SIMAUTHOR: HARNESSING FOUNDATION MODELS FOR PERSISTENT SCIENTIFIC SIMULATOR AUTHORING

Anonymous authors
Paper under double-blind review

ABSTRACT

Foundation models can generate scientific code, but authoring a scientific simulator (an executable program encoding hypotheses about how mechanisms generate observable signals) requires iterative refinement. Scientific adequacy rarely admits a unique implementation or exact test, so simulators must instead be judged against limited real observations. We study this setting as *scientific simulator authoring under weak empirical feedback*, where distributional comparisons between simulated and real signals guide revision, and the target is the simulator itself rather than only its generated samples. We introduce SIMAUTHOR, a persistent authoring harness that retains and revises executable simulators, separates scalar search scores from structured discrepancy feedback, and accumulates reusable implementation mechanisms. We evaluate SIMAUTHOR on six biomedical tasks spanning cardiac and respiratory audio, photoplethysmography (PPG), and electrocardiography (ECG). Under a fixed 100-attempt budget, SIMAUTHOR outperforms PUCT score search on all six tasks, generally outperforms textual-strategy optimization, and achieves the highest endpoint score on five of six. The authored simulators also improve on unseen recordings, transfer to independent pretrained representations, and yield substantial out-of-distribution gains in downstream ECG classification. Finally, 111 of 138 audited revisions alter program structure and account for 86.1% of the signed score improvement. These results suggest that persistent revision can progressively convert foundation-model knowledge into better executable scientific simulators from limited empirical evidence.

1 INTRODUCTION

Foundation models, when coupled with inference-time harnesses that organize search, execution, and revision, are becoming increasingly capable scientific and programming systems. Pretraining equips them with substantial domain knowledge and strong zero-shot code generation abilities (Singhal et al., 2023; Li et al., 2023; Guo et al., 2024), while inference-time scaffolding extends what they can accomplish through planning and iterative refinement (Shinn et al., 2023; Romera-Paredes et al., 2024; Novikov et al., 2025). These capabilities raise an underexplored question: *can foundation models author scientific simulators?* We consider a simulator an executable program that encodes hypotheses about how interpretable mechanisms and stochastic processes generate observable real-world signals. Unlike many LLM-guided program-search settings, simulator authoring typically lacks an exact verifier: there is rarely a unique correct implementation, and no unit test determines whether the generated behavior is scientifically adequate. Candidate simulators must therefore be judged indirectly against limited real-world observations.

We study this problem in *biomedical signal simulation*, where the opportunity and the difficulty are especially concrete. For example, a COPD respiratory simulator is an executable program that generates breathing recordings from explicit models of respiratory timing, adventitious sounds, and measurement variability. Yet this knowledge does not directly translate into a useful simulator: a plausible zero-shot program may capture broad phenomenological features while omitting the stochastic, temporal, and measurement processes (Figure 1, left). Simulation is particularly valuable when real recordings are scarce, imbalanced, or difficult to share (Rieke et al., 2020; Wagner et al., 2020; Rocha et al., 2017), but the same scarcity provides only limited empirical grounding for authoring and validating the simulator. In the absence of an exact verifier, we use distributional comparisons between generated and real recordings as empirical feedback for revision (Figure 1,

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

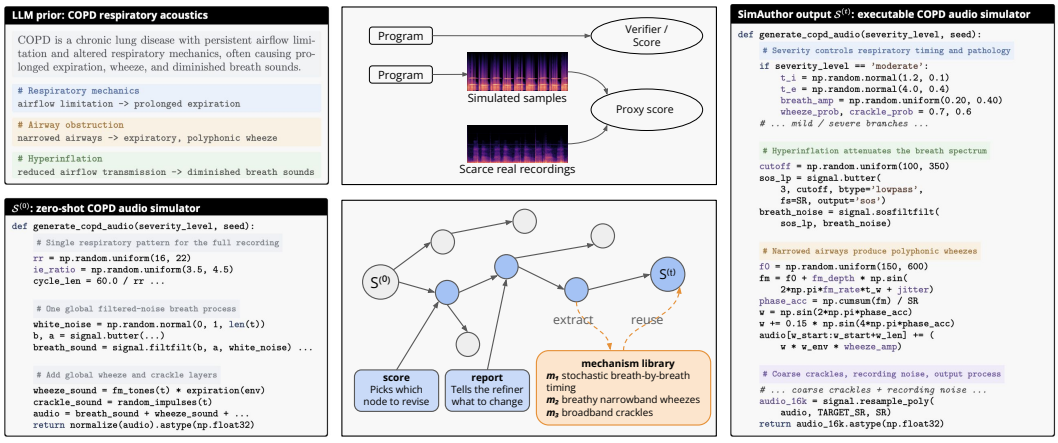

Figure 1: **Overview.** SimAuthor turns LLM prior knowledge into executable simulator code when exact verification is unavailable and feedback comes only from scarce real data.

middle top). In this work, the target is the simulator itself rather than only its samples: an explicit, executable model that can be inspected, modified, and controlled (Figure 1, right).

We introduce SIMAUTHOR, a harness designed for this weak-feedback authoring regime (Figure 1, middle). Rather than repeatedly generating programs from scratch, SIMAUTHOR persistently develops executable simulator candidates, using empirical feedback to guide revision while retaining useful mechanisms discovered across model calls. In this way, simulator construction becomes a cumulative authoring process rather than zero-shot code generation.

We evaluate this setting on six biomedical simulator-authoring tasks spanning cardiac audio, respiratory audio, electrocardiography (ECG), and photoplethysmography (PPG). Under a fixed 100-attempt authoring budget, SIMAUTHOR outperforms repeated independent generation and score-only program search on all six tasks, generally outperforms textual-strategy optimization, and achieves the strongest endpoint on five of six. Additional evaluations on unseen representations and downstream prediction tasks provide evidence that these gains extend beyond the authoring objective. Component ablations show that structured discrepancy feedback is critical for final quality, while mechanism memory consistently improves search efficiency. A revision audit further finds that structural program changes account for 86.1% of the signed score improvement, indicating that progress is realized through changes to the explicit, inspectable simulator rather than only parameter retuning. These results support persistent artifact-centered revision as a way to convert foundation-model knowledge into better executable scientific models under weak empirical feedback.

Our contributions can be summarized as follows:

1. We introduce **foundation-model-based scientific simulator authoring** as a new problem setting: foundation models construct executable simulators that encode scientific assumptions and generative mechanisms. We formalize this setting under weak empirical feedback, where only limited real observations and no exact verifier are available.
2. We introduce SIMAUTHOR, a persistent authoring harness that progressively revises executable simulators while carrying useful information and mechanisms across model calls.
3. We evaluate simulator authoring across six biomedical tasks and four signal modalities, showing consistent gains over independent sampling, score-only search, and a textual-strategy optimization baseline, with additional evidence that these gains extend beyond the authoring objective.

2 PERSISTENT SIMULATOR AUTHORING

Let $\boldsymbol{\theta} \in \Theta$ denote the simulator's explicit variables, $\mathbf{z} \in \mathcal{Z}$ the values it samples at run time, such as beat-to-beat timing or recording noise, and $\mathbf{y} \in \mathcal{Y}$ an observable signal. A simulator is an executable program

$$\mathcal{S} : \Theta \times \mathcal{Z} \to \mathcal{Y}. \tag{1}$$

The target of authoring is the program $\mathcal{S}$, not an individual sample. Authoring begins from a small reference set $\mathcal{D}_{\text{ref}}$ of real recordings and a root simulator $\mathcal{S}^{(0)}$ written in a single call. Every subsequent attempt revises an existing candidate, so a trajectory has the form

$$\mathcal{S}^{(0)} \to \mathcal{S}^{(1)} \to \cdots \to \mathcal{S}^{(T)}, \tag{2}$$

where each revision can change program structure, parameters, which quantities are sampled and how, and how the signal is recorded (sensor response, filtering, noise). A single model call decides only what to write next. Everything else is the harness's to decide: which of the accumulated candidates to revise, what evidence about that candidate the model is shown, and what from earlier attempts is carried forward. These three decisions define the three components below.

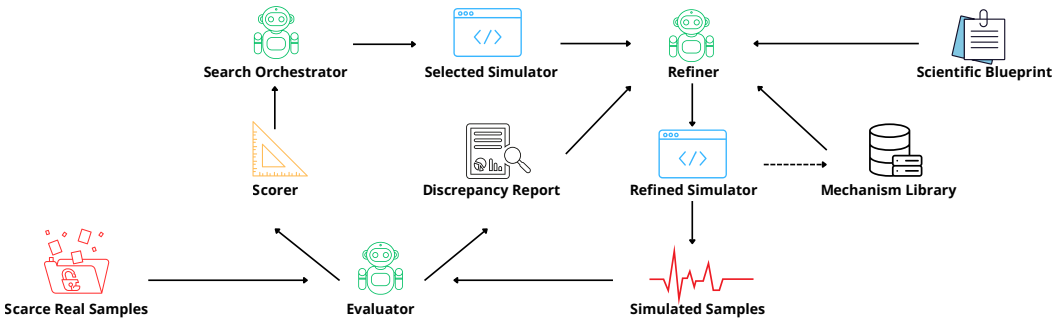

Figure 2: **The SIMAUTHOR authoring harness.** A search orchestrator selects simulator candidates based on numerical scores, while the refiner updates them using the scientific blueprint, discrepancy report, and mechanism library.

2.1 SEARCH SCORE AND DISCREPANCY REPORT

Each candidate is executed to generate a set of simulated observations, and a modality-specific **evaluator** compares these with the real reference observations, returning two complementary outputs. The **search score** is a scalar used by the search policy to select candidate programs. The **discrepancy report** is structured evidence describing where generated and real signals disagree. The distinction is deliberate: the score answers which candidate should search revise next, the report answers what should the refiner change.

Evaluator. We fix one evaluator for each modality before search and never update it using generated programs. Audio and PPG are evaluated in signal space: the audio evaluator averages marginal distribution overlap for 13 MFCCs and the zero-crossing rate, while the PPG evaluator compares ten pulse-level rhythm and morphology statistics. Because delineation-based objectives were brittle for ECG, we instead compare distributions of frozen ECGFounder embeddings using a scaled energy-distance score (Li et al., 2025). All scores lie in $[0, 1]$, with higher values indicating greater similarity to the reference set. The discrepancy report exposes the components underlying each score, such as feature-level mismatches, distribution spread, and possible mode collapse. These evaluators are fixed authoring objectives rather than complete measures of physiological fidelity; we therefore separately test generalization to held-out recordings, alternative representations, and downstream utility in sections 4.2 and 5.2. Full definitions are provided in Appendix A.

2.2 SCIENTIFIC BLUEPRINT AND MECHANISM LIBRARY

The refiner is conditioned on two complementary textual contexts: a *static scientific blueprint* that anchors revision in domain knowledge, and a *dynamic mechanism library* that accumulates useful implementation knowledge over the course of authoring. For each task, the scientific blueprint provides fixed context about relevant physiology, signal manifestations, and implementation constraints. It is generated by the foundation model before any recording is consulted, externalizing the model's prior knowledge as a document available to the refiner on every attempt. The blueprint complements empirical discrepancy feedback: the latter indicates how simulated outputs differ from real recordings, while the former provides a stable scientific reference for how those discrepancies might be addressed. The prompt for blueprint generation is provided in Appendix J.1.

The mechanism library, in contrast, evolves during authoring. It stores reusable implementation mechanisms distilled from productive parent–child revisions. When a revision improves clearly over its parent, the program difference is summarized as a named mechanism, and recent mechanisms are made available to subsequent refinement attempts. This allows implementation knowledge discovered in one branch of the search to persist and be reused later. Representative mechanism-library entries, implementation details and full prompts for mechanism extraction are provided in Appendix C, Appendix D, and Appendix J.4.

2.3 SEARCH ORCHESTRATION

The orchestrator holds every candidate authored so far together with the tree of revisions that produced them. Following Aygün et al. (2025), we use a flat-pool PUCT search that balances exploitation of high-scoring programs with exploration of less-visited ones:

$$n_t = \arg\max_{n \in \mathcal{T}} \left[\tilde{v}(n) + c_{\text{puct}}\, p(n) \frac{\sqrt{N_{\text{tot}}}}{1 + N(n)} \right], \tag{3}$$

where $\tilde{v}(n) \in [0, 1]$ is the candidate's rank-normalized search score, $N(n)$ its visit count, N_{tot} the total visits, and $p(n) = 1/|\mathcal{T}|$ a uniform prior. The first term favors high-scoring candidates and the second favors less-visited ones. PUCT score search uses the same selection rule, so comparing it with SIMAUTHOR holds search orchestration fixed and varies the information available to the refiner. The end-to-end loop is given in Appendix B.

3 EXPERIMENTAL SETUP

Tasks. We evaluate six simulator-authoring tasks spanning four signal modalities: ventricular septal defect (VSD) and aortic stenosis (AS) in cardiac audio, chronic obstructive pulmonary disease (COPD) in respiratory audio, atrial fibrillation (AF) in PPG, and long-QT syndrome (LQT) and Wolff–Parkinson–White syndrome (WPW) in two-lead ECG. Several conditions are rare in public data: PTB-XL (Wagner et al., 2020) contains only 118 LQT and 80 WPW recordings among 21,837 ECGs. Each task has a small search-reference set (37–96 recordings), which the evaluator compares against during search, and a held-out evaluation set never seen during search: independent external datasets for four tasks, held-out same-source patient-disjoint splits for two. Details are in Appendix E.

Baselines. We compare SIMAUTHOR against three alternatives spanning independent sampling, score-guided program search, and natural-language optimization. **Independent Sampling (Sampling)** spends the authoring budget on independently generated programs and returns the best-of-N, following the repeated-sampling strategy commonly used in code generation (Chen et al., 2021). **PUCT Score Search** selects parents with rank-based flat PUCT and revises each selected program using only its scalar score as feedback, following prior score-guided search over executable programs (Aygün et al., 2025). **Textual Strategy Optimization (Text-Opt).** Text-Opt uses the same blueprint, evaluator feedback, visual feedback, and PUCT policy as SIMAUTHOR, but searches over natural-language strategies rather than programs. Each node carries a strategy σ_p, from which the LM generates an updated strategy and a new simulator from scratch:

$$(\sigma', P') \sim \text{LM}(\sigma_p, B, O, R_p, V_p).$$

By contrast, SIMAUTHOR inherits and revises the selected program P_p and can reuse mechanisms M discovered across revisions:

$$P' \sim \text{LM}(P_p, M, B, O, R_p, V_p).$$

Thus, Text-Opt isolates strategy-space optimization from program inheritance and mechanism reuse (Yang et al., 2024; Agrawal et al., 2025).

Authoring budget. One authoring attempt is one foundation-model call proposing an executable simulator: for the search methods it revises an existing candidate, for the independent sampling baseline it is an independent program generation. Every method receives the same 100 attempts, and invalid or non-executable code consumes the budget. All primary trajectories use the same foundation model, Gemini 3.1 Pro (Google DeepMind, 2026). Numerical settings are in Appendix D.

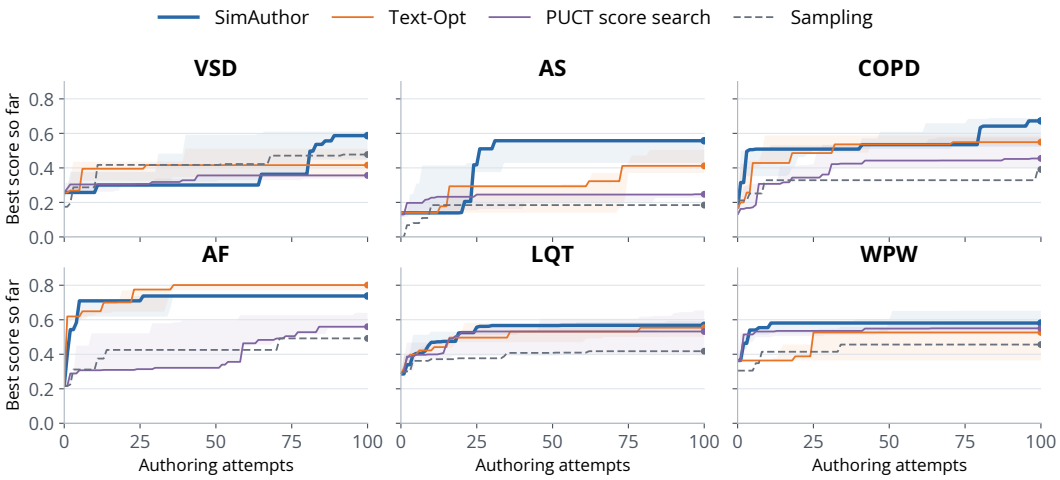

Figure 3: **Best-so-far performance over 100 authoring attempts.** For seeded methods, curves show the complete trajectory of the median-Best@100 run, with min–max shading across three seeds. Independent sampling follows its fixed proposal order.

Table 1: **Endpoint and upper-tail performance after 100 authoring attempts.** Audio tasks use MFCC feature overlap, AF uses hand-designed PPG rhythm and morphology statistics, and ECG tasks use frozen ECGFounder embeddings (Li et al., 2025). Entries report Best@100 / top-10 mean. Seeded methods are summarized by the median over three seeds; Sampling uses the fixed first 100 proposals. Bold denotes the best result and underline the second best.

Method	*MFCC*						*Morphology*		*ECGF12*			
	VSD $\mathcal{S}^{(0)}$: .258		**AS** $\mathcal{S}^{(0)}$: .139		**COPD** $\mathcal{S}^{(0)}$: .165		**AF** $\mathcal{S}^{(0)}$: .216		**LQT** $\mathcal{S}^{(0)}$: .286		**WPW** $\mathcal{S}^{(0)}$: .364	
	$S_{\max}$	S_{top10}	$S_{\max}$	S_{top10}	$S_{\max}$	S_{top10}	$S_{\max}$	S_{top10}	$S_{\max}$	S_{top10}	$S_{\max}$	S_{top10}
Sampling	.477	.424	.184	.152	.391	.259	.492	.412	.418	.401	.456	.406
PUCT Score Search	.356	.327	.247	.202	.455	.444	.560	.528	.532	.530	.550	**.546**
Text-Opt	.416	.348	.412	.250	.549	.505	**.801**	**.770**	.554	.477	.526	.457
SimAuthor	**.587**	**.483**	**.557**	**.443**	**.672**	**.628**	.738	.701	**.568**	**.553**	**.581**	.527

4 Results

We organize the experiments around three questions. First, does SimAuthor discover higher-quality simulators than independent sampling, score-guided program search, and textual strategy optimization, and how efficiently does it reach them (section 4.1)? Second, do the resulting simulators generalize beyond the feedback used during authoring (section 4.2)? Third, which parts of the authoring harness account for these gains (section 4.3)?

4.1 SimAuthor pushes the simulator quality frontier

We run each search-based method with three random seeds to capture run-to-run variation. To visualize one coherent trajectory, Figure 3 shows the run with the median final Best@100, with min–max shading across the three seeds. Table 1 reports $S_{\max}$ and S_{top10} from the same run; complete seed-level results are reported in Appendix F.1. SimAuthor achieves the highest $S_{\max}$ on five of six tasks (Figure 3, Table 1). AF is the exception: Text-Opt reaches .801, compared with .738 for SimAuthor. The top-10 results sharpen this picture. SimAuthor has the highest S_{top10} on four tasks, showing that its advantage often extends beyond a single high-scoring endpoint to the quality of the candidate pool; Text-Opt remains strongest on AF, while PUCT Score Search is slightly higher on WPW.

Table 2: **Generalization beyond the authoring objective.** **(a)** Held-out representation evaluation on audio and PPG; entries are $R_{\text{sel}} \mid R_{\max}$. **(b)** ECG downstream utility with 300-example augmentation, reported as AUPRC; evaluation is in-distribution (ID) on PTB-XL and out-of-distribution (OOD) on Chapman–Shaoxing. Corresponding AUROC results in Appendix H. Bold indicates the best result.

(a) Independent representation evaluation

	OPERA-CE			*PaPaGei*
Method	**VSD**	**AS**	**COPD**	**AF**
Root $\mathcal{S}^{(0)}$	.317 | .317	.192 | .192	.152 | .152	.354 | .354
Sampling	.123 | .715	.106 | .612	.162 | .799	.434 | **.830**
Text-Opt	**.443** | .784	.258 | .685	.468 | **.847**	.631 | .734
SimAuthor	.204 | **.789**	**.306** | **.906**	**.561** | .752	**.665** | .737

(b) ECG downstream utility

	LQT		**WPW**	
Training	**ID**	**OOD**	**ID**	**OOD**
Real	**.406**	.633	.857	.691
+ Conv.	.405	.642	.810	.733
+ SimAuthor	.404	**.747**	**.861**	**.804**

The controlled comparisons further indicate that search allocation alone does not explain these gains. Against PUCT Score Search, which uses the same candidate-selection rule, SIMAUTHOR improves $S_{\max}$ on all six tasks (.031–.310). Under weak empirical feedback, the limiting factor therefore appears to be not only which candidate is revisited, but whether the refiner receives enough structured information for useful changes to accumulate in the retained program. Compared with Text-Opt, SIMAUTHOR is stronger on five of six tasks, suggesting that making the executable artifact itself persistent is generally more effective than carrying forward only a textual strategy and regenerating the program from scratch. AF is the lone exception. Its evaluator exposes physiologically interpretable rhythm statistics whose discrepancies map relatively directly to synthesis changes, making textual strategy updates particularly effective.

Equal authoring attempts do not imply equal inference cost: SIMAUTHOR uses approximately $3.8\times$ the tokens of best-of-100 independent sampling because it carries substantially more context across revisions. Expanding independent sampling to the corresponding token-matched budget of roughly 380 proposals does not close the gap, indicating that the gains are not explained by additional inference budget alone (Appendix G).

4.2 SIMAUTHOR'S GAINS PERSIST BEYOND THE AUTHORING OBJECTIVE

The authoring objectives are deliberately narrow, raising the possibility that SIMAUTHOR merely exploits the feedback function it is given. We therefore test generalization in two complementary ways: by changing the evaluator after authoring, and by changing the objective from distribution matching to downstream classification. For VSD, AS, COPD, and AF, programs are authored using only the original signal-space objectives and then re-evaluated with frozen pretrained representations that are never exposed to the authoring loop: OPERA-CE for audio (Zhang et al., 2024) and PaPaGei-S for PPG (Pillai et al., 2025). Each entry in table 2a reports $R_{\text{sel}} \mid R_{\max}$, the representation score of the signal-selected program and the best representation score anywhere in its 100-candidate pool. SimAuthor achieves the strongest R_{sel} on AS, COPD, and AF, and the strongest $R_{\max}$ on VSD and AS. It also exceeds Independent Sampling at the selected endpoint on all four tasks and Text-Opt on three of four. The often substantial gap between R_{sel} and $R_{\max}$ shows that the authoring and representation evaluators are not interchangeable, while improvement under an unseen representation provides evidence that the search is not merely exploiting the handcrafted authoring score.

For LQT and WPW, we test whether authored simulators improve learning from scarce real labels. We compare real-only training with two matched augmentations of 300 positive examples: standard waveform perturbations (Raghu et al., 2022) and samples from the canonical SimAuthor program. A frozen ECGFounder representation feeds a class-weighted logistic-regression probe, evaluated on patient-disjoint PTB-XL as in-distribution (ID) and external Chapman–Shaoxing as out-of-distribution (OOD) (Wagner et al., 2020; Zheng et al., 2022). SimAuthor raises OOD AUPRC from .633 to .747 for LQT and from .691 to .804 for WPW, compared with .642 and .733 under conventional augmentation, while preserving ID performance; OOD AUROC reaches .889 and .965, respectively. These results show that the authored ECGs transfer from representation matching to classification of unseen real recordings, with the largest gains under distribution shift. Because authoring and downstream probing both use ECGFounder, we interpret this as task-aligned downstream utility rather than encoder-independent generalization or clinical validity; full details are in Appendix H.

Table 3: **Focused component ablations. (a)** Final candidate quality, measured by $S_{\max}$ after 100 authoring attempts. **(b)** Search efficiency, measured by AUC_{100}, the mean best-so-far score over the 100-attempt trajectory. All values are medians over three seeds. Parentheses report the change relative to SimAuthor for the same metric.

(a) Final quality: $S_{\max}@100$

Task	SimAuthor	− Report	− Library
VSD	.587	.366 (−.221)	.399 (−.188)
AS	.557	.152 (−.405)	.383 (−.174)
COPD	.672	.609 (−.063)	.662 (−.010)
AF	.738	.589 (−.148)	.719 (−.019)
LQT	.568	.460 (−.108)	.540 (−.028)
WPW	.581	.492 (−.089)	.573 (−.008)

(b) Search efficiency: AUC_{100}

Task	SimAuthor	− Report	− Library
VSD	.395	.356 (−.039)	.348 (−.047)
AS	.458	.143 (−.315)	.324 (−.134)
COPD	.544	.585 (+.041)	.529 (−.016)
AF	.717	.556 (−.161)	.686 (−.031)
LQT	.539	.439 (−.101)	.514 (−.026)
WPW	.574	.486 (−.088)	.547 (−.027)

4.3 Structured feedback and memory support cumulative authoring

We focus on the two components that determine how information from previous attempts is carried into subsequent revisions: the structured discrepancy report and the mechanism library. Table 3 evaluates both final candidate quality, $S_{\max}@100$, and search efficiency, AUC_{100}. All values are medians over three seeds; complete seed-level results are provided in Appendix F.2. Removing the discrepancy report reduces final quality on every task, by .063–.405, with particularly large losses on AS (−.405), VSD (−.221), and AF (−.148). Search efficiency also decreases on five of six tasks. This indicates that a scalar score is insufficient for reliable revision: the report is what turns empirical mismatch into directional information about how the retained program should change. COPD is the only exception in AUC_{100}, where removing the report improves the trajectory average despite lowering the final score, showing that more informative feedback does not necessarily improve every intermediate search step. Removing the mechanism library has a smaller effect on final endpoints (−.008 to −.188), but lowers AUC_{100} on all six tasks. The contrast suggests a different role for memory: rather than primarily unlocking isolated high-scoring candidates, the library helps useful implementation mechanisms persist and be reused, making progress accumulate more efficiently across revisions.

5 Analysis

Three alternative explanations could account for the score gains in section 4: parameter retuning rather than genuine program authoring, overfitting to the small search-reference set rather than the condition, or sensitivity to the exploration setting or foundation model. We test each in turn.

5.1 SimAuthor improves simulators through structural revision

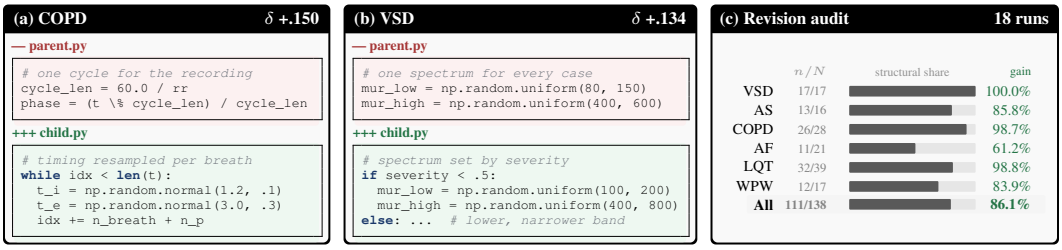

Figure 4: **Structural revision during authoring. (a,b)** Abridged code changes. **(c)** Structural share of 138 selected-lineage revisions and their signed score gain.

To test whether iterative authoring changes simulator logic rather than merely retuning constants, we trace the selected final simulator in each of the 18 SimAuthor runs back to the root and audit every scored parent–child transition along that ancestry path. Of these, 111 are structural and account for 86.1% of the net score improvement (Figure 4; full audit protocol in Appendix I). Structural

revisions form the majority on five of six tasks and explain 83.9–100% of the gain on those tasks. The examples show that these changes introduce new generative logic (e.g. breath-level stochastic timing in COPD and severity-conditioned spectra in VSD) rather than merely adjusting existing constants. This suggests that cumulative authoring improves simulators primarily by evolving their executable structure.

5.2 Score gains generalize to held-out recordings

Table 4: **Generalization to held-out recordings.** VSD and AS use patient-disjoint ID splits; the remaining tasks use external OOD datasets. Scores use the same evaluator as authoring. SimAuthor reports the median across three seeds; parentheses show gains over the root. See Appendix E for dataset details.

	Patient-disjoint ID		*External OOD*			
Method	VSD	AS	COPD	AF	LQT	WPW
Root $\mathcal{S}^{(0)}$	.223	.103	.088	.104	.277	.385
Sampling	.425 (+.202)	.146 (+.043)	.362 (+.275)	.117 (+.013)	.341 (+.064)	.443 (+.059)
SimAuthor	**.577** (+.354)	**.440** (+.338)	**.507** (+.420)	**.204** (+.100)	**.480** (+.203)	**.538** (+.153)

A high search score could reflect fitting the few reference recordings rather than the underlying condition, so we re-evaluate selected candidates on OOD recordings whenever available. VSD and AS use patient-disjoint in-domain splits, while COPD, AF, LQT, and WPW are evaluated on external OOD datasets (table 4; Appendix E). Every final simulator improves on its root, by $+.100$ to $+.420$, and median SimAuthor exceeds the authoring-objective-selected best-of-100 sampling candidate on all six conditions, by $+.087$ to $+.294$. Thus the gains transfer to unseen recordings under the authoring evaluator rather than merely fitting the search patients.

5.3 Robustness to the exploration setting and the foundation model

Table 5: **Sensitivity to exploration and authoring backbone.** Final $S_{\max}@100$ over three seeds. Each cell lists scores in ascending order (median in the middle); bold marks the highest median for each task.

	PUCT exploration		
Task	$c = .10$	$c = .25$	$c = .50$
VSD	[.330, .362, .472]	[.448, **.587**, .611]	[.488, .504, .535]
LQT	[.513, .548, .554]	[.541, .568, .594]	[.572, **.596**, .637]

	Authoring backbone	
Task	DeepSeek-V4 Pro	Gemini 3.1 Pro
VSD	[.543, .551, .579]	[.448, **.587**, .611]
LQT	[.575, **.598**, .614]	[.541, .568, .594]

We repeat VSD and LQT authoring with three exploration coefficients and two strong models from different families, Gemini 3.1 Pro (Google DeepMind, 2026) and open-weight DeepSeek-V4 Pro (DeepSeek-AI et al., 2026), using three seeds per setting. The three-seed results show task-dependent sensitivity rather than a universally optimal configuration. For exploration, $c = .25$ gives the highest median on VSD (.587), whereas $c = .50$ performs best on LQT (.596). Backbone choice shows a similar crossover: Gemini 3.1 Pro has the higher VSD median (.587 vs. .551), while DeepSeek-V4 Pro has the higher LQT median (.598 vs. .568). Thus, the observed gains are not specific to a single search setting or foundation-model backbone, although their relative effectiveness depends on the task.

6 Related Work

LM-guided program search and scientific software. Foundation models are increasingly embedded in execution-and-feedback loops that iteratively improve programs, with applications spanning mathematics, algorithm discovery, reward design, symbolic regression, and scientific software (Romera-Paredes et al., 2024; Novikov et al., 2025; Ma et al., 2023; Shojaee et al., 2024; Ma et al.,

2024; Aygün et al., 2025; Lehman et al., 2023). Related work on test-time scaling likewise studies how inference budgets should be divided between independent sampling and iterative refinement (Brown et al., 2024; Snell et al., 2024); our focus is how such refinement can accumulate when the feedback itself is only a weak proxy for artifact quality.

Agentic refinement and persistent state. Iterative agents improve outputs through repeated critique and revision (Madaan et al., 2023; Shinn et al., 2023), while other systems retain state across iterations in the form of skills, strategies, reusable code, or heuristics (Wang et al., 2023; Lehman et al., 2023; Ye et al., 2024). Prompt and textual optimization methods provide another form of persistent state, in which the object being updated is a natural-language prompt or strategy (Yang et al., 2024; Agrawal et al., 2025). SIMAUTHOR combines iterative refinement with artifact-centered persistence: the executable simulator itself is inherited and revised across calls, while a mechanism library provides additional reusable implementation knowledge.

Scientific simulation. Classical biomedical simulators encode explicit generative mechanisms with interpretable parameters, as in established ECG and PPG models (McSharry et al., 2003; Sološenko et al., 2017). Learned generators such as GANs and diffusion models instead approximate biomedical or audio signal distributions implicitly (Golany et al., 2020; Kong et al., 2020; Liu et al., 2023). Existing approaches therefore largely either hand-design an explicit simulator or learn an implicit generator from data. We study a complementary setting: a foundation model authors and iteratively refines the explicit, executable simulator itself from weak empirical feedback. The resulting artifact is an inspectable generative model rather than only a source of synthetic samples.

7 CONCLUSION AND DISCUSSION

This work establishes scientific simulator authoring as a viable use of foundation models under weak empirical feedback. Across six biomedical tasks, SIMAUTHOR generally outperforms independent sampling, PUCT score search, and textual-strategy optimization; its gains also persist on unseen recordings, alternative representations, and downstream ECG evaluation. Together with the revision audit, these results suggest that foundation-model knowledge can be progressively converted into better executable scientific simulators.

Limitations and future directions. Our evaluation focuses primarily on *intrinsic* agreement between simulated and observed signals, which does not establish physiological or mechanistic fidelity. Establishing such fidelity would also enable uses beyond distribution matching. In the forward direction, explicit simulator variables could be set to represent rare conditions or prescribed parameter regimes, producing targeted synthetic data under controlled assumptions (van Breugel et al., 2024). In the reverse direction, simulator parameters or latent mechanisms could be inferred by searching for settings whose simulated observations best match a given real recording, connecting authored simulators to simulation-based inference (Cranmer et al., 2020). Once sufficiently validated and calibrated to an individual, the same simulator could be perturbed to study how changes in specific mechanisms or interventions affect predicted observations, providing a possible route toward digital-twin applications (Björnsson et al., 2020; Corral-Acero et al., 2020). These capabilities are outside the present study; evaluating such *extrinsic* utility is an important direction for future work.

Authoring performance also depends on the evaluator: the model can only improve discrepancies that the feedback exposes. This may be particularly important in modalities where clinically relevant morphology is difficult to summarize with generic signal statistics, and may partly explain the weaker advantage of persistent authoring on AF. Future work should incorporate expert-designed features, physiological constraints, and complementary evaluators, and systematically study how feedback quality affects authoring. Our empirical scope is also limited to three seeds per setting and predominantly strong frontier models; testing smaller or less capable models would help characterize how persistent authoring interacts with foundation-model capability.

Finally, the framework is not specific to biomedical signals. The same setting arises wherever an executable simulator can be compared with observations, including industrial sensing and physical systems. Extending SIMAUTHOR to such domains, with domain-specific evaluation and stronger expert involvement, will test the broader generality of scientific simulator authoring.

REPRODUCIBILITY STATEMENT

The appendices report dataset identities and splits, evaluator definitions, the end-to-end authoring algorithm, complete prompts, numerical settings, multi-seed results, compute and token-matched controls, downstream ECG evaluation, and the revision-audit procedure. We will release the code for executing and evaluating simulators, all authored simulator programs and search traces needed to reconstruct the reported trajectories.

ETHICS STATEMENT

All biomedical datasets used in this study are publicly released, de-identified research datasets used under their original licenses; no new human-subject data were collected. The authored simulators and synthetic signals are research artifacts and may inherit biases or incidental properties of the source data and evaluators. We make no diagnostic or clinical claims. Physiological validation, patient-specific use, or applications such as intervention modeling would require independent expert and prospective validation beyond the scope of this work.

AI USE STATEMENT

Foundation models are experimental components of the method rather than auxiliary tools. Canonical SIMAUTHOR runs use Gemini 3.1 Pro for the authoring pipeline, including generation of the scientific blueprint and executable simulators; the backbone robustness experiment additionally uses DeepSeek-V4 Pro. All authoring prompts are reproduced in Appendix J. A separate DeepSeek-V4-Flash model is used only in the revision audit to help classify program changes after deterministic diff processing and to assess mechanism reuse. These annotations affect the audit analysis, but not the authoring trajectories, evaluator scores, or downstream results.

REFERENCES

Lakshya A. Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J. Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. *arXiv preprint arXiv:2507.19457*, 2025. URL https://arxiv.org/abs/2507.19457.

Shams Nafisa Ali, Afia Zahin, Samiul Based Shuvo, Nusrat Binta Nizam, Shoyad Ibn Sabur Khan Nuhash, Sayeed Sajjad Razin, SM Sakeef Sani, Farihin Rahman, Nawshad Binta Nizam, Farhat Binte Azam, et al. Buet multi-disease heart sound dataset: A comprehensive auscultation dataset for developing computer-aided diagnostic systems. *Computer Methods and Programs in Biomedicine Update*, pp. 100237, 2026.

Gökhan Altan, Yakup Kutlu, Yusuf Garbi, Adnan Özhan Pekmezci, and Serkan Nural. Multimedia respiratory database (respiratorydatabase@ tr): Auscultation sounds and chest x-rays. *Natural and Engineering Sciences*, 2(3):59–72, 2017.

Eser Aygün, Anastasiya Belyaeva, Gheorghe Comanici, Marc Coram, Hao Cui, Jake Garrison, Renee Johnston, Anton Kast, Cory Y McLean, Peter Norgaard, et al. An ai system to help scientists write expert-level empirical software. *arXiv preprint arXiv:2509.06503*, 2025.

Justinas Bacevičius, Vilma Pluščiauskaitė, Žygimantas Abramikas, Ignas Badaras, Monika Butkuvienė, Saulius Daukantas, Ernestas Dvinelis, Modestas Gudauskas, Edvardas Jukna, Margarita Kiseliūtė, Ričardas Kundelis, Julija Marinskienė, Birutė Paliakaitė, Andrius Petrėnas, Marija Petrylaitė, Aistė Pilkienė, Gintarė Pudinskaitė, Vytautas Radavičius, Andrius Rapalis, Daivaras Sokas, Andrius Sološenko, Justina Staigytė, Guostė Stankevičiūtė, Neringa Taparauskaitė, Gintarė Zarembaitė, Audrius Aidietis, and Vaidotas Marozas. Long-term electrocardiogram and wrist-based photoplethysmogram recordings with annotated atrial fibrillation episodes, 2024.

Bergthor Björnsson, Carl Borrebaeck, Nils Elander, Thomas Gasslander, Danuta R. Gawel, Mika Gustafsson, Rebecka Jörnsten, Eun Jung Lee, Xinxiu Li, Sandra Lilja, et al. Digital twins to personalize medicine. *Genome Medicine*, 12:4, 2020. doi: 10.1186/s13073-019-0701-3.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

Peter H Charlton, Kevin Kotzen, Elisa Mejía-Mejía, Philip J Aston, Karthik Budidha, Jonathan Mant, Callum Pettit, Joachim A Behar, and Panicos A Kyriacou. Detecting beats in the photoplethysmogram: benchmarking open-source algorithms. *Physiological Measurement*, 43(8):085007, 2022.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Jorge Corral-Acero, Francesca Margara, Maciej Marciniak, Cristobal Rodero, Filip Loncaric, Yingjing Feng, Andrew Gilbert, Joao F. Fernandes, Hassaan A. Bukhari, Ali Wajdan, et al. The 'digital twin' to enable the vision of precision cardiology. *European Heart Journal*, 41(48):4556–4564, 2020. doi: 10.1093/eurheartj/ehaa159.

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. doi: 10.1073/pnas.1912789117.

DeepSeek-AI, Anyi Xu, Bangcai Lin, Bing Xue, et al. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026. URL https://arxiv.org/abs/2606.19348.

Mohammad Fraiwan, Luay Fraiwan, Basheer Khassawneh, and Ali Ibnian. A dataset of lung sounds recorded from the chest wall using an electronic stethoscope. *Data in Brief*, 35:106913, 2021.

Tomer Golany, Kira Radinsky, and Daniel Freedman. Simgans: Simulator-based generative adversarial networks for ecg synthesis to improve deep ecg classification. In *International Conference on Machine Learning*, pp. 3597–3606. PMLR, 2020.

Google DeepMind. Gemini 3.1 Pro model card. https://deepmind.google/models/model-cards/gemini-3-1-pro/, 2026.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Yifan Wu, YK Li, et al. Deepseek-coder: when the large language model meets programming–the rise of code intelligence. *arXiv preprint arXiv:2401.14196*, 2024.

Weijie Jia, Yunyan Wang, Renwei Chen, Jingjing Ye, Die Li, Fei Yin, Jin Yu, Jiajia Chen, Qiang Shu, and Weize Xu. ZCHSound: Open-source ZJU paediatric heart sound database with congenital heart disease. *IEEE Transactions on Biomedical Engineering*, 71(8):2278–2286, 2024. doi: 10.1109/TBME.2023.3348800.

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2020.

Joel Lehman, Jonathan Gordon, Shawn Jain, Kamal Ndousse, Cathy Yeh, and Kenneth O. Stanley. Evolution through large models. In Wolfgang Banzhaf, Penousal Machado, and Mengjie Zhang (eds.), *Handbook of Evolutionary Machine Learning*, pp. 331–366. Springer Nature, Singapore, 2023. doi: 10.1007/978-981-99-3814-8_11.

Jun Li, Aaron D. Aguirre, Valdery Moura Junior, Jiarui Jin, Che Liu, Lanhai Zhong, Chenxi Sun, Gari D. Clifford, M. Brandon Westover, and Shenda Hong. An electrocardiogram foundation model built on over 10 million recordings. *NEJM AI*, 2(7):AIoa2401033, 2025. doi: 10.1056/AIoa2401033. URL https://doi.org/10.1056/AIoa2401033.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161*, 2023.

Haohe Liu, Zehua Chen, Yi Yuan, Xinhao Mei, Xubo Liu, Danilo Mandic, Wenwu Wang, and Mark D Plumbley. Audioldm: Text-to-audio generation with latent diffusion models. *arXiv preprint arXiv:2301.12503*, 2023.

Pingchuan Ma, Tsun-Hsuan Wang, Minghao Guo, Zhiqing Sun, Joshua B Tenenbaum, Daniela Rus, Chuang Gan, and Wojciech Matusik. Llm and simulation as bilevel optimizers: A new paradigm to advance physical scientific discovery. *arXiv preprint arXiv:2405.09783*, 2024.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*, 2023.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Patrick E McSharry, Gari D Clifford, Lionel Tarassenko, and Leonard A Smith. A dynamical model for generating synthetic electrocardiogram signals. *IEEE transactions on biomedical engineering*, 50(3):289–294, 2003.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

Arvind Pillai, Dimitris Spathis, Fahim Kawsar, and Mohammad Malekzadeh. PaPaGei: Open foundation models for optical physiological signals. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=qQebH9B2oT.

Aniruddh Raghu, Divya Shanmugam, Eugene Pomerantsev, John Guttag, and Collin M. Stultz. Data augmentation for electrocardiograms. In *Proceedings of the Conference on Health, Inference, and Learning*, volume 174 of *Proceedings of Machine Learning Research*, pp. 282–310. PMLR, 2022.

Nicola Rieke, Jonny Hancox, Wenqi Li, Fausto Milletarì, Holger R Roth, Shadi Albarqouni, Spyridon Bakas, Mathieu N Galtier, Bennett A Landman, Klaus Maier-Hein, Sébastien Ourselin, Micah J Sheller, Ronald M Summers, Andrew Trask, Daguang Xu, Maximilian Baust, and M Jorge Cardoso. The future of digital health with federated learning. *npj Digital Medicine*, 3(1):119, 2020. doi: 10.1038/s41746-020-00323-1.

BM Rocha, Dimitris Filos, Lea Mendes, Ioannis Vogiatzis, Eleni Perantoni, Evangelos Kaimakamis, P Natsiavas, Ana Oliveira, C Jácome, A Marques, et al. A respiratory sound database for the development of automated classification. In *International Conference on Biomedical and Health Informatics*, pp. 33–37. Springer, 2017.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Parshin Shojaee, Kazem Meidani, Shashank Gupta, Amir Barati Farimani, and Chandan K Reddy. Llm-sr: Scientific equation discovery via programming with large language models. *arXiv preprint arXiv:2404.18400*, 2024.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S. Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, Perry Payne, Martin Seneviratne, Paul Gamble, Chris Kelly, Nathanael Scharli, Aakanksha Chowdhery, Philip Mansfield, Blaise Aguera y Arcas, Dale Webster, Greg S. Corrado, Yossi Matias, Katherine Chou, Juraj Gottweis, Nenad Tomasev, Alan Karthikesalingam, and Vivek Natarajan. Large language models encode clinical knowledge. *Nature*, 620:172–180, 2023. doi: 10.1038/s41586-023-06291-2.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Andrius Sološenko, Andrius Petrėnas, Vaidotas Marozas, and Leif Sörnmo. Modeling of the photoplethysmogram during atrial fibrillation. *Computers in Biology and Medicine*, 81:130–138, 2017. doi: 10.1016/j.compbiomed.2016.12.016.

Boris van Breugel, Tennison Liu, Dino Oglic, and Mihaela van der Schaar. Synthetic data in biomedicine via generative artificial intelligence. *Nature Reviews Bioengineering*, 2:991–1004, 2024. doi: 10.1038/s44222-024-00245-7.

Patrick Wagner, Nils Strodthoff, Ralf-Dieter Bousseljot, Dieter Kreiseler, Fatima I Lunze, Wojciech Samek, and Tobias Schaeffter. Ptb-xl, a large publicly available electrocardiography dataset. *Scientific data*, 7(1):154, 2020.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Bb4VGOWELI.

Haoran Ye, Jiarui Wang, Zhiguang Cao, Federico Berto, Chuanbo Hua, Haeyeon Kim, Jinkyoo Park, and Guojie Song. ReEvo: Large language models as hyper-heuristics with reflective evolution. In *NeurIPS*, 2024.

Yuwei Zhang, Tong Xia, Jing Han, Yu Wu, Georgios Rizos, Yang Liu, Mohammed Mosuily, Jagmohan Chauhan, and Cecilia Mascolo. Towards open respiratory acoustic foundation models: Pretraining and benchmarking. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Jianwei Zheng, Hangyuan Guo, and Huimin Chu. A large scale 12-lead electrocardiogram database for arrhythmia study. *PhysioNet*, August 2022. doi: 10.13026/wgex-er52. URL `https://doi.org/10.13026/wgex-er52`. Version 1.0.0.

A Evaluator Definitions

Common interface. Each evaluator compares a set of generated recordings with a fixed reference set and returns a score in $[0, 1]$ together with a structured discrepancy report. The evaluator and all of its parameters are fixed before search. Candidates that fail to execute or violate the output contract are discarded before evaluation.

Audio: acoustic-feature overlap. Recordings are resampled to 16 kHz and peak-normalized. From each recording we extract the frame-averaged values of 13 MFCCs and the zero-crossing rate (ZCR), giving 14 features. Each feature is standardized using the reference-set mean and standard deviation. For feature k, we construct reference and generated histograms over shared bin edges and compute their overlap

$$o_k = \sum_b \min\left(\hat{p}_{kb}^{\mathrm{ref}}, \hat{p}_{kb}^{\mathrm{gen}}\right).$$

The final score is

$$s_{\mathrm{audio}} = \frac{1}{14} \sum_{k=1}^{14} o_k.$$

PPG: pulse-level feature similarity. Signals are processed at 125 Hz using their first 30 s. After a 0.5–10 Hz Butterworth bandpass, systolic peaks are detected with a minimum spacing of 300 ms. We extract ten segment-level features: heart rate, pulse-interval mean and standard deviation, RMSSD, systolic-amplitude mean and standard deviation, amplitude range, 25–75% rise time, pulse width at half amplitude, and pulse count. Non-physiological intervals outside 300–2000 ms are removed.

For each feature k, similarity is computed from the difference between the reference and generated means:

$$q_k = \exp\left[- \frac{2\left|\mu_k^{\mathrm{ref}} - \mu_k^{\mathrm{gen}}\right|}{\sqrt{\left((\sigma_k^{\mathrm{ref}})^2 + (\sigma_k^{\mathrm{gen}})^2\right)/2} + \epsilon} \right], \qquad s_{\mathrm{PPG}} = \frac{1}{10} \sum_{k=1}^{10} q_k.$$

After bandpass filtering, systolic peaks are detected using a fixed amplitude threshold and a minimum spacing of 300 ms.

ECG: frozen representation-distribution similarity. Each generated example contains two 10 s leads at 500 Hz. Lead I and lead II are placed in the corresponding channels of a 12-lead input, with the remaining channels set to zero. After high-pass filtering and per-lead standardization, a frozen ECGFounder encoder maps each recording to a 1,024-dimensional embedding (Li et al., 2025).

Let $\{x_i\}_{i=1}^n$ and $\{y_j\}_{j=1}^m$ denote the reference and generated embeddings. We compute the unbiased energy distance

$$D_{\mathrm{E}} = \frac{2}{nm} \sum_{i,j} \|x_i - y_j\|_2 - \frac{1}{n(n-1)} \sum_{i \neq i'} \|x_i - x_{i'}\|_2 - \frac{1}{m(m-1)} \sum_{j \neq j'} \|y_j - y_{j'}\|_2,$$

and transform it into a bounded search score:

$$s_{\mathrm{ECG}} = \frac{1}{1 + \max(D_{\mathrm{E}}, 0)/0.15}.$$

Discrepancy reports. The report makes the fixed objective interpretable to the refiner. For audio it lists each feature's overlap, standardized mean shift, and generated-to-reference spread ratio. For PPG it reports each feature's similarity and the corresponding means in physical units. For ECG it reports energy distance, centroid similarity, spread ratio, and within-set similarity as a mode-collapse diagnostic. In the full system, this report and a rendered comparison of three reference and three generated recordings are provided to the refiner.

B END-TO-END AUTHORING LOOP

Algorithm 1 End-to-end simulator authoring with SIMAUTHOR

Require: condition–modality task, search-reference set $\mathcal{D}_{\text{ref}}$, evaluator, budget $T = 100$, extraction threshold δ

$B \leftarrow$ BLUEPRINT() ▷ one model call; persistent context
$\mathcal{S}^{(0)} \leftarrow$ ROOTSIMULATOR() ▷ one independent model call
$\mathcal{T} \leftarrow \{\mathcal{S}^{(0)}\}$; $M \leftarrow \varnothing$ ▷ candidate pool, mechanism library
execute $\mathcal{S}^{(0)}$, score against $\mathcal{D}_{\text{ref}}$
for $t = 1, \ldots, T$ **do**
  $n_p \leftarrow \arg\max_{n \in \mathcal{T}} \left[\tilde{v}(n) + c_{\text{puct}}\, p(n)\sqrt{N_{\text{tot}}}/(1 + N(n))\right]$ ▷ select parent
  $c \leftarrow \big(B,\ \text{code}(n_p),\ \text{report}(n_p),\ \text{visual}(n_p),\ \text{last } k \text{ entries of } M\big)$ ▷ assemble context
  $\mathcal{S}_{\text{child}} \leftarrow$ REFINE(c) ▷ one model call: one focused revision
  if $\mathcal{S}_{\text{child}}$ executes and satisfies the modality output contract **then**
    execute $\mathcal{S}_{\text{child}}$; score and produce a discrepancy report against $\mathcal{D}_{\text{ref}}$
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{\mathcal{S}_{\text{child}}\}$ with a link to n_p
    if $\text{score}(\mathcal{S}_{\text{child}}) - \text{score}(n_p) \geq \delta$ **then**
      $M \leftarrow M \cup \{$EXTRACTMECHANISM$(n_p, \mathcal{S}_{\text{child}})\}$ ▷ one model call
    end if
  end if ▷ invalid children consume the attempt
end for
return best-so-far candidate at each budget

C MECHANISM LIBRARY

The mechanism library is a compact program-level memory of productive revisions. When a valid child program S_{n+1} improves its selected parent S_n by at least δ, an extraction call summarizes the program diff $S_n \to S_{n+1}$ as a normalized mechanism name, a behavioral description, and a representative code fragment. The score improvement and source revision are retained as provenance. Entries with the same normalized name are merged by incrementing their observation count. Before each refinement call, up to eight recent mechanisms are included as optional implementation suggestions. The refiner is instructed to reuse or adapt an entry only when it is relevant to the current parent program and discrepancy report. The library therefore transfers implementable discoveries across iterations without forcing every later program to contain every stored mechanism.

For example, one VSD revision replaced a severity-independent murmur distribution with a severity-conditioned generator. The child improved its parent from .3631 to .4971 ($\Delta = +.1341$), producing the following library entry:

Example mechanism-library entry

Name. `severity_dependent_murmur_spectrum`

Behavior. Vary the murmur frequency band and resonance with disease severity to represent the acoustic differences between restrictive and unrestrictive ventricular septal defects.

Representative code.

```
if severity < 0.5:
   murmur_low = np.random.uniform(100, 200)
   resonance = np.random.uniform(300, 500)
else:
   murmur_low = np.random.uniform(50, 100)
   resonance = np.random.uniform(150, 250)
```

Provenance. One parent–child revision, .3631 → .4971 ($\Delta = +.1341$).

A later refiner receives the entry under a *Reusable Mechanisms* heading, together with the instruction that it may reuse, modify, or ignore the mechanism. The exact extraction prompt, storage format, and complete per-run libraries are released with the code.

D IMPLEMENTATION DETAILS

Search. We use a flat candidate pool with parent links and rank-normalized PUCT, with $c_{\text{puct}} = 0.25$ and a uniform prior. Each trajectory receives 100 authoring attempts.

Mechanism library. Mechanism extraction is triggered when a valid child improves over its parent by at least $\delta = 0.02$. Up to the eight most recent library entries are included in each refinement prompt.

Foundation models. Canonical experiments use Gemini 3.1 Pro (`gemini-3.1-pro-preview`) (Google DeepMind, 2026). The backbone robustness experiment uses DeepSeek-V4 Pro (`deepseek-v4-pro`) (DeepSeek-AI et al., 2026), while keeping the authoring harness and evaluation protocol fixed.

Generation settings. Blueprint generation uses temperature 0.2; root-simulator generation and refinement use temperature 0.3. Generated simulators expose explicit random seeds so that their outputs can be reproduced.

Visual comparison. Each refinement call in the SIMAUTHOR configuration receives a real-versus-synthetic visualization containing three representative samples generated with a fixed simulator seed. The same visual feedback is provided to Text-Opt. Failure to render the visualization does not invalidate an authoring attempt.

Waveform contracts. Each candidate generates 100 outputs per evaluation. Audio simulators produce 10-s signals evaluated after resampling to 16 kHz; PPG simulators produce 30-s signals at 125 Hz; and ECG simulators produce 10-s two-lead signals at 500 Hz. The output contract, including format, sampling rate, duration, sample count, and channel structure, is fixed throughout authoring.

Budget accounting. Invalid or non-executable revisions consume an authoring attempt. API timeouts and rate-limit failures are retried and do not consume the budget.

E BENCHMARK DATA

Table 6: **Reference data for the six-condition benchmark.** Search recordings are available during authoring; held-out recordings are used only for evaluation. Counts are recordings/windows, with unique participants in parentheses where available.

Condition	Split	Dataset	Modality	n
VSD	Search	ZCHSound (Jia et al., 2024)	Cardiac audio	60 (60)
	Held-out	ZCHSound	Cardiac audio	40 (40)
AS	Search	BMD-HS (Ali et al., 2026)	Cardiac audio	37 (5)
	Held-out	BMD-HS	Cardiac audio	23 (3)
COPD	Search	ICBHI 2017 (Rocha et al., 2017)	Resp. audio	65 (25)
	Held-out	KAUH + TR (Fraiwan et al., 2021; Altan et al., 2017)	Resp. audio	99 (45)
AF	Search	MIMIC PERform AF (Charlton et al., 2022)	PPG	96 (12)
	Held-out	Long-term AF PPG (Bacevičius et al., 2024)	PPG	40 (8)
LQT	Search	PTB-XL (Wagner et al., 2020)	ECG	95 (86)
	Held-out	Chapman–Shaoxing–Ningbo (Zheng et al., 2022)	ECG	394
WPW	Search	PTB-XL (Wagner et al., 2020)	ECG	60 (49)
	Held-out	Chapman–Shaoxing–Ningbo (Zheng et al., 2022)	ECG	72

F MULTI-SEED RESULTS

Unless otherwise stated, all stochastic authoring experiments are run with three random seeds. For each method–task pair in the main comparison, we rank the three runs by their final best-so-far score $S_{\max}@100$ and use the median-scoring run for main-text reporting. Figure 3 shows the complete trajectory of this representative run, while Table 1 reports its endpoint statistics. This appendix reports the complete seed-level results underlying the main comparison, focused ablations, sensitivity analyses, and held-out evaluation.

F.1 MAIN AUTHORING RESULTS ACROSS SEEDS

Table 7 reports the final $S_{\max}@100$ for all three runs underlying the main benchmark. The representative results in Table 1 and Figure 3 correspond to the run with the median final score for each method–task pair.

Table 7: **Seed-level main authoring results.** Final best-so-far score $S_{\max}@100$ for all three random seeds. For each method–task pair, the main paper reports the run with the median final score; Figure 3 shows the complete trajectory of the corresponding representative run. The final column gives the sample standard deviation across seeds. Root $\mathcal{S}^{(0)}$ and Independent Sampling are fixed references reported in Table 1.

Task	Method	Seed 1	Seed 2	Seed 3	Std. dev.
VSD	PUCT Score Search	.365	.356	.337	.014
	Text-Opt	.416	.512	.346	.083
	SIMAUTHOR	.587	.448	.611	.088
AS	PUCT Score Search	.254	.247	.227	.014
	Text-Opt	.370	.412	.504	.069
	SIMAUTHOR	.429	.559	.557	.074
COPD	PUCT Score Search	.577	.455	.398	.091
	Text-Opt	.529	.588	.549	.030
	SIMAUTHOR	.672	.521	.689	.092
AF	PUCT Score Search	.549	.640	.560	.050
	Text-Opt	.771	.806	.801	.019
	SIMAUTHOR	.748	.738	.738	.006
LQT	PUCT Score Search	.395	.655	.532	.130
	Text-Opt	.554	.501	.582	.041
	SIMAUTHOR	.541	.568	.594	.027
WPW	PUCT Score Search	.550	.516	.589	.037
	Text-Opt	.364	.548	.526	.100
	SIMAUTHOR	.581	.543	.653	.056

F.2 Focused ablations across seeds

Table 8 expands the focused ablations from the main paper to all three random seeds. We report both endpoint quality, $S_{\max}@100$, and normalized AUC_{100} of the best-so-far trajectory, which captures performance across the full authoring budget.

Table 8: **Seed-level focused ablations.** Each cell reports Seed 1 / Seed 2 / Seed 3. **(a)** Final best-so-far score $S_{\max}@100$. **(b)** Normalized AUC_{100} of the best-so-far trajectory. LQT and WPW use the final ECGFounder representation-space evaluator.

(a) $S_{\max}@100$

Configuration	VSD	AS	COPD	AF	LQT	WPW
SimAuthor	.587/.448/.611	.429/.559/.557	.672/.521/.689	.748/.738/.738	.541/.568/.594	.581/.543/.653
− Report	.417/.366/.270	.152/.452/.152	.603/.609/.648	.638/.548/.589	.448/.460/.463	.491/.497/.492
− Library	.399/.409/.283	.383/.447/.287	.662/.704/.540	.719/.725/.703	.627/.540/.488	.573/.555/.623

(b) Normalized AUC_{100}

Configuration	VSD	AS	COPD	AF	LQT	WPW
SimAuthor	.359/.395/.497	.415/.461/.458	.544/.481/.599	.699/.723/.717	.521/.539/.549	.574/.528/.614
− Report	.356/.359/.269	.143/.355/.141	.456/.589/.585	.574/.501/.556	.439/.447/.431	.465/.486/.492
− Library	.348/.394/.282	.327/.324/.281	.528/.559/.529	.668/.686/.687	.576/.514/.474	.547/.522/.599

F.3 Held-out evaluation across seeds

Held-out evaluation follows a different aggregation protocol because held-out recordings must not influence model or seed selection. For each authoring seed, we evaluate the simulator selected solely by its search score, without access to the held-out set. Table 9 therefore reports all three independent runs and their column-wise median; no seed is selected using held-out performance.

Table 9: **Seed-level held-out evaluation.** Each SimAuthor row evaluates the search-score-best simulator from one independent authoring seed; held-out recordings are never used for node or seed selection. Root and best-of-100 sampling are fixed references. The final row reports the column-wise median across the three SimAuthor runs and gives the values reported in Table 4.

Method	VSD	AS	COPD	AF	LQT	WPW
Root $\mathcal{S}^{(0)}$	.2229	.1027	.0875	.1044	.2771	.3848
Sampling	.4246	.1460	.3624	.1173	.3408	.4433
SimAuthor seed 1	.5771	.4264	.6109	.2040	.4926	.4965
SimAuthor seed 2	.4093	.4402	.4095	.2428	.4245	.5379
SimAuthor seed 3	.5786	.4737	.5072	.1813	.4804	.6155
SimAuthor median	**.5771**	**.4402**	**.5072**	**.2040**	**.4804**	**.5379**

G COMPUTE AND TOKEN-MATCHED CONTROLS

All methods in the main comparison receive the same budget of 100 authoring attempts, but their model inputs differ substantially in length. In particular, SIMAUTHOR carries the selected program, scientific blueprint, discrepancy feedback, and mechanism memory across revisions. We therefore audit token usage and additionally compare against independent sampling under a matched token budget.

G.1 COMPUTE ACCOUNTING

Table 10 reports the model usage of a 100-attempt run, averaged over conditions and completed seeds. Evaluation cost is identical per candidate and is excluded. SIMAUTHOR uses approximately $3.8\times$ the total tokens of independent sampling, almost entirely because of its larger input context; output-token usage remains comparable. This corresponds to token parity with best-of-100 sampling after roughly 26 SIMAUTHOR attempts, or to an independent-sampling budget of approximately 380 proposals.

Table 10: **Compute per 100-attempt run.** Mean $\pm$ s.d. across conditions and completed seeds. Token counts are estimated from character counts and exclude image tokens.

Method	Calls	Input (k)	Output (k)	Total (k)
Independent Sampling	100.0	46 ± 1	185 ± 22	231
PUCT Score Search	100.0	314 ± 65	270 ± 68	584
SIMAUTHOR	106.4 ± 1.4	634 ± 68	244 ± 71	878

The additional SIMAUTHOR calls are auxiliary blueprint generation and mechanism extraction after productive revisions. Because token counts are estimated rather than taken from provider usage metadata, the absolute values should be interpreted approximately; the large difference in input context is the relevant comparison. Image inputs used by SIMAUTHOR are not counted, so its reported input usage is a lower bound.

G.2 TOKEN-MATCHED INDEPENDENT SAMPLING

To test whether the gain can be explained by the larger inference budget alone, we expand independent sampling from 100 to 380 proposals per task, matching the approximate token budget of a 100-attempt SIMAUTHOR run.

Table 11: **Independent sampling at a token-matched budget.** Best-of-N over an approximately 380-program independent pool, compared with SIMAUTHOR at its standard 100-attempt budget. "Index of best" gives the position of the highest-scoring program within the sampling sequence.

	VSD	AS	COPD	AF	LQT	WPW
Best-of-100	.477	.184	.391	.492	.418	.456
Best-of-380	.479	.298	.475	.530	.505	.458
index of best	118	371	254	324	313	338
SIMAUTHOR	**.587**	**.557**	**.672**	**.738**	**.568**	**.581**

Increasing the independent pool from 100 to 380 improves the best score only modestly on most tasks, while SIMAUTHOR remains higher on all six. The largest sampling gain occurs on AS, whose best proposal appears late in the pool, but a substantial gap to SIMAUTHOR remains. Thus, the advantage of iterative authoring is not explained by the additional token budget alone.

H DOWNSTREAM ECG AUGMENTATION

We provide additional details for the downstream ECG experiment in Section 4.2. The experiment tests whether ECGs produced by an authored simulator improve classification of real recordings, rather than merely matching the representation-space objective used during authoring.

Classification setup. We consider LQT and WPW as binary classification tasks. All conditions use the same real PTB-XL training set, frozen ECGFounder representation, and class-weighted logistic-regression probe. Evaluation is performed on a patient-disjoint PTB-XL split (ID) and on Chapman–Shaoxing as an external OOD dataset. Augmented conditions add exactly 300 positive training examples; all other representation extraction and classifier-training settings are held fixed.

Conventional waveform augmentation. As a non-generative augmentation baseline, we sample positive LQT or WPW training recordings with replacement and generate 300 additional positives. For each sampled recording, one of three waveform perturbations is selected with equal probability:

$$\text{Gaussian noise:} \quad x' = x + \epsilon, \qquad \epsilon \sim \mathcal{N}\left(0, (0.01\sigma_x)^2\right), \tag{4}$$

$$\text{Baseline wander:} \quad x' = x + 0.05\sigma_x \sin(2\pi f t), \qquad f \sim \mathcal{U}(0.1, 0.5)\ \text{Hz}, \tag{5}$$

$$\text{Amplitude scaling:} \quad x' = ax, \qquad a \sim \mathcal{U}(0.9, 1.1), \tag{6}$$

where σ_x is the standard deviation of the source waveform. These operations are common ECG augmentation primitives and are contained in the augmentation space studied by Raghu et al. (Raghu et al., 2022). Our baseline is a fixed, simplified subset of that augmentation family rather than a reproduction of their learned TaskAug policy.

SimAuthor augmentation and selection. For the SimAuthor condition, the 300 additional positive examples are sampled from an authored simulator rather than obtained by perturbing existing recordings. Among candidates produced across the three independent authoring seeds, the simulator is selected solely by its representation-space authoring score.

Evaluation metrics. The main paper reports AUPRC in Table 2. We additionally report AUROC in Table 12. Both metrics are computed from the same trained probes and evaluation sets. AUROC reproduces the AUPRC pattern: SimAuthor gives the strongest result in all four ID/OOD comparisons, with the largest improvements occurring on the external OOD evaluation.

Table 12: **AUROC for downstream ECG classification.** All conditions use frozen ECGFounder representations and the same class-weighted logistic-regression probe. "+ Conv." and "+ SimAuthor" augment the real training set with 300 positive examples. The SimAuthor program is selected using only its representation-space authoring score across three independent authoring runs; downstream performance is never used for selection. ID denotes patient-disjoint PTB-XL and OOD denotes Chapman–Shaoxing. Bold indicates the best result in each column.

	LQT		WPW	
Training	**ID**	**OOD**	**ID**	**OOD**
Real	.965	.797	.951	.926
+ Conv.	.968	.821	.953	.926
+ SimAuthor	**.974**	**.889**	**.973**	**.965**

I REVISION AUDIT

Each 100-attempt SIMAUTHOR run produces a branching tree of simulator candidates. Our audit does not treat all attempts as revisions of the final artifact. Instead, for each of the 18 runs, we trace the ancestry of the selected simulator back to the root and audit every valid, scored parent–child transition on that lineage. These selected lineages contain 138 revisions in total.

For each revision, added and removed lines are first aligned deterministically. Edits that become identical after masking numeric literals are marked as scalar changes; a separate labelling model (DeepSeek-V4-Flash) then determines whether the overall behavioral change is pure parameter tuning or structural. All audited programs were authored by Gemini 3.1 Pro. The structural-gain statistic in Figure 4 is computed directly from the signed parent–child score deltas after this partition.

Mechanism reuse is assessed separately by matching stored mechanism code against later programs and verifying that the matched code implements the same behavior. By this criterion, 69 of 86 extracted mechanisms are reused. We will release all authored simulator scripts together with the search lineages and audit metadata, enabling future work to inspect revisions and alternative branches directly.

J PROMPTS

Below are the complete prompts used by SIMAUTHOR. Four are issued during authoring, in the order they occur in algorithm 1: the scientific blueprint (Prompt 1, Line 1), the root simulator (Prompt 2, Line 2), refinement (Prompt 3, Line 8), and mechanism extraction (Prompt 4, Line 13). No prompt was tuned per condition; the same four templates are used for all six tasks.

Braced tokens such as `{condition_name}` are runtime slots filled by the harness. The refinement prompt is the one that carries persistence, and its slots map one-to-one onto the context tuple assembled in Line 8 of algorithm 1: `{blueprint}` is B, `{parent_code}` is $\mathrm{code}(n_p)$, `{feedback_report}` is $\mathrm{report}(n_p)$, and `{mechanism_context}` is the most recent at most eight entries of M (Appendix D). The visual comparison $\mathrm{visual}(n_p)$ is attached to the same call as an image rather than as text, so it does not appear in the template.

The `{modality_specification}` slot is a fixed per-modality document stating the immutable output contract: file format, sampling rate, duration, sample count, channel or lead structure, and naming convention. It is identical across all conditions within a modality.[1]

[1]The four modality specifications (cardiac audio, respiratory audio, ECG, PPG) are not reproduced here because they are configuration rather than instruction: they restate the waveform contracts already summarised in Appendix D and are held fixed throughout every run and every method. They are released with the code, together with the per-condition blueprints these prompts produce, all model responses, and complete run traces.

J.1 SCIENTIFIC BLUEPRINT

Prompt 1: Scientific blueprint generation. One call per task, issued before any recording is consulted.

Produce a comprehensive Scientific Blueprint for `{condition_name}` (`{signal_modality}` signal). This document will be the sole clinical reference for a DSP team building a computational simulator.

Modality Specification: `{modality_specification}`

Required Sections

1. Condition Overview
- Definition, pathophysiology, and clinical significance.
- Key subtypes or severity gradations (if applicable).
- Prevalence and typical patient demographics.

2. Signal Manifestation
Describe in quantitative detail how `{condition_name}` manifests in the `{signal_modality}` domain. Follow the Modality Specification above for which properties to characterise and at what level of detail.

3. Severity and Variability
- How the signal characteristics change with condition severity or disease stage.
- Sources of natural variability (subject-to-subject, beat-to-beat, recording-condition, acquisition-related, or other).

4. Key Discriminators
- The 3–5 most important features that distinguish this condition from similar conditions in the same modality.
- For each: typical range, how it differs from confounds.
- Include a differential diagnosis table comparing against 2–4 similar conditions.

5. Synthesis Considerations
- Recommended computational approach for reproducing this condition in the `{signal_modality}` domain.
- Concrete parameter ranges the DSP engineer should target.
- Common pitfalls and how to avoid them.

Output Format. Write in clear, technical markdown. Use tables for comparisons and reference values. Be SPECIFIC with numbers. If you are uncertain about a value, give your best estimate and mark it with (est.). This document should be 800–1500 words — comprehensive but focused.

J.2 ROOT SIMULATOR

Prompt 2: Root simulator $\mathcal{S}^{(0)}$. One independent call; no discrepancy report exists yet.

Construct a self-contained Python simulator for:

- **Clinical condition:** `{condition_name}`
- **Signal modality:** `{signal_modality}`

Simulate how the target condition manifests specifically in the requested signal modality. Use your biomedical and computational knowledge to identify the principal physiological or signal-generating mechanisms and implement them as an executable simulator.

The simulator should provide a simple but scientifically meaningful starting point for later refinement. It should:

- organize the synthesis into interpretable components;
- expose important physiological and simulation parameters explicitly;
- generate exactly `{num_samples}` diverse samples;
- represent plausible variation in severity, physiology, subjects, and acquisition where appropriate;
- avoid near-identical outputs that differ only through trivial noise;
- remain modular, robust, and easy to revise;
- use an explicit random seed for reproducibility.

Modality Specification: `{modality_specification}`

Output. Return only a complete executable Python script in one fenced `python` code block. Do not include any text outside the code block. The script must run without manual modification as:

```
python script_000.py
```

J.3 REFINEMENT

Prompt 3: Refinement. One call per authoring attempt; this is the call that carries persistence.

You are a senior DSP engineer. Your task is to improve a biomedical signal simulator by applying one focused, hypothesis-driven refinement.

Task Identity. Clinical condition: `{condition_name}`. Signal modality: `{signal_modality}`.

Scientific Reference. `{blueprint}`

Immutable Output Contract. `{modality_specification}`

You may change simulator mechanisms and internal parameters, but you must NOT change the output format, sampling rates, duration, sample count, channel or lead structure, units, filename convention, or expected waveform shape defined in the contract above. The number of generated recordings, output format, sampling rate, duration, and output interface are controlled by the framework and are not simulator design variables. Do not modify them during refinement.

Current Simulator (`script_{parent_idx}`):

```
```python
{parent_code}
```
```

Numerical Discrepancy Report. `{feedback_report}`

Mechanism Library. `{mechanism_context}`

Refinement Policy

1. Study the Numerical Discrepancy Report carefully. Identify the most plausible causes of the reported discrepancies in the current simulator.
2. Formulate ONE primary scientific hypothesis about what change would most effectively address the largest or most scientifically meaningful discrepancy.
3. Implement a FOCUSED MUTATION that tests this hypothesis:
   - Change only what is necessary to test the hypothesis.
   - Preserve all unaffected working components exactly as they are.
   - Do NOT rewrite the entire script unless the parent is fundamentally invalid (non-executable, wrong modality, violates the output contract).
   - Each refinement should represent one main scientific change, not a collection of unrelated edits.
4. The final output must respect the Immutable Output Contract above.

Important: Role of the Numerical Discrepancy Report. The Search Reference Set contains a limited sample of real recordings. It provides empirical grounding but does not exhaustively define the target distribution. Use the discrepancy report to identify clear unrealistic behaviours (e.g., missing physiological features, implausible morphologies, severe mode collapse). However, scientifically plausible variability described in the Scientific Blueprint should be preserved even if it is not fully represented in the finite reference sample. Do not simply imitate the reference set statistics. Address genuine discrepancies while respecting the Scientific Blueprint's broader description of the condition.

Output Format. Return ONLY the complete, self-contained Python script in one fenced `python` code block. Do not include explanations, commentary, or Markdown outside the code block.

J.4 MECHANISM EXTRACTION

Prompt 4: Mechanism extraction. One call per accepted improvement, i.e. when child − parent $\geq \delta$.

Compare two versions of a biomedical signal synthesis script. The child achieved a better evaluation than the parent.

Identify ONE plausible reusable behavioral change associated with the improvement.

Focus on substantive changes to signal generation, physiological modeling, parameterization, temporal structure, spectral structure, waveform morphology, or stochastic variation.

Ignore changes limited to formatting, comments, logging, file paths, code organisation without behavioral effect, or error handling.

Use the unified diff below as the primary evidence:

```
```diff
{diff}
```
```

Return ONLY a single-line JSON object (no Markdown, no literal newlines inside string values):

```
{"name": "snake_case_3_5_words",
 "mechanism": "one-sentence description",
 "code_snippet": "key function or short description"}
```