crucially, the behavior of the neuron depends on its inputs, as illustrated in Fig. 12. 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, Hodgkin and Huxley (Hodgkin & Huxley, 1952) introduced 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. 13. 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 4, each of which have their own parameters and dynamics. We can write the generalized model as a set of nonlinear ODEs, which follow from

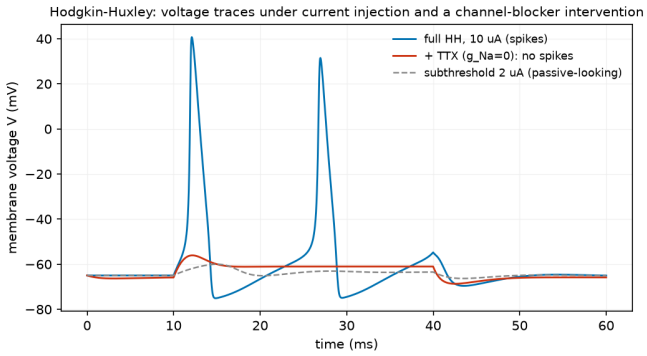


Figure 12: **Example spike traces from a single neuron under different conditions.** Membrane voltage under current injection: a supra-threshold step (10 μA) elicits overshooting action potentials (blue); the sodium blocker TTX (g_{Na}=0, a do on the mechanism) abolishes them (red); a sub-threshold current gives a passive response (grey).

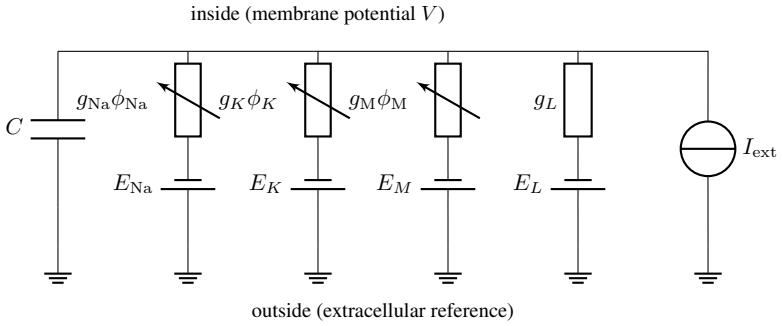


Figure 13: **The Hodgkin–Huxley equivalent circuit (Eq. (81)).** The membrane is a capacitor C; each ion channel is a branch with a variable conductance $g_c\phi_c$ (opening/closing gates ϕ_c) in series with a battery E_c (the reversal potential). The injected current I_{ext} charges the capacitor and 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:

$$C\frac{dV(t)}{dt} = I_{\mathrm{ext}}(t) \; - \; \sum_{c\in\mathcal{C}} I_c(t) \tag{81}$$

$$I_c(t) = g_c\phi_c(t)(V(t) - E_c) \tag{82}$$

$$\phi_c(t) = m_c^{p_c}(t)\; n_c^{q_c}(t)\; h_c^{r_c}(t) \tag{83}$$

$$\frac{dx_c(t)}{dt} = \frac{T_{x,c}^{\infty}(V(t)) - x_c(t)}{\tau_{x,c}(V(t))},\; x \in \{m, n, h\} \tag{84}$$

Here C is the capacitance, $V(t)$ is the voltage, $I_c(t)$ 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 3 for a list of gates, and Table 4 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

$$T_{x,c}^{\infty}(V) = \frac{\alpha_x(V)}{\alpha_x(V) + \beta_x(V)} \tag{85}$$

$$\tau_{x,c}(V) = \frac{1}{\alpha_x(V) + \beta_x(V)} \tag{86}$$

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 3: **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 4) carry their own gates m_c, n_c, h_c with the same form but different half-voltages and kinetics.

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 model

$$C\dot{V} = I_{\mathrm{ext}} - g_{\mathrm{Na}} m_{\mathrm{Na}}^3 h_{\mathrm{Na}} (V - E_{\mathrm{Na}}) - g_K n_K^4 (V - E_K) - g_L (V - E_L) \tag{87}$$

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.

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 ($N_{\mathrm{noise}} \to \infty$) 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. (86); the deterministic HH gating ODE is the $N_{\mathrm{noise}} \to \infty$ 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 $1/N_{\mathrm{noise}}$:

$$dx_c = \left[\alpha_x(V)(1-x_c) - \beta_x(V)\, x_c\right] dt + \sqrt{\frac{\alpha_x(V)(1-x_c) + \beta_x(V)\, x_c}{N_{\mathrm{noise}}}}\; dW_t, \qquad x \in \{m, n, h\}, \tag{88}$$

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 $N_{\mathrm{noise}} \to \infty$ recovers the deterministic gate. We integrate Eq. (88) by Euler–Maruyama and substitute the noisy gates into the membrane equation (81), 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. We sweep a *noise ladder* $N_{\mathrm{noise}} \in \{50, 100, 300, 1000, 3000\}$ from strongly stochastic to near-deterministic; unless noted we use N_{noise}=100 (fairly noisy), and $N_{\mathrm{noise}} \to \infty$ recovers the deterministic benchmark.

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).

D.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 5: five are *novel* mechanisms and the sixth is a recallable textbook M-current control. The five nonstandard

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 4: **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 identify available simulator interventions; the current headline acquisition menu does not include blockers.

mechanisms are intended to be difficult to distinguish using the two weak initial probes. This is not a claim of equivalence under every textbook stimulus or blocker. The evaluated agent selects among specified stimulation protocols, rather than inventing arbitrary protocols.

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

$$I_Z = g_Z \, m_Z^p \, h_Z^q \, (V - E_Z), \quad T_{m,Z}^{\infty}(V) = \sigma\Big(\tfrac{V - V_{1/2}^m}{k_m}\Big), \quad T_{h,Z}^{\infty}(V) = \sigma\Big(-\tfrac{V - V_{1/2}^h}{k_h}\Big), \tag{89}$$

with $\sigma(u) = 1/(1{+}e^{-u})$ the logistic (Boltzmann) sigmoid, half-voltages $V_{1/2}^m, V_{1/2}^h$, nonzero signed slopes k_m, k_h, 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 $V_{1/2}^h$ 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. (86), 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 G.3). It is also given the menu of stimulation protocols (Table 6) 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 $I_{\mathrm{ext}}(t) = a_t$. Rather than specifying an (open loop) policy mapping time step to current value, we instead let the agent choose one of 90 possible policies. The action library contains the nine protocols in Table 6, plus three 27-action grids: single pulses cross nine amplitudes $\{4, 6, 8, 11, 12, 14, 16, 18, 20\}$ μA with durations $\{60, 160, 300\}$ ms; conditioning–test protocols cross conditioning amplitude $\{-35, -20, +15\}$ μA, conditioning duration $\{60, 150, 270\}$ ms, and test amplitude $\{6, 12, 18\}$ μA; paired pulses independently choose the first and second amplitudes from $\{6, 12, 18\}$ μA and the zero-current gap from $\{30, 90, 180\}$ ms. These families probe excitability and adaptation, rebound and de-inactivation, and fatigue and recovery, respectively.

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 give a larger action set. Blockers

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 5: **The six worlds of NEURONBENCH**. Each current is added to a Na+K+leak spiker via Eq. (89). †the activation/inactivation columns are the tuples $(V_{1/2}, k, \tau, \text{power})$ and $(V_{1/2}, k, \tau)$ of Eq. (89). 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). The final column lists intended revealing protocols, not a proof that all other stimuli are non-discriminating. 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.

#	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 6: **Nine seed protocol types for NEURONBENCH.** The headline acquisition enumerates the expanded 90-action menu described above. 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 5.

remain available in the released benchmark and interactive app, but the experiment below searches only the 90 current-clamp protocols.

Experimental setup. Each run is seeded with a *fixed* passive set $\mathcal{D}_0$ of two conventional experiments: a brief $12\,\mu\mathrm{A}/40\,\mathrm{ms}$ step and a weak $5\,\mu\mathrm{A}/120\,\mathrm{ms}$ step. These probes avoid the conditioning and paired-pulse families designed to expose the novel currents. The stochastic MDA arm then selects eight further experiments using its model-disagreement/EIG surrogate. For acquisition, we average deterministic HH feature predictions over the weighted parameter particles and score between-model disagreement. Thus this surrogate has no stochastic-rollout Monte Carlo noise, but it approximates, rather than integrates, the SDE predictive mean. Likelihood evaluation and held-out SDE predictions still use stochastic dynamics. For the headline comparison, all four model-based arms and ICL (+MDA design) replay the archived stochastic MDA histories. Only ICL (+LLM design) selects its own eight actions. The earlier controlled inference ladder in Fig. 19 also uses shared histories. Each observed trajectory contains exactly 20 sampled voltage values, stratified over time with endpoints retained; the dense simulator grid is not available as training data. Process noise uses $N_{\mathrm{noise}} = 100$ channels and observation noise has SD 2 mV. Oracle and filtering integration steps are 0.025 and 0.05 ms, respectively.

Evaluation. 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 (which is very hard, as shown in Fig. 4b), 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. 14. These are computed as follows:[12]

$$F(y) = \Big(\underbrace{n_{\text{test}}}_{\text{test spikes}}, \underbrace{n_{\text{pre}}}_{\text{pre-pulse spikes}}, \underbrace{n_{\text{pre}} - n_{\text{test}}}_{\text{run-down}}, \underbrace{n_{\text{test}}^{\text{early}} - n_{\text{test}}^{\text{late}}}_{\text{adaptation}}, \underbrace{\min_t V(t)}_{V_{\min}}, \underbrace{\bar{V}_{\text{end}}}_{\text{steady state}} \Big), \tag{90}$$

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.

We compute this target using a fixed *held-out query set* $\mathcal{Q}$ of 24 designs, with eight each from the single-pulse, conditioning–test, and paired-pulse families. The queries have no exact overlap with the 90-action design menu, are shared across worlds, and are hidden from the agents (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 candidate mechanisms may disagree. For this benchmark, Eq. (2) uses $t_c = 0$, six features, 24 queries, one entry per feature per query, and fixed feature scales:

$$L = \frac{1}{24 \cdot 6} \sum_{q=1}^{24} \sum_{i=1}^{6} \left(\frac{\widehat{\mathbb{E}}[F_i(Y) \mid \chi_q, \mathcal{D}] - \widehat{\mathbb{E}}_\star[F_i(Y) \mid \chi_q]}{s_i} \right)^2, \qquad s = (3, 3, 3, 3, 12, 8). \tag{91}$$

The first four scales apply to spike-count differences and the last two to mV. The scaling makes errors in counts and voltages commensurate: an error of three spikes, 12 mV in the minimum voltage, or 8 mV in the tail mean each contributes one before averaging over features and queries. The unequal voltage scales assign $(12/8)^2 = 2.25$ times more weight per squared mV to the sustained tail level than to the single minimum. These are fixed benchmark weighting conventions, not physiological constants or estimated noise standard deviations; the precise values are not uniquely determined. The count-derived features overlap, so the metric deliberately gives more weight to spike-count behavior; it is not a covariance-whitened distance. The oracle expectation is estimated using independent stochastic rollouts; this is error in expected features, not squared error against one noisy path.

Distributional feature scores. A complementary diagnostic is CRPS for each of the six marginal feature distributions. If $f_i^{(r)} = F_i(Y^{(r)})$ are R independent posterior-predictive draws and f_i^{obs} is the feature of an independent held-out rollout, an unbiased Monte Carlo estimate is

$$\widehat{\text{CRPS}}_i = \frac{1}{R} \sum_{r=1}^{R} |f_i^{(r)} - f_i^{\text{obs}}| - \frac{1}{2R(R-1)} \sum_{r \neq r'} |f_i^{(r)} - f_i^{(r')}|. \tag{92}$$

We can average this over held-out rollouts and queries and report each feature separately; a dimensionless aggregate uses $\frac{1}{6}\sum_i \text{CRPS}_i / s_i$ (not s_i^2). This tests uncertainty about counts and voltage summaries without requiring prediction of precise spike times. Unlike MSE of expected features, it must be scored against realized held-out features, not their oracle mean.

Baseline. As a baseline, we use a transductive ICL which computes $\hat{F}(\chi)$ by passing in the observed data $\mathcal{D} = \{(\chi_i, \{(t_{ij}, y_{ij})\}_{j=1}^{20})\}$ into the prompt, along with the novel test input χ, and asking the LLM to predict the resulting features. ICL (+MDA design) replays the earlier archived MDA histories, not the new online-expansion histories (i.e., it does not choose its own experiments), but ICL (+LLM design) uses an LLM to design the experiments.

Interactive app. Figure 15 shows a screenshot for a web app we built that lets users try this benchmark for themselves. (URL available after publication.)

[12] Run-down shares information with the pre/post spike counts, so these six components are not independent.

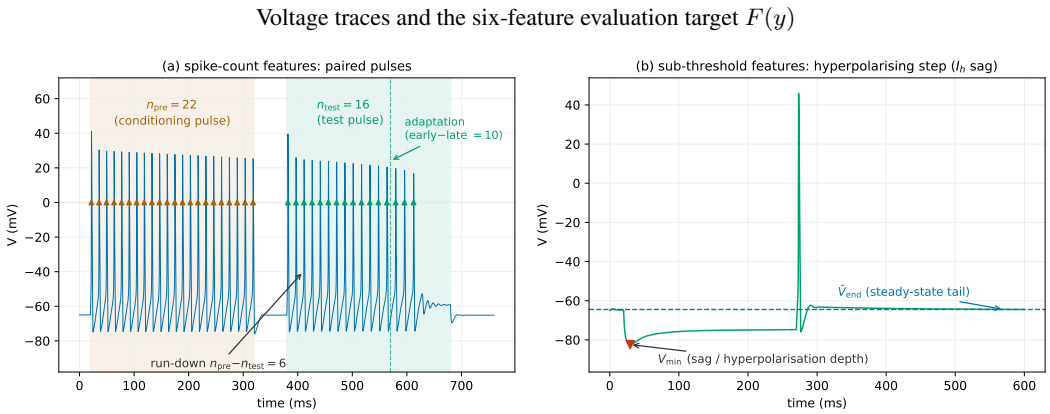


Figure 14: **From a raw voltage trace to the per-trace feature vector $F(y)$ of Eq. (90)**, computed on real NEURONBENCH traces. *(a)* On a paired-pulse protocol the spike-count features are the test- and pre-pulse counts (n_{test}, n_{pre}; upward 0 mV crossings, triangles), their use-dependent *run-down* $n_{\text{pre}} - n_{\text{test}}$ (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 $\bar{V}_{\text{end}}$.

Quantity	Symbol	Prior / value
Na^+ and K^+ conductances	$g_{\text{Na}}, g_{\text{K}}$	$120, 36$ (fixed)
capacitance	C	$1.0\,\mu$F/cm^2 (fixed)
leak conductance	g_{L}	$\mathcal{U}(0.2, 0.45)$
I_h	$g, V_{1/2}, \tau$	$\mathcal{U}(2, 10), \mathcal{U}(-105, -85), \mathcal{U}(60, 240)$
T-type Ca^{2+}	$g, V_{1/2}, \tau_h$	$\mathcal{U}(1.5, 8), \mathcal{U}(-65, -45), \mathcal{U}(12, 180)$
D-type K^+	$g, V_{1/2}, \tau_h$	$\mathcal{U}(4, 14), \mathcal{U}(-40, -20), \mathcal{U}(100, 300)$
M-type K^+	$g, V_{1/2}, \tau$	$\mathcal{U}(1, 6), \mathcal{U}(-45, -25), \mathcal{U}(30, 90)$
slow Na^+ inactivation	τ_s	$\mathcal{U}(200, 600)$

Table 7: **Named-archetype reference parameter ranges.** These describe the earlier fixed-archetype setup, not the LLM-declared headline priors. Conductances are in mS/cm^2, voltages in mV, and time constants in ms. Kinetic functional forms, reversal potentials, and the canonical excitable backbone are fixed by the archetype; the extra mechanism's parameters and shared leak are inferred.

D.3 MDA

In this section, we describe how MDA is applied to this domain.

Models. The agent is told that the cell is a standard Na+K+leak spiker (the familiar null model) that may additionally carry "an additional membrane current of unknown identity, voltage-dependence, and kinetics" — an *opaque* prior that does not name or parameterise the novel current. The LLM must propose suitable mechanisms, which are then simulated by the SDE sampler.

Priors. In the headline open-library runs, the LLM declares uniform bounds for conductance, half-activation voltage, and activation time constant (or just the slow-Na time constant). The backend adds shared leak $g_{\text{L}} \sim \mathcal{U}(0.2, 0.45)$ and, for transient extra currents, $\tau_h \sim \mathcal{U}(20, 300)$ ms. Reversal potential and activation direction are proposed structural choices; activation slopes are fixed to 8 mV (depolarizing) or -6 mV (hyperpolarizing). Inactivation half-voltage is the midpoint of the proposed activation bounds minus 22 mV, with slope 5 mV. These data-dependent priors differ from the named-archetype reference ranges in Table 7. Channel-noise scale and observation noise are fixed. See Section G.3 for the current proposer interface.

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

$$p(y_{1:T} \mid m, \theta) = \int p(y_{1:T} \mid z_{0:T}, m, \theta)\, p(z_{0:T} \mid m, \theta)\, dz_{0:T}, \tag{93}$$



Figure 15: **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 $a_{1:T}$ 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.

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).

Particle filtering. The agent fits a stochastic state-space model (4) where the latent state $z_t = (V_t, \{x_c(t)\})$ (voltage and gates) evolves by the discretised Fox–Lu transition $p(z_t \mid z_{t-1}, \chi)$ of Eqs. (81) and (88), 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 $p(z_t \mid z_{t-1}, \chi)$ 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 (successive 0.05-ms Euler–Maruyama substeps with Gaussian gate noise, Eq. (88)) 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 $p(y_{1:T} \mid m, \theta)$ is estimated; parameter integration is a separate outer calculation. So the intractable-likelihood regime needs only a *simulator* of the latents plus an evaluable observation model.

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. (23)), 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 5. The noisy voltage traces from the two hypotheses are shown in Fig. 16 — the I_h sag is subtle relative to the channel noise, so they overlap. In Fig. 17 we plot the fixed-parameter log-likelihood gap, $\log p(y \mid m_{I_h}, \theta_{I_h}) - \log p(y \mid m_{\text{plain}}, \theta_{\text{plain}})$, vs noise level N_{noise}. Parameters are fixed in this illustrative diagnostic; it does not integrate parameter uncertainty. We

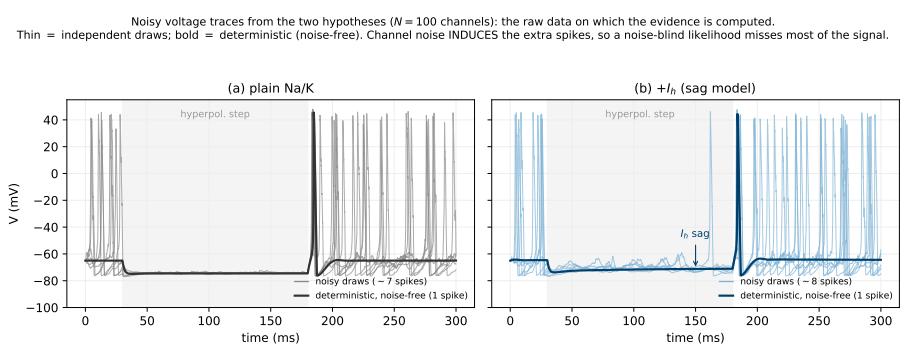


Figure 16: **Stochastic-latent NEURONBENCH: the raw data.** Noisy voltage traces from the two competing hypotheses under a moderate hyperpolarising-step protocol, at N_{noise}=100 channels (thin: independent draws; bold: the *deterministic*, noise-free trace). *(a)* A plain Na/K cell. *(b)* The same cell plus a hyperpolarisation-activated I_h current (the H-SAG model of Table 5), 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. 17.

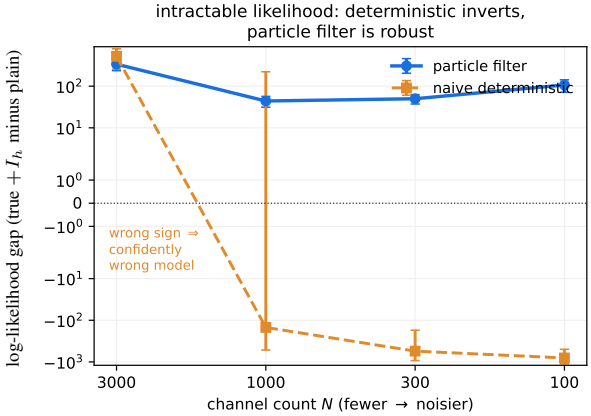


Figure 17: **Stochastic-latent NEURONBENCH: estimating the intractable likelihood by simulation.** We score the I_h-vs-plain decision on the data of Fig. 16, sweeping the channel count N_{noise} (fewer = noisier): the fixed-parameter log-likelihood gap vs. N_{noise}. A likelihood that ignores the process 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.

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.

D.4 RESULTS

Sample efficiency. The main sample efficiency plot is shown in Fig. 4a. MDA is given two initial trajectories in $\mathcal{D}_0$, and then creates an initial hypothesis library using the LLM proposal. We used particle filtering to approximate the likelihood (marginalizing over stochastic trajectories), an initial tempered-SMC parameter fit followed by recursive likelihood-weight updates on retained parameter particles, and evidence weights over the archive. The new controlled cohort uses six worlds and three seeds, with $N_p = 128$, $N_z = 128$, and 16 initialization tempering steps. Subsequent updates retain parameter particles without rejuvenation or full refitting. All four model-based arms share the initial archive and ten observed trajectories; later model births are disabled. Prediction uses two independent 128-draw samples from the full joint model–parameter mixture in Eq. (5), without top-component truncation. Both ICL baselines are restricted to the same worlds and seeds.

The factorial crosses ODE/SDE dynamics with recursive SMC/direct optimization+BIC. ODE likelihoods are Gaussian; SDE likelihoods use a PF. BIC optimization uses three rounds of 24 candidates and independently validates four finalists with three likelihood estimates each. All arms share the structural prior $\log p(m) = -2\dim(\theta_m) + \text{const}$. SDE/ODE paired area-under-learning-curve ratios are 0.288 under SMC and 0.195 under BIC, favoring SDE in all six worlds. The SDE BIC/SMC ratio is 1.106 (95% hierarchical-bootstrap interval $[0.972, 1.354]$), so this study does not establish a difference between those inference stacks. At ten trajectories, the MAP-model parameter population has only one distinct particle in 13 of 18 ODE-SMC runs, versus none of the SDE-SMC runs. Thus particle impoverishment may contribute to ODE-SMC deterioration; directly optimized ODE+BIC also performs poorly. Median worker runtimes are 9.1, 9.2, 11.2, and 18.6 minutes for ODE-SMC, SDE-SMC, ODE-BIC, and SDE-BIC, respectively, including predictive evaluation.

CRPS. Figure 18 shows CRPS scores for all 36 final online $\mathcal{M}$-open posteriors, at ten total experiments, using a stochastic forecast vs just predicting the mean. Mean standardized CRPS is 0.687, versus 1.049 for a point distribution at the same predictive mean. Retaining predictive spread improves the score for all six features in aggregate. This diagnostic does not isolate model or parameter uncertainty, nor compare independently fitted inference arms.

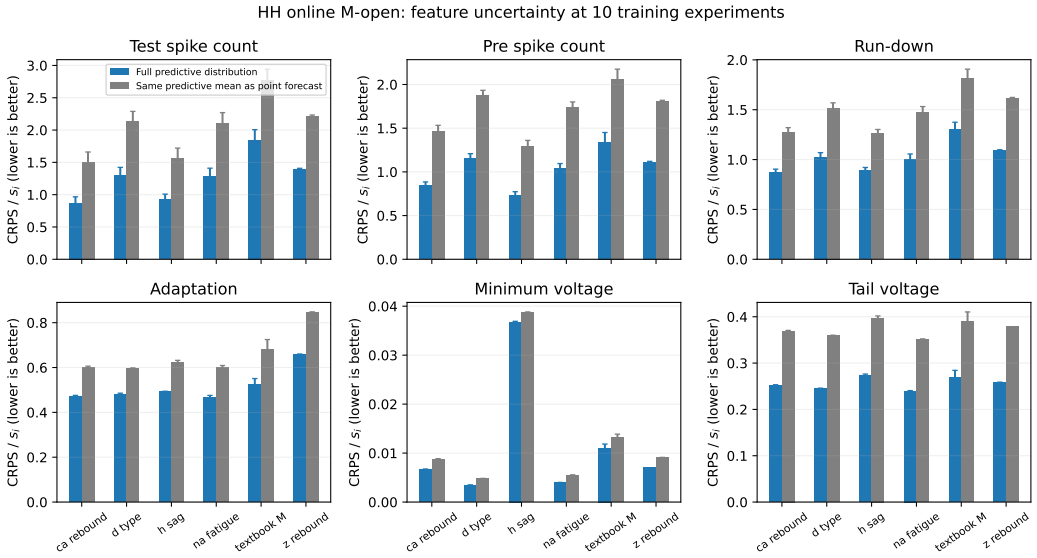


Figure 18: **HH marginal feature CRPS at ten experiments.** Six worlds, six seeds each, and 24 held-out queries per cell. Blue: samples from the full model–parameter posterior followed by stochastic simulation. Gray: a point distribution at the same estimated predictive mean, whose CRPS equals absolute error. Each cell averages two 128-draw predictive repeats against independent oracle rollout features; repeats share oracle draws. Scores are divided by the fixed s_i of Eq. (91). Bars average seeds; error bars are ± 1 seed-level s.e. within each world. Features are extracted from simulated signal voltage before adding measurement noise, matching the existing feature target. These are final-budget distributional diagnostics, not learning curves or an ablation of parameter integration.

Controlled inference ablations. Our main method uses the SDE model, but also uses PF for estimating the likelihood and tempered SMC for estimating the evidence. We wanted to determine how important each of these components were, so we did a controlled ablation, where we retain SDE+PF but replace parameter integration with optimization and use BIC model weights, keeping archives, designs, and observations fixed. We also optimize Bayesian Synthetic Likelihood (Section A.5.2) and use BIC. The results are shown in Fig. 19. The optimization+BIC configurations have slightly lower point estimates, but neither paired contrast is statistically resolved.

To quantify this, we consider the area under the learning curve (AULC), which we define as

$$\text{AULC}_{a,w,s} = \int_{b_{\min}}^{b_{\max}} L_{a,w,s}(b)\, \mathrm{d}b. \tag{94}$$

Latent dynamics	Likelihood	Parameter treatment	Model evidence	R
SDE	bootstrap PF	initial SMC + recursive updates	SMC normalizer	1.000
SDE	bootstrap PF	bounded optimization	BIC	0.894
SDE	Gaussian BSL	bounded optimization	BIC	0.897
ODE	Gaussian residual	bounded optimization	BIC	2.974

Table 8: **Controlled inference ladder.** Every row replays the same frozen archives, actions, and realized data; hence this table isolates inference and prediction rather than comparing independently selected acquisition policies. R is the geometric learning-curve nMSE ratio in Eq. (95), relative to SDE+PF+tempered SMC (lower is better). The PF and BSL plug-in rows are standalone estimators: each obtains $\hat{\theta}_m$ by bounded optimization, independently validates the finalists, and then applies BIC. The ODE row is the matched deterministic replay from the controlled ladder.

where $L_{a,w,s}(b) > 0$ denotes the held-out loss of method a at the displayed total-experiment budget b, in world w and replicate s. When comparing methods a and c, we report the geometric learning-curve ratio

$$R_{a/c} = \exp\left\{ \mathbb{E}_{w,s}\left[\frac{1}{b_{\max} - b_{\min}} \int_{b_{\min}}^{b_{\max}} \log \frac{L_{a,w,s}(b)}{L_{c,w,s}(b)} \, \mathrm{d}b \right] \right\}, \tag{95}$$

using trapezoidal integration over the displayed budgets, averaging replicates on the log scale within each world and then averaging worlds. This geometric learning-curve loss ratio is not generally the ratio of the ordinary AULCs in Eq. (94); it averages log loss ratios before exponentiation. Thus $R_{a/c} < 1$ favors a; for example, $R_{a/c} = 0.80$ means that a has 20% lower error than c on the multiplicative learning-curve summary. The results are shown in Table 8, and show once again that all SDE methods are much better than ODE, while the stochastic inference configurations have similar predictive accuracy. This is not a direct measurement of evidence-estimation error.

Optimizing the stochastic likelihood. For the PF+optimization+BIC arm, we use

$$\mathrm{BIC}_m^{\mathrm{PF}} = -2 \log \widehat{p}_{\mathrm{PF}}(\mathcal{D} \mid m, \hat{\theta}_m) + C_m \log N, \qquad p(m \mid \mathcal{D}) \approx \frac{\exp(-\mathrm{BIC}_m^{\mathrm{PF}}/2)}{\sum_{m'} \exp(-\mathrm{BIC}_{m'}^{\mathrm{PF}}/2)}. \tag{96}$$

Here C_m counts fitted parameters, the model prior is uniform, and N is 20 times the number of observed trajectories. The PF integrates stochastic latent trajectories with fixed 2-mV observation noise; no additional residual-scale multiplier is fitted. To obtain $\hat{\theta}_m$, we use bounded derivative-free cross-entropy search: three batches of 24 parameter candidates, retaining the top quarter to adapt the sampling distribution. Search evaluations use 64 PF particles and common random numbers within each batch to reduce noise in candidate comparisons. The four best candidates are then rescored using three independent 128-particle PF evaluations each. We average these likelihood estimates (using log-mean-exp numerically), select the best finalist, and use its validated score in Eq. (96). This stabilizes the noisy search without claiming to find the exact likelihood maximum. BSL gave a smoother optimization objective in our pilot. In the matched optimization sweep, BSL used 1.71 aggregate A10G-hours versus 2.97 for PF, i.e., about 42% less GPU time. (These are aggregate compute times, not elapsed latency.) The speedup is limited because BSL still needs to sample trajectories to compute $\mu_{m,\theta}$ and $\Sigma_{m,\theta}$. True speedups would require avoiding these rollouts and the use of amortized inference methods, such as neural likelihood estimation, neural posterior estimation, or neural ratio estimation; we leave this to future work.

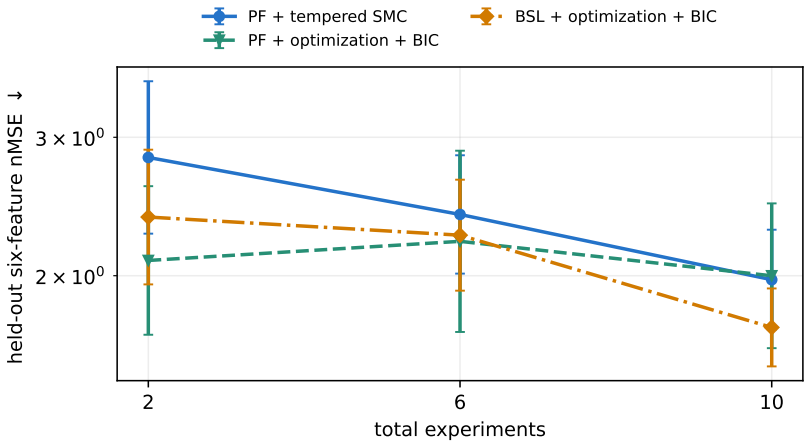


Figure 19: **Standalone stochastic likelihood optimization on NEURONBENCH.** All methods receive the same two initial trajectories, truth-blind frozen archives, eight EIG-selected actions and realized observations. PF+tempered SMC supplies the reference curve (initial fit followed by recursive updates). The other arms independently optimize either the particle-filter likelihood or a Gaussian synthetic likelihood, then use BIC model weights; stochastic posterior-predictive simulation is retained in both. Points show total budgets $2, 6, 10$ and are geometric means over 36 paired world–seed units, with seeds averaged on the log scale within each of six worlds; bars are ± 1 world-level s.e. Relative to PF+tempered SMC, the geometric learning-curve ratios are 0.894 for PF+optimization+BIC and 0.897 for BSL+optimization+BIC; neither paired six-world contrast is statistically resolved.

E BOXINGGYM

E.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. We focus on predictive loss to match the rest of the paper. We found the "explain-to-a-novice" metric gave very high variance results. Thus we just report predictive loss (under novel perturbations), to be consistent with the rest of this paper.

Note: Unlike the main paper, all results in this section use Opus 4.7 (with reasoning turned off, which is its default setting). Comparisons within this auxiliary study all share that LLM, so we can compare the relative performance of methods. However we make no claims about absolute top performance, or the relative performance of LLMs across domains.

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 from this auxiliary study. This leaves us with the seven domains shown in Table 9, which we partition into two clusters.

The first cluster consists of covariate-regression models, where the agent needs to learn the form of the function that defines the mean of the (scalar) output, analogous to learning the symbolic rate law in the chemistry domain. Since there are no latent variables, the GLM likelihood is fast to compute.

The second cluster consists of latent variable models, where the number of parameters can grow with the size of the data. To represent these models compactly, we use a probabilistic programming language (PPL), as we explain in Section E.4.

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. 27 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.

Table 9: 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 χ
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	$p(y\|x,\theta) = \mathcal{N}(\mu = b + \sum_{k=1}^{K} \frac{\alpha}{c+\|x-\theta_k\|^2}, \sigma^2)$	$x \in \mathbb{R}^2$

Baselines. As the main baseline, 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 data so far, fits $\hat{m}$, injects $\hat{m}$ (with a residual critique) into the LLM's context,

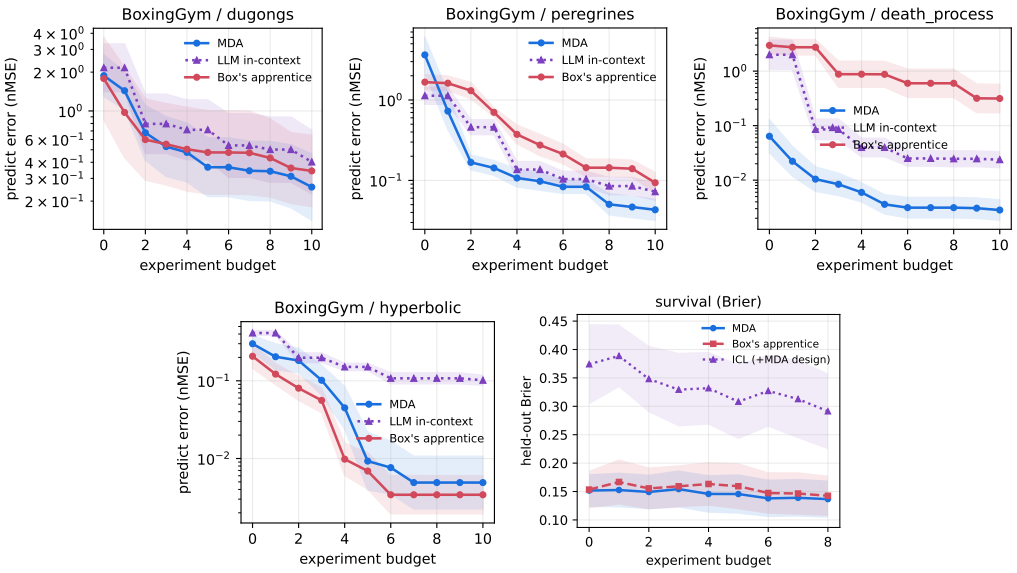


Figure 20: **BOXINGGYM learning curves for the five covariate-regression domains.** MDA, Box's Apprentice, and the model-free ICL (+MDA design) baseline (mean ± SE; held-out nMSE against the noise-free mean $\mathbb{E}[Y \mid x]$, except SURVIVAL, whose Bernoulli outcome is scored by the proper Brier score).

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. As another baseline, we use ICL (reusing MDA's data/ design decisions).

E.2 PRIORS AND LIKELIHOODS

For the GLM domains, each candidate is a mean function $\mu_\theta(x)$. The likelihood is tractable in closed form — Gaussian with a known scale σ for the real-valued domains, or the domain's specified count likelihood or Bernoulli likelihood for binary outcomes — 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 particle tempered SMC run per model, with N_p=1500 particles (which are cheap to compute due to the closed form likelihood). For the latent variable models, we group the latents together with the fixed parameters, and use tempered SMC to compute $p(\mathcal{D}|m) = \int p(\mathcal{D}|m, z, \theta)p(z, \theta|m)dzd\theta$, rather than nesting PF inside of SMC.

E.3 RESULTS ON GLMS

The results on the 5 GLM domains are shown in Fig. 20. 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, consistent with averaging over models rather than using a single MAP estimate; acquisition and model evolution also differ, so this is not an isolated averaging ablation. (Both average over their parameters.)

E.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; our implementation uses NUMPYRO. This is an implementation choice, not a controlled comparison of the two probabilistic-programming systems. The LLM proposes each candidate as a NUMPYRO program — a `def model(ctx)` that declares the priors over latents (z, θ) with `numpyro.sample` (a `numpyro.plate` for the per-unit latents) and registers the expected observable at every candi-

```
# 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 21: **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. (97) — so there are no fixed parameters, and because the latent scale is 1 the induced $P(\text{correct})$ sits near $\frac{1}{2}$.

date 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

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

These evidences determine normalized weights over the finite candidate pool.

E.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

$$\begin{aligned}
\text{Student}: &\quad p_{sq} = \sigma\big(\theta_s^{\text{abil}}\big) \\
\text{1PL (Rasch)}: &\quad p_{sq} = \sigma\big(\theta_s^{\text{abil}} - \theta_q^{\text{diff}}\big) && \text{(Rasch, 1960)} \\
\text{2PL}: &\quad p_{sq} = \sigma\big(\theta_q^{\text{disc}}\,(\theta_s^{\text{abil}} - \theta_q^{\text{diff}})\big) && \text{(Birnbaum, 1968)} \\
\text{3PL}: &\quad p_{sq} = \theta_q^{\text{guess}} + (1-\theta_q^{\text{guess}})\,\sigma\big(\theta_q^{\text{disc}}(\theta_s^{\text{abil}} - \theta_q^{\text{diff}})\big) && \text{(Birnbaum, 1968)} \\
\text{MIRT}: &\quad p_{sq} = \sigma\big(\theta_q^{\text{disc}\top}\theta_s^{\text{abil}} - \theta_q^{\text{diff}}\big),\ \ \theta_s^{\text{abil}} \in \mathbb{R}^D && \text{(Reckase, 2009)}
\end{aligned} \tag{97}$$

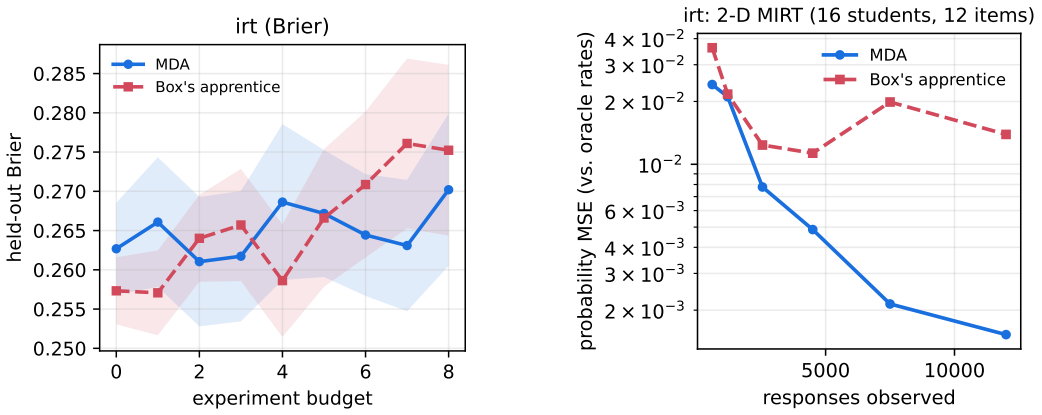
The simplest model just has one ability parameter per student; 1PL (one-parameter logistic, the *Rasch* model) has student ability and question difficulty; 2PL adds a per-item *discrimination* θ_q^{disc} (how sharply the item separates abilities); 3PL adds a per-item *guessing* floor θ_q^{guess}; multidimensional IRT (MIRT) makes ability a D-vector. The simulator's 2PL latents have $\mathcal{N}(0,1)$ priors; candidate programs specify their own priors (Fig. 24). These latents are marginalized out, so there are no fixed parameters in the model. We use conventional IRT θ notation here; the z labels in Fig. 25 denote the same per-unit latent variables. The rungs differ only in *which latents exist*, and hence in their complexity (expressive power).

BOXINGGYM's ground truth is 2PL (see Fig. 21 for their code). The resulting predictive distribution has the form

$$p(Y_{sq}{=}1|m) = \int \text{Ber}\Big(1 \,\Big|\, \sigma\big(\theta_q^{\text{disc}}(\theta_s^{\text{abil}} - \theta_q^{\text{diff}})\big)\Big)\, p(\theta_s^{\text{abil}})\, p(\theta_q^{\text{diff}})\, p(\theta_q^{\text{disc}})\ \mathrm{d}\theta_s^{\text{abil}}\, \mathrm{d}\theta_q^{\text{diff}}\, \mathrm{d}\theta_q^{\text{disc}}. \tag{98}$$

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 22a 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. The domain is very small: there are only 6 students and 6 questions, and only 8 of 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. 24), but there is not enough evidence to move away



(a) default 6×6 IRT (near-chance)

(b) higher-signal 2-D MIRT IRT

Figure 22: **Probabilistic-program comparisons on IRT, at two signal levels.** (a): the default 6×6 BOXINGGYM instance uses LLM-authored NUMPYRO programs. Its data are weakly informative (Fig. 21), 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. 25: 16 students × 12 items with a hidden second ability axis), the probability MSE against the true response rates falls as the response matrix fills. This controlled illustration compares a fixed five-model ladder against a single evolving LLM program; both use numerical Brier-risk design. Scores cover all cells, not just unobserved ones.

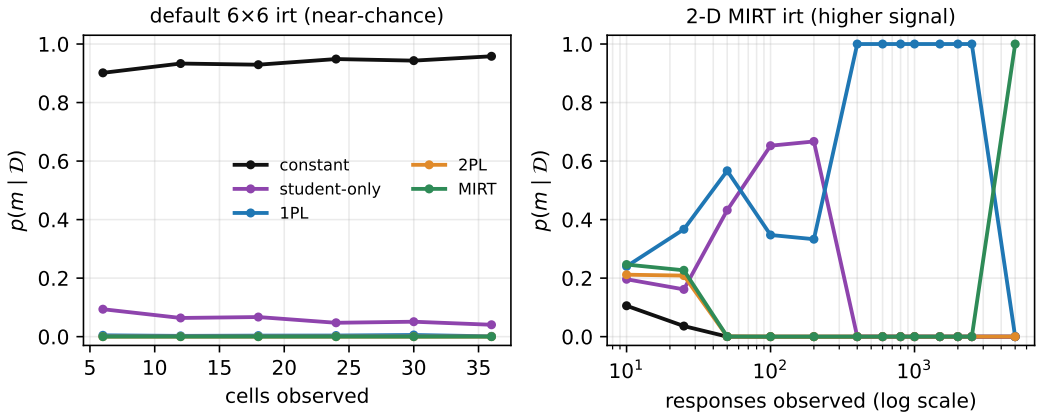


Figure 23: **Posterior over fixed model ladders for the two IRT domains.** These controlled inference illustrations do not generate new structures. *Left:* low-signal BOXINGGYM regime — with few experiments / little data, MDA keeps most of its mass on a simple constant model. *Right:* high-signal regime — as the data grows, MDA climbs the complexity ladder, eventually converging on the true 2-D MIRT model.

from a constant predictor[13], as shown in Fig. 23 (left). This is an example of the Bayesian Occam's razor penalizing overly complex models. 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 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 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

[13] At a global base rate $\mu \approx 0.667$, the constant-predictor risk is $\mu(1-\mu) \approx 0.22$. The Bayes floor is instead $\mathbb{E}_i[p_i(1-p_i)]$, which is smaller when conditional rates vary. The observed risk near 0.27 is above the constant-predictor reference; small samples and nonrepresentative acquisition are possible contributors, not a demonstrated causal explanation.

```
# 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 24: **The IRT candidate programs, as *verbatim* LLM-generated NumPyro code.** Each is one rung of the ladder Eq. (97): 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. (97)) across proposals.

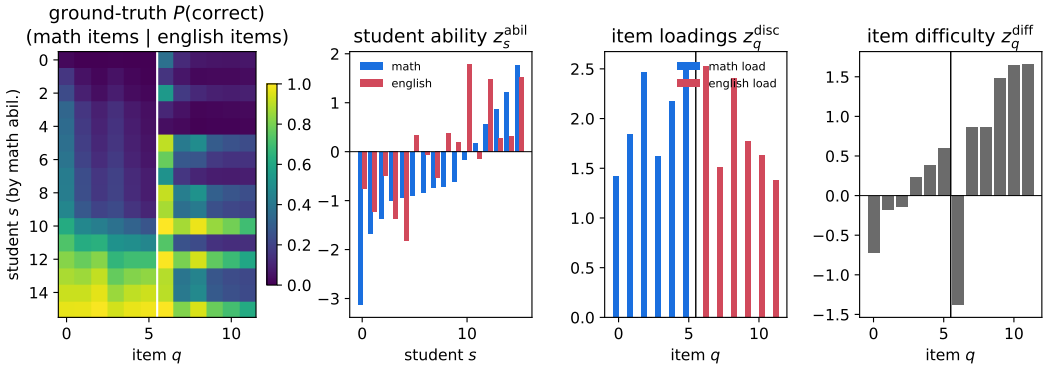


Figure 25: **Ground truth for the stronger-signal MIRT IRT instance.** Left: the ground-truth $P(\text{correct})$ response grid (16 students $\times$ 12 items, sorted by math ability; the vertical rule splits the math and english item blocks). Right: the generative latents — a *two*-dimensional student 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.

Fig. 25. 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 k, the agent gets to pick N_k 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 $N_k(s, q)$, of which $S_k(s, q)$ are successful, which the agent models using a Binomial likelihood. Evaluation measures squared error against the oracle probabilities over all 192 cells, including queried cells; it is not unseen-cell generalization.

```python
# 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 26: **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. (99)). The `deterministic('mu', ...)` node is the mean observable at every candidate probe.

In Figure 22b, we let the agent pick $N_k = 150$ cells per round, chosen freely from all 192 cells. This controlled PPL illustration uses a fixed five-model ladder for MDA (constant, student-only, 1PL, 2PL, MIRT), and a single evolving LLM-written program for Apprentice. Both use numerical expected Brier-risk reduction to allocate responses; this is distinct from the headline EIG surrogate. MDA has lower probability error in this illustration; no significance claim is made from the single synthetic instance. In Fig. 23 (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.

E.6 RESULTS ON LOCATION

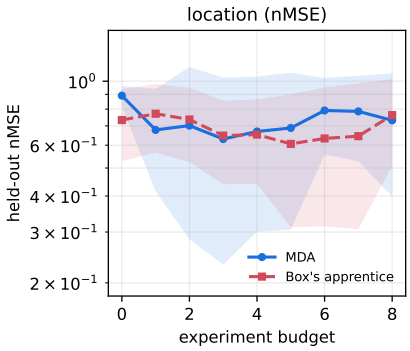
The true predictive model for the LOCATION (source finding) problem, at location $x \in \mathbb{R}^2$, is

$$p(Y_x \mid m) = \int \mathcal{N}\Big(Y_x \,\Big|\, b + \sum_{k=1}^{K} \frac{\alpha}{c + \|x - \theta_k\|^2},\ \sigma^2\Big) \left[\prod_{k=1}^{K} p(\theta_k)\right] p(\theta_b, \theta_c, \theta_\alpha, \theta_\sigma) d\theta \tag{99}$$

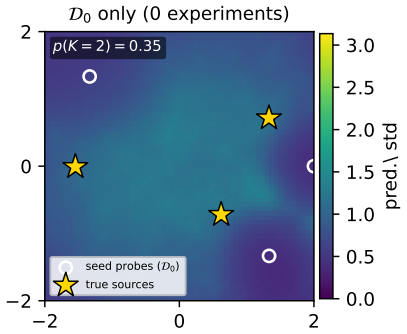
with latent source positions $\theta_k \sim \mathcal{N}(0, I_2)$ and fixed globals (b, c, α, σ) all marginalized out. The model structure m corresponds to the number of sources K. The agent-generated code for each of these models is shown in Fig. 26.

The performance of MDA and Apprentice on this domain are shown in Fig. 27a. 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).

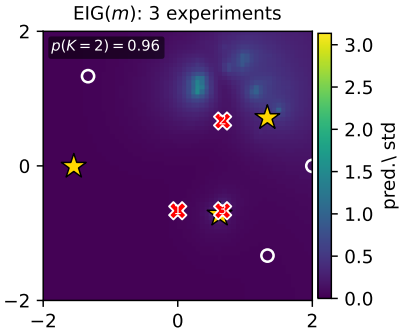
We also tried maximizing the EIG for the model and its parameters (Eq. (63)) — which would additionally pin down the source locations θ_k — but we found that the greedy variance objective is too myopic. In particular, because the signal field is near-singular next to a source, it piles probes onto a single hypothesized source (where a few particles' predictions blow up) rather than spreading them, giving unstable results worse than random design.



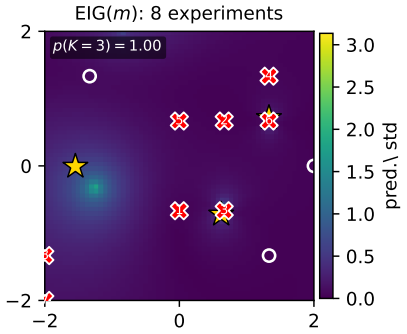
(a) data efficiency



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



(c) 3 EIG experiments



(d) 8 EIG experiments

Figure 27: **Code-MDA on LOCATION (source finding over LLM-authored NUMPYRO programs).** (a): held-out nMSE vs. budget (20 seeds, median ± 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 red × the probes EIG chooses, 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.

In summary, on this domain active design gives no advantage: the EIG policy and random sampling perform equally well. (Fig. 27(c,d) illustrate where the EIG policy places its probes; the advantage it would buy on a more identifiable field simply vanishes here.)

F FURTHER RELATED WORK

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

F.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 ACTIVESCIBENCH-CHEM (Kabra et al., 2026), and BOXINGGYM (Gandhi et al., 2025). We discuss some other benchmarks below.

NewtonBench. NEWTONBENCH (Zheng et al., 2026) contains 324 interactive law-discovery tasks across 12 areas of physics. It generates memorization-resistant problems by systematically modifying canonical laws (while preserving dimensional consistency), and varies both the complexity of the hidden equation and the complexity of the larger system in which it is embedded. The agent chooses experiments and must recover the hidden equation, with success assessed primarily by symbolic equivalence. This complements our emphasis on predictive accuracy under new interventions, including partially observed latent dynamics.

Science-Gym. SCIENCE-GYM (Cerrato et al., 2026) is a Gym-compatible testbed in which an agent collects experiments and then seeks an interpretable equation. Its seven environments cover the lever, projectile motion, an inclined plane, Lagrange points, the brachistochrone, an epidemiological SIRV model, and droplet friction; the interface can vary the observations and amount of reward supervision. Its reported "Threshold-and-Save" baseline uses an RL policy to find high-reward episodes and applies guided symbolic regression to the retained tabular data. Our experiments instead score forecasts on held-out interventions, without a discovery-directed reward.

SciGym. SCIGYM is 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 underlying models are ODEs whose laws must be discovered.

ARC-AGI-3. ARC-AGI-3 (ARC Prize Foundation, 2026) evaluates interactive exploration and goal-directed action in unfamiliar visual environments. Unlike the earlier static ARC input/output tasks, it is not simply transductive prediction at a supplied test input. MDA instead evaluates scientific-model forecasts on held-out experimental conditions, with numerical inference in an explicit mechanistic model class.

ZendoWorld. (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.

Other benchmarks. LLM-SRBENCH (Shojaee et al., 2025) targets equation discovery with problems designed to reduce memorization, while AUTO-BENCH (Chen et al., 2025) uses causal-graph-based interactive environments.

F.2 OTHER "AI-SCIENTIST" SYSTEMS

A fast-growing line of work builds LLM *agents* that automate parts — or all — of the scientific workflow. The literature-grounded systems below contrast with MDA along two useful 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 approximate numerical weights 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 hypotheses 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), distinct from Gottweis et al.'s Co-Scientist described above, 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.

Numerical equation discovery. SINDy (Brunton et al., 2016a) fits sparse combinations of candidate nonlinear functions to measured dynamics; SINDYc (Brunton et al., 2016b) includes control inputs. It is a relevant non-LLM comparator, but applying the basic formulation to our partially observed systems requires choosing observable or delay-embedded coordinates and handling noise and missing observations, rather than providing the hidden simulator states. Our glucose comparison includes an observation-space SINDYc control with training-only library selection and stability screening (Section C.3); we do not yet evaluate a corresponding sparse-dynamics baseline for HH. The PySR system (Cranmer, 2023) uses evolutionary structure search and coefficient fitting for symbolic regression of static equations; it is used internally by LLM-AUTOSCILAB, so we already indirectly compare to it in Fig. 2a.

Active theory learning. ATLAS (Elteto et al., 2026) alternates fitting an ensemble of interpretable DisRNNs and optimizing experiments to distinguish their predictions. Its evaluation recovers Q-learning and actor–critic agents from bandit behavior; it also examines GRU ensembles. MDA shares the population-based active-learning principle but searches explicit non-neural algebraic, compartmental, and stochastic channel models across domains. The different interfaces prevent a plug-in baseline comparison.

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) Numerical design:* our implemented acquisition is also a model-discrimination EIG surrogate (Eq. (59)); task-oriented VoI is a future direction, not a distinguishing empirical claim. *(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 chemistry, glucose dynamics, electrophysiology, and the BOXINGGYM suite.

Amortized design. ASIG (Hartmann et al., 2026) trains LLM information-gathering policies with an EIG reward. This complements our online numerical acquisition: it amortizes the design policy rather than fitting executable scientific models at each experiment.

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 archive of *executable models* (rate laws, compartment ODEs, channel mechanisms) with numerically fitted parameters, and evaluates held-out predictive accuracy.

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: fitting executable predictive models and selecting informative interventions, using numerical likelihood and evidence approximations 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 an explicit mechanistic 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; numerical inference scores models and guides design*.

G LLM PROMPTS

This appendix documents agent interfaces and prompt templates. Historical examples are marked explicitly and are not the exact prompts of every current run; archived per-call transcripts are the source of record. 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. ICL (+MDA design) forecasts from MDA's history; ICL (+LLM design) also chooses its own experiments. For each existing benchmark, we also run their own agent (LLM-AUTOSCILAB and APPRENTICE), which use the LLM in different ways (e.g., experiment design and/or forecasting). The main-text runs use GPT-5.6-Luna at medium reasoning effort. Older templates and BoxingGym results used Opus 4.7; their generation settings must not be read as universal defaults for the current comparisons.

G.1 CHEMBENCH (ENZYME RATE LAWS, §4.1)

G.1.1 MDA PROPOSER

The LLM proposer sees the experiments collected so far (design inputs → 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.

MDA proposer — rate law (CHEMBENCH)

```
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}]}]}
```

G.1.2 LLM-AUTOSCILAB AGENT

The baseline's upstream equation-discovery system prompt is shown below (from `autoscilab/llm/prompts.py`). In the audited run it is paired with the generic goal "Discover the governing rate law for this enzyme-catalyzed biochemical reaction," the benchmark's universal grammar, and no domain tags. Ground-truth evaluation is disabled during discovery and becomes available only after the final submitted law is frozen.

LLM-AUTOSCILAB — equation discovery

```
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.
```

G.2 BOXINGGYM (SECTION E)

G.2.1 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).

MDA proposer — GLM mean function (BOXINGGYM)

```
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 PPL implementations (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.

MDA proposer — NumPyro program (BOXINGGYM)

```
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."]
```

G.2.2 APPRENTICE AGENT

The original Box's Apprentice (Li et al., 2024a) (a.k.a. Box's LOOP) is an all-LLM loop: an LLM *synthesises* a PyMC probabilistic program for the data; the program is fit; an LLM *critic* inspects the posterior-predictive fit and returns revision hypotheses; and the loop repeats, with a separate LLM call choosing the next experiment. Its two core prompts — the PyMC program-synthesis prompt and the critic prompt — are reproduced here for reference (verbatim from `boxing_gym/agents/box_loop_prompts.py`, with runtime placeholders shown as `<...>`):

Box's LOOP (original baseline) — PyMC program synthesis [reference]

````
You are a brilliant statistician modeling a dataset.
Your job is to come up with a generative model that explains the true data by writing a
pymc probabilistic program.
Here is a description of the dataset
<dataset description>
Here is a description of the columns
<column descriptions>
If you are in the first round, you will not receive any additional information.
However, for the second round and beyond, I will give you the model you proposed
previously.

Please import pymc NOT pymc3!
Note that there are differences in the arguments pymc expects.
IMPORTANT: do not use sd as an argument use sigma instead!
It is crucial that you pass the idata_kwargs argument to pm.sample!!
IMPORTANT: Use the variable name "y_obs" for the observations when you define it!
IMPORTANT: Index the appropriate column names when grabbing data from observed_data. These
column names are indicated in the column description.
IMPORTANT: If appropriate, you may want to apply transformations to the columns.

Your answer should follow the template in the following order.
1. First, sketch a high-level probabilistic program for the data.
   You will go through multiple rounds of revision.
   If there's a previous program in your context window and a list of hypotheses, revise
   based on this information!
   Explicitly cite the hypotheses (if there are any) that you address in your sketch.
2. After coming up with a plan, write your program and add comments to lines of code that
address certain hypotheses.
```python
  import pymc as pm
  import numpy as np
  def gen_model(observed_data):
      # convert observed_data columns to numpy arrays
      # index the appropriate column names
      ...
      rng1 = np.random.default_rng(42)
      rng2 = np.random.default_rng(314)
      with pm.Model() as model:
          # create a pm.MutableData object for each non-observation column
          ...Your code here...
          trace = pm.sample(1000, tune=500, target_accept=0.90, chains=3, cores=1,
          random_seed=rng1)
          posterior_predictive = pm.sample_posterior_predictive(trace, random_seed=rng2,
          return_inferencedata=False)
          return model, posterior_predictive, trace
```
Do not forget to pass idata_kwargs={"log_likelihood": True} to pm.sample!
The input to gen_model, observed_data, contains the columns described earlier.
Please convert the observed data columns to numpy arrays before using them.
IMPORTANT: set seed for reproducibility! Only use the random seeds in the trace and
posterior_predictive line!!
````

```
IMPORTANT: Don't use Gaussian processes; the log prob is broken.
IMPORTANT: Please explicitly index column names into the dataframe! Never grab the raw
index.
IMPORTANT: Pass in cores=1 to pm.sample but NOT for pm.sample_posterior_predictive.
IMPORTANT: Do not forget to actually write the program! You will cost me millions if you
don't return the program.
I do not have fingers so I can't code the program on my own.
IMPORTANT: Make sure you actually write the program! Don't just sketch it!
IMPORTANT: In gen_model, stick to this return order: model, posterior_predictive, trace
For every feature, you should wrap it in a MutableData container. For example, if your
columns are 'x1', 'x2', and 'y' and 'y' is your observation:
with pm.Model() as model:
    x1 = pm.MutableData("x1", observed_data['x1'], dims='obs_id')
    x2 = pm.MutableData("x2", observed_data['x2'], dims='obs_id')
Please give the features the exact same name as the columns in the dataset!
Please also pass dims="obs_id" to the definition of the likelihood of the observations,
e.g.:
    y_obs = pm.Normal("y_obs", mu=mu, sigma=sigma, observed=observed_data['y'],
    dims="obs_id")
IMPORTANT: Do not use pm.Constant() for parameters. Do not pass in dims for latent
variables or parameters.
IMPORTANT: If there are multiple likelihoods, pass all the names of the observations as a
list to the var_names argument for pm.sample.

Here are previous hypotheses and syntheses. Revise accordingly.
Hypotheses: <previous hypotheses>
Synthesis: <previous synthesis>
```

Box's LOOP (original baseline) — model critic [reference]

```
You are a brilliant statistician specializing in critiquing and proposing revisions of
models!
Your equally brilliant colleague has come up with a list of probabilistic programs in pymc
that hypothesize generative models for the data.
I have fit these models, plotted their posterior predictive mean and variance, and LOO
scores (higher is better) for each program.

Your job: figure out if the probabilistic programs are consistent with the actual data!
To do this, you will compare the hypothesized models against the actual data by looking at
the plot of the posterior predictive against the actual data.

<expert context>

Here is a description of the dataset:
<dataset description>

Here is a description of the columns
<column descriptions>

Once you see the plot comparing the posterior predictive against the actual data, do the
following:
  1) Provide some natural language hypotheses for discrepancies between the models and the
  data by looking at the plots.
     Please explicitly let me know if you see the image, describe the properties of the
     model vs data that you see in the plot.
  2) By looking at all the programs and their scores, provide a synthesis of what
  strategies/approaches did and didn't work.
     Make this synthesis as informative as possible. I will pass this synthesis back to
     your colleague to revise the model.
     Make sure your suggestions are related to modeling (do not suggest collecting more
     data).

Here is the format I'm expecting in your response. Please stick to this format!

  ``` Hypotheses
      * Hypothesis
      * Hypothesis
  ```

  ```Synthesis
    (Your synthesis here)
  ```
```

Our reimplementation keeps this propose → fit → critique → refine loop but makes two changes so the comparison shares a model generator and numerical fitter. (i) It uses NumPyro instead of PyMC, fit by the *same* evidence-SMC as MDA. (ii) In place of the heavily-scaffolded prompt above, the program is generated by *MDA's own* structured proposer — the GLM proposer on the covariate-regression domains and the NumPyro proposer on the latent-variable domains (both shown in §G.2, above) — so the Apprentice and MDA share an *identical* model generator and differ only in

the inference/design strategy (a single evolving program vs. MDA's posterior over programs with VoI design). This is an adapted single-program control, not an evaluation of the unmodified published Box's Apprentice system. The only APPRENTICE-specific prompts are therefore the experiment-design call and the residual critique fed back to that shared proposer. First it picks its next experiment by an LLM call conditioned on its current program:

APPRENTICE (ours) — experiment design

```
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 each round it re-fits the single evolving program and feeds its residual back to the shared proposer:

APPRENTICE (ours) — residual critique fed to the shared 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.
```

G.2.3 ICL FORECASTER

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

ICL forecaster (BOXINGGYM)

```
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>]}
```

G.3 ELECTROPHYSIOLOGY (ION CHANNELS, §4.3)

This is our new benchmark, so there are no existing prompts to compare to. Both methods receive the same observation type and protocol information. Their processing differs: MDA computes summaries and fit diagnostics from these observations, whereas ICL forecasts directly from the observation table. These diagnostics contain no additional oracle measurements.

Current sparse-observation interface. The headline HH agents receive the protocol and exactly 20 observed time–voltage pairs per experiment, not a dense trace or the oracle's true six-feature vector. The numerical PF conditions on these sparse voltages. The ICL forecaster sees the accumulated observations and the 24 query schedules, then predicts six features per query. ICL (+LLM design) sees the unused action menu but not the held-out query schedules when choosing actions; it receives no mechanism archive or model-fitting tools. A malformed action receives one format-only retry and then the smallest unused index; a missing forecast is scored as a zero feature vector. These fallbacks

are retained in the audit. The fixed text below reproduces the current implementation's design and forecast templates; braces around data/menu blocks indicate substituted run-specific observations rather than extra instructions.

Current HH ICL (+LLM design): action selection

```
System:
You are an electrophysiologist choosing the next current-clamp experiment for a model-free
in-context learner. Use only the observed sparse voltage trajectories. Do not posit or fit
an explicit mechanism. Return one JSON object only.

User template:
Choose one unused action that should most improve prediction across diverse future current
stimuli. The held-out evaluation stimuli are not shown.

Observed experiments:
{experiment_i: design=<schedule>; trajectory=<20 time-voltage pairs>}

Unused action menu:
{- <index>: <schedule>}

Reply exactly as {"action_index": integer}. The integer must be one listed above.
```

Current HH ICL: sparse-voltage forecast

```
System:
You are an electrophysiologist making direct transductive forecasts from raw sparse
voltage trajectories. Do not propose or fit an explicit mechanistic model. Return one JSON
object only.

User template:
Each requested response is [n_test, n_pre, run_down, adaptation, V_min, V_end]; counts and
count differences are dimensionless and voltages are in mV.

Observed experiments:
{experiment_i: design=<schedule>; trajectory=<20 time-voltage pairs>}

Held-out designs:
{- <ID>: design=<schedule>}

Predict all six numbers for every ID. Reply exactly as {"predictions":{"<ID>":[six
numbers], ...}}.
```

Historical feature-summary template (not the headline interface). The following older example compresses each trajectory to six features:

What the LLM sees as "data" (NEURONBENCH)

```
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_end=-65
  - paired long pulses (12/300, 60 gap, 12/300)  -> n_test=16, n_pre=8, run_down=-8,
  adaptation=5, V_min=-74, V_end=-64
  - hyperpol step then release (rebound)         -> n_test=4,  n_pre=0, run_down=-4,
  adaptation=1, V_min=-88, V_end=-65
```

$n_{\text{test}}/n_{\text{pre}}$ are the test-window / pre-pulse spike counts; run-down $= n_{\text{pre}} - n_{\text{test}}$; adaptation is early-minus-late test spikes; $V_{\text{min}}/V_{\text{end}}$ are the sub-threshold sag depth and steady-state tail (mV). This older representation must not be confused with the raw 20-point observations used in the current comparison.

G.3.1 MDA PROPOSER

The current generic proposer specifies additional currents or slow-Na inactivation through executable JSON, not a named-archetype menu. It sees intervention schedules, 20 observed voltages, and summaries computed from those sparse observations, not dense oracle features. Later calls include archived mechanisms and predictive-check diagnostics. The legacy system sentence still says "spike-count data"; the payload below includes the sparse voltages used in the headline run.

Current MDA proposer — generic HH mechanisms

```
System (verbatim):
You are an electrophysiologist proposing candidate ion-channel mechanisms to explain
current-clamp spike-count data. Reply with a JSON object only.
```

```
User template (generic mode; placeholders denote the observed data):
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 standard
Na+/K+ spiking currents. The current model may be incomplete. Infer one or more additional
membrane mechanisms from the interventions and observed trajectory summaries. Do not
assume that a named benchmark archetype is present. Candidate mechanisms must use one of
the executable forms in the schema below.

<Collected protocol schedules and summaries computed from sparse observations>
Sparse raw trajectories (relative time in [0,1], voltage in mV; every trajectory has 20
displayed points):
  <protocol label> -> [(relative_time, voltage), ...]

[On later proposal calls:]
Mechanisms ALREADY TRIED and their fit residuals -- propose ones STRUCTURALLY DIFFERENT
(different reversal / direction / inactivation):
<archive descriptions and predictive-check diagnostics; no hidden truth or future
observations>

Propose <N> distinct plausible membrane mechanisms (or return an empty list if the
standard Na+K model is adequate).
JSON schema: {"hypotheses":[{"name":str,
"mechanism_type":"extra_current"|"slow_na_inactivation", "reversal_mV":float,
"opens_on":"depol"|"hyperpol", "inactivates":bool, "g_bounds":[lo,hi],
"half_mV_bounds":[lo,hi], "tau_ms_bounds":[lo,hi]}]}. For
mechanism_type="slow_na_inactivation", give only name and tau_ms_bounds; the other fields
are ignored.
```

G.3.2 ICL FORECASTER

The following is the historical six-feature forecaster template, not the current sparse-observation prompt described above:

ICL forecaster (NEURONBENCH)

```
[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`).

G.4 GLUCOSEBENCH: COMPARTMENT MODELS AND DIRECT FORECASTING

All glucose LLM calls use GPT-5.6 Luna with medium reasoning effort. The following system text reproduces the implemented prompts; user payloads are shown schematically, with run-specific arrays substituted at execution. Patient identity, age, hidden states, and simulator parameters are withheld. MDA sees the fitted candidate pool and its training predictions, whereas ICL sees observations and input schedules, not MDA's fitted models.

G.4.1 MDA MODEL PROPOSER

The census prompt includes explicit parent IDs and edit descriptions for genealogy logging. This metadata is stored separately from executable ODE expressions; it does not require a single parent when a proposal combines mechanisms. The earlier illustrative pilot used the same base prompt without the genealogy instructions. The multi-cut follow-up reuses these frozen models and makes no new model-proposal calls.

Glucose MDA: ODE proposal and genealogy

```
System (census version, including genealogy instructions):
Infer a reduced continuous-time compartmental ODE from noisy CGM and intermittent
plasma-insulin observations and known meal/insulin inputs. You have no simulator access.
Return ONLY JSON {"models":[...]} with at most two new models. Each model has name, states
(1-6 unique identifiers), rhs (one expression per state), initial_states, outputs (two
observation expressions: CGM mg/dL and plasma insulin pmol/L), bounds (positive [lo,hi]
per parameter), initial (parameter values). Parameters p0,p1,...,p11. Independent
constants: gb (mean CGM prefix), ib (mean observed insulin prefix), meal (g/min), insulin
(U/min above background). Time in minutes. Allowed arithmetic +,-,*,/,**,
exp,log,sqrt,tanh,abs,minimum,maximum; no arbitrary Python, indexing, conditionals, or
imports. Initial state expressions may use only gb, ib and parameters. Models restart at
minute45; effective meal/insulin deviation states start at zero. Insulin assays have
independent Gaussian SD 10 pmol/L and missing entries are null, never zero. All parameters
and CGM AR1 observation-noise parameters will be refitted by multistart optimization.
Propose mechanistic edits addressing training residuals, not polynomial time
interpolation. No patient identity or ground-truth model is known. Existing models are
examples of the exact schema. For EACH proposed model add parent_ids (a list of model_id
strings from existing_models), origin ("edit", "recombination", or "de_novo"), and
edit_summary (a nonempty explanation). An edit has exactly one parent; recombination has
at least two distinct parents; de_novo has none. Name actual source models, not simply the
best-scoring model. All models remain visible as context, regardless of which parents you
declare.

User payload (JSON; angle-bracket blocks are substituted run data):
{
 "observations": <observed training episodes: times_min, cgm_mg_dl, insulin_pmol_l,
 insulin_mask, insulin_noise_sd_pmol_l, protocol, prefix_end_min>,
 "delivery": "Meals consumed at 5g/min; bolus spread uniformly over 5min. Constant basal
 is absorbed into equilibrium.",
 "existing_models": <each fitted model's model_id, executable model specification, theta,
 bic, and training predictions rounded to three decimals>
}
```

G.4.2 ICL FORECASTING AND DESIGN

The completed census forecaster conditions on a 45-minute prefix. ICL (+MDA design) replays MDA's history; ICL (+LLM design) uses the following additional acquisition prompt. Both share the same forecast prompt.

Glucose ICL: completed census forecast

```
System (completed 45-minute-cut census):
Predict future CGM (mg/dL) and plasma insulin (pmol/L) directly from observed examples. Do
not propose a mechanistic model or run code. You have complete training episodes, only the
query observations through the forecast cut, and the complete known query input sequence.
Insulin nulls are missing assays, not zeros. CGM noise is correlated; insulin assay SD is
10 pmol/L. Predict conditional means, not random noise. Return ONLY JSON with cgm_mg_dl
and insulin_pmol_l arrays, in exactly the requested time order. No extra keys. No future
measurements are available.

User payload (JSON; angle-bracket blocks are substituted run data):
{
 "training_episodes": <complete public training episodes>,
 "query": <observations/masks through minute 45 plus the complete known input schedule>,
 "forecast_cut_min": 45,
 "requested_cgm_times_min": <50,55,...,360>,
 "requested_insulin_times_min": <60,75,...,360>
}

One schema-only retry appends:
Previous output failed schema validation: <error>. Return complete corrected arrays.

Public episode fields: times_min, cgm_mg_dl, insulin_pmol_l, insulin_mask,
insulin_noise_sd_pmol_l, minute_inputs, input_columns, protocol. No patient identity, age,
simulator parameters, clean state, or fitted MDA model is supplied.
```

Glucose ICL (+LLM design): experiment selection

```
System:
Choose the next experiment to improve prediction of CGM and plasma-insulin responses
across the declared meal/bolus design range. Only previous observations are available.
Basal is fixed; choose an unused menu index. Return ONLY JSON {"action_index": integer}.

User payload (JSON; angle-bracket blocks are substituted run data):
{
 "training_episodes": <complete public training episodes>,
 "available_templates": <unused index-to-protocol dictionary; each protocol gives meal_g,
 meal_minute, bolus_U, bolus_minute>
}

One schema-only retry appends:
Invalid previous selection: <error>

The design prompt contains no sealed test outcomes or fitted MDA models.
```

The separate multi-cut evaluation uses an explicit empty-prefix contract at cut zero and variable-length forecast targets. This prompt is not the one used for the completed census curves.

Glucose ICL: multi-cut forecast

```
System (multi-cut follow-up):
Forecast conditional means of CGM (mg/dL) and plasma insulin (pmol/L) from the observed
training episodes. Only the provided query prefix and complete known input schedule are
available. At cut zero there are NO query observations, including at time zero: infer
initialization from training episodes alone. Missing insulin assays are not zeros. Predict
directly without code or mechanistic model proposals. Return only JSON arrays cgm_mg_dl
and insulin_pmol_l in the requested time order, with no extra keys.

User payload (JSON; angle-bracket blocks are substituted run data):
{
 "training_episodes": <complete public training episodes>,
 "query": <masked prefix through the chosen positive cut, or empty observation/time/mask
 arrays at cut zero; always includes complete known input schedule>,
 "forecast_cut_min": <0,45,120,or 180>,
 "requested_cgm_times_min": <CGM times strictly after the cut>,
 "requested_insulin_times_min": <assay times strictly after the cut>,
 "zero_cut_means_no_query_observations": true
}

Up to two schema-only retries append:
Schema error: <error>

The launch permits three total attempts per forecast. Neither schema feedback nor retries
use forecast accuracy or future observations.
```