A Standard Physical Environment for Benchmarking AI-driven Cell Biology

Yang Choo [1] **Carter Allen** [1] **Jimmy Sastra** [1]

[1]*Monomer Bio, 1338 Mission Street, San Francisco*. Correspondence to: Yang Choo yang@monomerbio.com.

1. Introduction

Scientific research is shifting from human-driven hypothesis generation toward computational autonomy, often termed "Agentic Science." While traditional high-throughput screening industrialized the execution of experiments, the reasoning and decision-making have remained manual. Today, AI agents are poised to close this loop by conceiving experiments, interpreting results, and iteratively refining their understanding of biological systems.

There have been various demonstrations of these AI agents in biology [1][2][3] but comparing their effectiveness is currently impossible since they are implemented in dissimilar environments and solve different applications, or are expensive custom build setups. There is no "ImageNet" or "DARPA Grand Challenge" equivalent for physical wetlab biology: no standardized environment where an AI agent can be rigorously benchmarked and compared on its ability to manipulate biology, manage resources, and optimize outcomes in real time. Consequently, the field faces a "Sim-to-Real" gap, where agents that perform well in simulated environments fail when confronting the stochastic and physical constraints of a real laboratory.

This work proposes a standardized yet generalizable physical setup to evaluate AI agents on cell culture protocol optimization. By standardizing 1) the cell type, 2) the container type, and 3) the lab automation hardware, we can establish a standardized test bed to compare Active Learning (AL) and Reinforcement Learning (RL) approaches. We choose cell culture because it preserves system-level biological complexity while remaining tractable for high-iteration experimentation in a physical lab with real physical and cost constraints. The proposed benchmark tests not merely an agent's ability to predict a value, but its ability to suggest practical experiments, and generate meaningful scientific data empirically.

2. Requirements

To serve as an effective benchmark, the biological cell type must be fast-growing to enable many design-build-test-learn (DBTL) cycles and be low-cost to allow for high-volume experimentation. Safety requirements must be minimal, and the automation hardware and container formats must be standardized, off-the-shelf, and interoperable to facilitate broad adoption.

2.1 Model Organism

We selected *Vibrio natriegens* as the model organism due to its world-record doubling time of approximately 9.8 minutes under optimal conditions [4]. This rapid growth enables 6 sequential DBTRL cycles within a single working day in a 96-well plate, compared to the 2 cycles possible with *E. coli*.

Furthermore, *Vibrio* also possesses immense metabolic versatility, creating a high-dimensional combinatorial search space of media components and environment (carbon source, salinity, pH) ideal for testing active learning agents. Finally, unlike *E. coli*, the optimal culture conditions for *V. natriegens* are less characterized, reducing the risk of models relying on training data memorization rather than active discovery.

2.2 Container Type

We selected SBS-formatted microplates as a widely available container type commonly used in R&D experimentation. In particular, we found the 96-well flat and optically clear bottom plates to be well-suited for *V. natriegens*. This enabled automated plate reads without requiring a separate liquid transfer step.

2.3 Hardware

The selected lab automation hardware consists of standard, readily available instrument categories: a heating incubator (30°C), a cooling incubator (4°C) for media storage, a liquid handler for media changes and passaging, a plate reader for measuring growth via absorbance (OD600), and a robotic arm to move microplates and lids. Specific instrument models are listed in Appendix A, but the setup is designed to be agnostic to specific instrument models to allow benchmarking across different platforms should one wish to use a different model platereader or liquid handler for example.

Crucially, this physical setup introduces realistic scheduling constraints, such as the time plates spend outside incubators during operations, which are often ignored in in-silico benchmarks but significantly impact biological outcomes.

2.4 Software

To facilitate AI agents interacting with the lab automation hardware and software, we utilize the Model Context Protocol (MCP) [5]. Rather than relying on rigid, atomic commands, MCP abstracts the robotic workcell into Workflows, Routines, and Plates. The AI agent interacts with these tool definitions (see Appendix B) rather than low-level drivers. This universal interface allows general-purpose agents to "read" the lab's capabilities and generate valid experimental plans without specialized training on specific instrument drivers.

3. Validation and Case Studies

To validate the proposed standard, we conducted the 2025 "24-Hour Mystery Cell Culture Challenge". Teams were challenged to optimize the growth rate of *V. natriegens* starting from a baseline by manipulating variables such as seeding volume, mixing parameters, and media composition.

3.1 Infrastructure Validation

During the pilot study, diverse teams successfully executed closed-loop autonomous campaigns using the described setup, demonstrating the system's stability and extensibility.

- **Architecture Robustness:** The MCP setup proved robust, enabling diverse teams to build functional "self-driving" loops in under 24 hours.
- **Ecosystem Composability:** The modular architecture supported a heterogeneous software ecosystem. Teams successfully integrated specialized scientific agents (e.g., PotatoAI, FutureHouse) and leveraged diverse foundation models (Anthropic, OpenAI), often utilizing coding assistants like Cursor/Claude Code to accelerate development.
- **Agent Diversity:** Successful implementations ranged from classical Bayesian Optimizers to LLM-driven "Autonomous Researchers" that generated Python scripts to interact with workflow definitions in real-time.
- **Safety Layers:** A "Safety Sandbox" layer successfully intercepted and validated workflow parameters against physical constraints (e.g., volume limits) prior to execution, ensuring safe autonomous operation.

3.2 Growth Strategy Optimization Challenge

Two teams in particular highlighted different approaches to the challenge constraints, which required optimizing both biological parameters and operational timing (intra-plate passaging and reads) on a single 96-well plate. One team utilized a Bayesian optimization architecture (Fig. 1), while the other deployed a multi-agent architecture (Fig. 2).

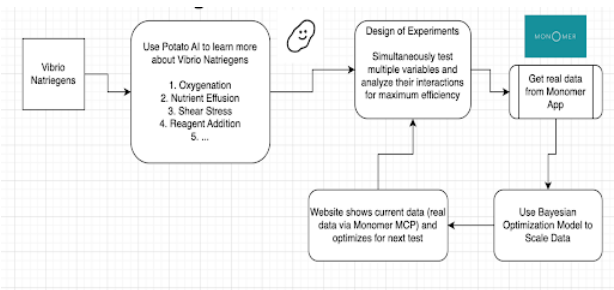

Fig. 1: Bayesian optimization architecture.

Both teams launched their first experiments within approximately 4 hours and completed 5 to 6 iterations over the subsequent 20 hours. Despite the inherent noise of physical biology, both managed to achieve significant growth, reaching peak densities of 9.9×10^8 and 7.6×10^8 cells/mL while autonomously executing DBTL cycles.

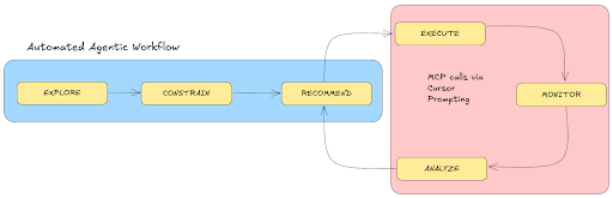

Fig. 2: Multi-agent architecture.

3.3 Benchmarking active learning strategies

Building on the pilot study, we are collaborating with external participants to more formally benchmark the performance of two distinct active learning strategies: the Automated Recommendation Tool (ART), which utilizes a Bayesian ensemble [6], and Optuna, utilizing Tree-structured Parzen Estimators (TPE) [7].

Both agents will optimize a 13-component media formulation space starting from identical Latin Hypercube Sampling (LHS) initialization data. We will analyze the transition from random initialization to optimized prediction using three key metrics:

1. **Top-3 Average Growth:** The mean optical density of the top three distinct formulations, prioritizing robust identification of high-performing regions over single-point outliers, which can be prone to biological noise in physical setups.
2. **Fold-Change Improvement:** The ratio of the agent's optimized growth rate compared to the standard LB broth baseline, quantifying the practical biological utility of the optimization campaign.
3. **Hit Rate per Batch:** The percentage of experiments in each iteration exceeding baseline performance, serving as a proxy for the agent's learning velocity and sample efficiency.

4. Conclusion

By defining a standard cell line, container format, automation setup, and algorithmic baseline, this work provides the necessary infrastructure to transition AI from passive data analysis to active, embodied scientific discovery. This benchmark offers the community a standardized arena to objectively compare the real-world efficiency of autonomous agents, addressing the critical "Sim-to-Real" gap in biological AI.

Acknowledgments

We thank the many participants in the challenge. The event attracted substantial interest, with 130 individuals registering to participate. Due to physical space constraints, participation was limited to six groups of five individuals. Attendees represented a diverse set of backgrounds, including machine learning and laboratory automation engineers from AI-driven biotech companies (e.g., Lila Biosciences), laboratory automation engineers and scientists from traditional biopharmaceutical organizations (e.g., insitro, BigHat, Eikon, and Merck). Notably, none of the participants had prior experience with the robotic platform or software system used in this study.

In particular, the following teams were referenced in this abstract:

- **Team ScaleMeMaybe:** Keltoum Boukra, Srusti Sain, Hanah Rahman, Julie Penzotti.
- **Team Bacter-AI:** Simon Coelho, Akash Ramachandran, Kanupriya Daga, Mandana Askarizadeh, Max Fereydouni, Vivek Ramaswamy.

We also thank the team at **Novel Bio** for providing the NBx CyClone strain, which served as our *V. natriegens* model organism for all experiments.

References

[1] OpenAI. Measuring ai's capability to accelerate biological research in the wet lab. `https://openai.com/index/accelerating-biological-research-in-the-wet-lab/`, dec 2025. Accessed: 2026-01-17.

[2] C Cizauskas et al. How the past is shaping the future of life science: The influence of automation and ai on biology. *New Biotechnology*, 88:1–11, 2025.

[3] Akiko Nishi Ryo Suzuki Masahiro Ikegami Risa Nimura Taichi Tomono Ryota Hidese Hisashi Yasueda Yusuke Tagawa Tomohisa Hasunuma Keiji Fushimi, Yusuke Nakai. Development of the autonomous lab system to support biotechnology research. *Scientific Reports*, 15(6648), 2004.

[4] RG Eagon. Pseudomonas natriegens, a marine bacterium with a generation time of less than 10 minutes. *Journal of Bacteriology*, 83(4):736–737, 1962.

[5] Anthropic. Model context protocol (mcp). `https://github.com/model-context-protocol`, 2024. Accessed: January 15, 2026.

[6] Tijana Radivojević, Zak Costello, Kenneth Workman, and Hector Garcia Martin. A machine learning automated recommendation tool for synthetic biology. *Nature Communications*, 11(1):4879, 2020.

[7] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Appendix A. Instrument List

The configuration is designed to be hardware-agnostic, but specific models used in the pilot study included:

- Liquid Handler - Opentrons OT-2 with Single Channel P1000
- Robotic Arm - PAA KX-2
- Automated Refrigerator - Liconic STX-110
- Automated Incubator - Liconic STX-220
- Automated Spectrophotomer - Tecan Infinite M Plex
- Automated Storage - Liconic LPX-220

Appendix B. Full MCP Specification

The Model Context Protocol (MCP) abstracts the robotics orchestration and cell data layer into the following tools and resources.

2.1 Tools

Workflow Definitions

- `list_workflow_definitions`: List all workflow definition templates.
- `get_workflow_definition`: Get detailed info about a specific workflow definition.
- `get_workflow_definition_schedule`: Get scheduled nodes with relative execution times.
- `get_workflow_definition_dag`: Get DAG structure showing nodes and dependencies.
- `list_workflow_definition_files`: List workflow definition files on disk.
- `get_workflow_dsl_schemas`: Get simplified schemas for DSL classes.
- `create_workflow_definition_file`: Create workflow definition Python file.
- `validate_workflow_definition_file`: Validate definition before registration.

- `register_workflow_definition`: Register validated file in database.

Workflow Instance

- `list_workflow_instances`: List workflow instance executions.
- `get_workflow_instance_details`: Get detailed instance info including status.
- `list_workflow_routines`: List scheduled steps for a specific instance.
- `list_pending_workflows`: List workflows awaiting operator approval.
- `instantiate_workflow`: Create workflow instance for operator approval.
- `check_workflow_cancellable`: Check if workflow can be safely cancelled.
- `cancel_workflow_instance`: Cancel a workflow (requires user confirmation).

Routine

- `list_available_routines`: List all available default-type routines.
- `get_routine_details`: Get complete routine template and parameters.
- `list_future_routines`: List scheduled future routines with filtering.
- `get_future_routine_details`: Get complete future routine details.
- `get_workflow_routine_with_children`: Get WorkflowRoutine with child FutureRoutines.
- `trace_future_routine_to_workflow`: Trace Future Routine back to workflow context.
- `check_consumables_for_timeframe`: Check consumables needed for routines.

Plate

- `list_plates`: List all plates with observation summaries.
- `get_plate_details`: Get plate info and its cultures.
- `get_plate_observations`: Get observations grouped by timestamp.
- `list_culture_plates`: List culture plates with status and workflow assignment.
- `check_plate_availability`: Check if plate is available for workflow assignment.
- `unlink_culture_plate_from_workflow`: Unlink plate from its current workflow.
- `list_reagent_plates`: List reagent plates and their media types.

Culture

- `list_cultures`: List cultures with latest readings.
- `get_culture_details`: Get culture info and observation history.
- `list_culture_statuses`: List available status options.
- `update_culture_status`: Update status for well(s).

2.2 Resources

- `schema://workflows/models`: Database schema models.
- `schema://workflows/dsl-api`: Auto-generated API reference for workflow domain specific language.
- `doc://workflows/api`: REST endpoints for workflows.
- `guide://workflows/concepts`: Workflow concepts and execution flow.
- `guide://workflows/creation`: Quick start guide for creating workflows.
- `guide://workflows/dsl`: Complete DSL reference with examples.
- `example://workflows/ipsc-maintenance`: Complete working example workflow file.
- `guide://future-routines/architecture`: Future routines architecture and relationships.
- `guide://future-routines/monitoring`: Monitoring guide for AI agents.
- `schema://routines/parameters`: Routine parameter types reference.
- `doc://cultures-and-plates/api-usage`: API usage guide with examples.
- `guide://cultures-and-plates/concepts`: Domain concepts explanation.
- `schema://cultures-and-plates/model`: Database schema reference.