The Socratic Scientist: Designing LLM Agentic Harness for Long-Running Computational Science

Anonymous Authors[1]

Abstract

Autonomous agents for computational physical science must execute multi-day campaigns on shared HPC queues, span multiple sessions and human handoffs, and survive late-caught method errors that silently discard days of compute. Fast-loop ideation harnesses break down in this regime. We present **Superscientist**, an agentic harness with two distinguishing features: *Socratic human-in-the-loop gates* at exactly two moments (pre-flight specification and mid-campaign amendment), and *durability via session refresh*, in which a fresh process with no conversational context recovers a campaign by reading three on-disk files. The harness layers ten orchestration skills over a nineteen-skill `comp_chem` application library, dispatched through various backends (e.g., Shell, Slurm, PBS, LSF). Three end-to-end molecular-dynamics case studies evidence the design: Green–Kubo polymer-melt viscosity; non-equilibrium molecular dynamics (NEMD) for 3C-SiC thermal conductivity executed across two sessions after a mid-campaign amendment; and Bayesian-optimized polymer sequences.

1. Introduction

Recent AI-scientist harnesses target tight, low-stakes feedback loops: ideation pipelines that draft hypotheses and papers in minutes (Lu et al., 2024; Yamada et al., 2025), ML-research agents over benchmarks in controlled sandboxes (Nathani et al., 2025), chemistry agents over small reversible experiments (Bran et al., 2024), and reactive controllers (Yao et al., 2023) that close fast feedback loops. Computational physical science breaks the assumption: a single molecular-dynamics stage can run hours to days on shared HPC queues, realistic campaigns span multiple interactive sessions and human handoffs, and a method-applicability error caught late silently discards days of compute. A second wave of agentic systems targets this regime directly. El Agente for quantum chemistry (Zou et al., 2025), ChemGraph for computational chemistry (Pham et al., 2025), hierarchical multi-agent reasoners for functional materials (Rothfarb et al., 2025), end-to-end atomistic-simulation frameworks (Vriza et al., 2026; Xia et al., 2025), autonomous synthesis labs (Boiko et al., 2023; Szymanski et al., 2023), and agent-as-research-assistant systems (Schmidgall et al., 2025) contribute task-graph orchestration, reusable tool registries, and artifact-first evidence trails.

What remains underspecified across this wave is the combination of two design decisions that, together, determine whether a campaign survives a multi-day, multi-session, multi-handoff lifecycle: *where the human enters the loop*, and *how state survives session boundaries*. Reactive controllers (Yao et al., 2023) mutate an in-context plan without a durable trace of what changed, when, or why; workflow-specification systems (Wang et al., 2026) couple a YAML schema to a Python runtime, mediating the agent through a translation layer prone to silent semantic drift. We argue both choices should be deliberate. The agent should not narrate a per-stage "confirm?" gate that produces human-in-the-loop theater without an audit trail, and the campaign should not depend on any one session staying alive.

We present **Superscientist**, an agentic harness with two distinguishing features: *Socratic human-in-the-loop gates* at exactly two moments (pre-flight specification and mid-campaign amendment), and *durability via session refresh*, in which a fresh process with no conversational context recovers a campaign by reading three on-disk files. Three contributions. **(1)** A two-layer architecture: a ten-skill orchestration harness over a nineteen-skill `comp_chem` application-skill library, dispatched via DPDispatcher (Yuan et al., 2025) to Shell, Slurm, PBS, LSF, or Bohrium. **(2)** Socratic-only-where-it-matters HITL, with the deliberate absence of a per-stage gate as a design claim, not an oversight; all other transitions are deterministic and recorded in `progress.log`. **(3)** Three end-to-end MD campaigns, fully executed: Green–Kubo viscosity ($\eta = 16.02 \pm 0.44$ LJ), NEMD 3C-SiC

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

($\kappa_{\text{bulk}} = 8.87 \pm 0.06$ W/mK, two-session resume after a mid-campaign amendment), and Bayesian-optimized polymer sequences ($R_g = 1.4672 \pm 0.0016\,\sigma$, 3-round convergence).

2. System

Two layers, skills not a DSL. Superscientist is a Claude Code plugin of ten Markdown skills (seven phase skills for the Design → Plan → Execute → Verify → Complete pipeline; three cross-cutting: `session-resume`, `systematic-debugging`, `checkpoint-management`) sitting above a nineteen-skill `comp_chem` application-skill library that wraps domain tools (LAMMPS, ASE, MACE, PySCF, CP2K, `bayes-opt`, `polymer-build`, Materials Project, RDKit, ...) behind uniform Python entry points (Figure 1; full catalog in Appendix C). Skills are natural-language prompts, not orchestration code, so a domain scientist edits workflow logic in the same medium the agent executes it.

Three-file durability substrate. State lives on disk in three files. `workflow-state.json` owns the stage DAG, success criteria, amendment records, and running-process metadata. `progress.log` is append-only and ISO-8601 timestamped via `$(date -Iseconds)` at the *tool-call* level, never LLM-typed. This closes a failure mode in which models post-hoc invent times. `init.sh` is an idempotent environment bootstrap re-executed at every session start. No hidden in-memory state persists between sessions: a fresh process with no conversational context recovers a campaign entirely by reading these three files. Each stage executes in a clean-context subagent (Anthropic, 2025); compute is submitted via DPDispatcher (Yuan et al., 2025) unifying Shell, Slurm, PBS, LSF, and Bohrium behind one API. In asynchronous mode a `tmux` wrapper writes a `DPDISP_DONE` marker on job completion, so a later session resumes monitoring just by reading the marker (Appendix B).

3. Three Case Studies

Polymer-melt viscosity (Panel A). A five-stage Green–Kubo (Green, 1954; Kubo, 1957) workflow on a Kremer–Grest bead-spring melt (Kremer & Grest, 1990) of volume $5832\,\sigma^3$ ran to completion in a single local Shell session (9 h wall-clock, LAMMPS Aug-2023 (Thompson et al., 2022; Plimpton, 1995)). The stress-autocorrelation running integral plateaus at 100–300 τ, giving $\eta = 16.02 \pm 0.44$ LJ units averaged over five independent components. This is the quiet case: no amendments, no verification failures, no cross-session handoff. It evidences that harness overhead is negligible when a campaign proceeds as specified.

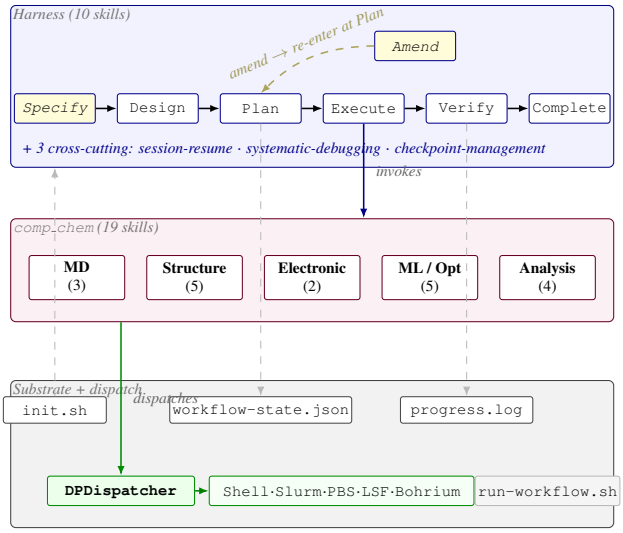
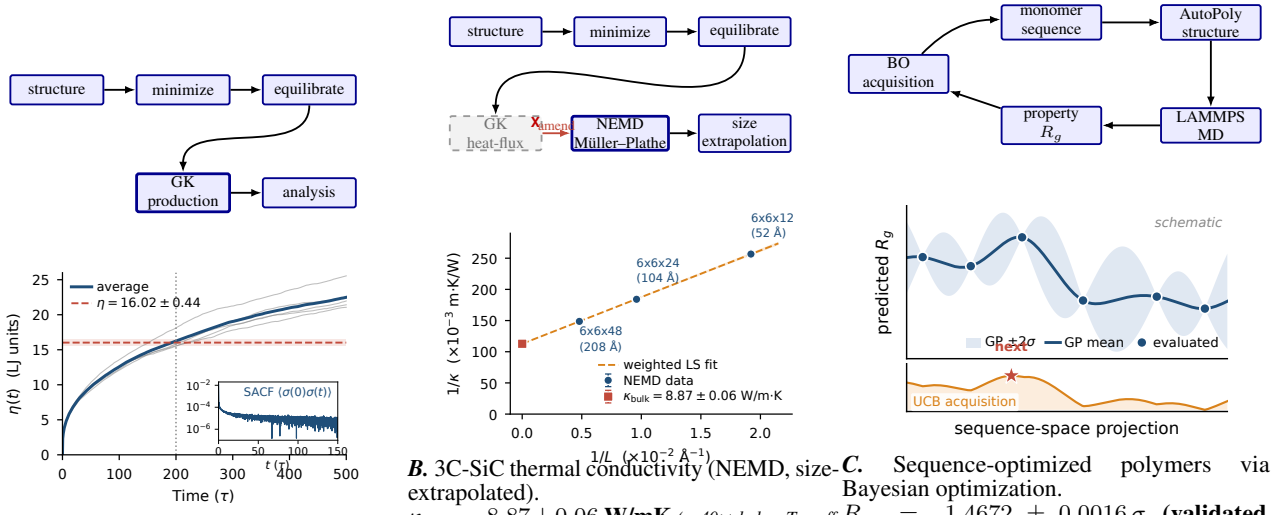


Figure 1. **Two-layer architecture.** *Top* (blue): Superscientist orchestration harness, ten skills realizing Design → Plan → Execute → Verify → Complete plus three cross-cutting skills, with two Socratic gates (Specify, Amend) marked. *Middle* (purple): `comp_chem` application-skill library, nineteen domain skills (Appendix C). *Bottom* (grey/green): three durability files plus DPDispatcher fanning compute to Shell/Slurm/PBS/LSF/Bohrium; an optional `run-workflow.sh` wrapper chains fresh `claude -p` sessions for hands-off multi-day runs.

3C-SiC thermal conductivity (Panel B). The initial design specified Green–Kubo with a Tersoff (Tersoff, 1989) potential. At stage 4 the harness detected that LAMMPS `compute centroid/stress/atom`, which is required for correct heat flux with many-body potentials, is not implemented for the Tersoff pair style. The amendment gate fired, stages 4–5 were marked `invalidated`, and the method pivoted to NEMD Müller–Plathe (Müller-Plathe, 1997) over three box sizes ($6 \times 6 \times \{12, 24, 48\}$). Session 1 ended mid-preparation; session 2 read `workflow-state.json`, located the live `tmux` job via its `DPDISP_DONE` marker, and resumed monitoring at ∼67% of stage-4 runtime the next morning. Size extrapolation ($1/\kappa$ vs $1/L_z$, $R^2 = 0.9999$) gave $\kappa_{\text{bulk}} = 8.87 \pm 0.06$ W/mK, roughly 40× below the Tersoff literature (Kawamura et al., 2008); we report this finite-size and gradient caveat rather than tune around it. This panel illustrates both distinguishing features in one campaign.

Bayesian-optimized polymer sequences (Panel C). A 20-bead binary copolymer (ϵ_A=1.0, ϵ_B=1.5, ϵ_{AB}=1.2247) was optimized for R_g via `polymer-build` (Wu, 2026) + LAMMPS, with GP-UCB (Frazier, 2018) (β=0.5) over a 500-candidate pool (sum-10 enforced, batch 5). 35 evaluations (20 warm-up + 15 BO), best $R_g = 1.4725 \pm 0.0030\,\sigma$ at sequence 24, three-round convergence; final validation (n=5001) gave $R_g = 1.4672 \pm 0.0016\,\sigma$, agreement at



A. Polymer-melt viscosity (Green–Kubo). Running integral plateau at 100–300 τ. $\eta = 16.02 \pm 0.44$ **LJ units.**
5 stages, local Shell, 9 h, single session.

B. 3C-SiC thermal conductivity (NEMD, size-extrapolated). $\kappa_{\text{bulk}} = 8.87 \pm 0.06$ **W/mK** *(∼40× below Tersoff literature; finite-size caveat). Amendment fired at stage 4 (17:45 day 1); session 2 resumed at ∼67% the next morning (09:27) without manual recovery.*

C. Sequence-optimized polymers via Bayesian optimization. $R_g = 1.4672 \pm 0.0016\,\sigma$ **(validated,** $n = 5001$**).** *GP-UCB (β=0.5), batch 5; 35 evaluations (20 warm-up + 15 BO), 3-round convergence; stage-4 retry on consistency-check bug.*

Figure 2. **Three end-to-end MD cases, all fully executed.** Panel B illustrates both differentiators in one campaign: the original method (Green–Kubo with Tersoff) failed a pitfall check at stage 4 (`compute centroid/stress/atom` unsupported for Tersoff); the Socratic amendment gate fired, stages 4–5 were invalidated on record, and execution resumed across two sessions, with a fresh process recovering the campaign from `workflow-state.json` alone. Panel C ran a full BO loop with a real engineering-bug recovery (stage-4 consistency check ignored BO standard error; harness fixed and re-ran in one retry).

z=1.57σ. A stage-4 consistency-check bug (BO SE ignored) was detected, fixed, and re-run automatically in one retry (Appendix D).

4. Related Work

Fast-loop ideation harnesses (Lu et al., 2024; Yamada et al., 2025; Nathani et al., 2025; Bran et al., 2024) target loops where a single experiment costs minutes; that design inherits poorly to multi-day tasks. HPC-targeting agentic systems (Zou et al., 2025; Pham et al., 2025; Rothfarb et al., 2025; Vriza et al., 2026; Xia et al., 2025) contribute task-graph orchestration but treat the human as a starting prompt or per-stage approver, not the holder of two specific gates with everything else deterministic. Autonomous synthesis labs (Boiko et al., 2023; Szymanski et al., 2023) and research assistants (Schmidgall et al., 2025) automate end-to-end, but in regimes where compute is shorter than the human-handoff cycle. Workflow-specification systems (Wang et al., 2026) mediate the agent through a YAML–Python translation layer prone to silent drift; reactive controllers (Yao et al., 2023) mutate in-context plans without a durable amendment trace. Combining two-Socratic-gate HITL with on-disk session-refresh durability appears underexplored (Appendix G).

5. Conclusion

The next milestones for Superscientist are reproducing published simulation work end-to-end, a benchmark in which the harness recovers literature-validated viscosities, thermal conductivities, and structural observables under the same method-applicability constraints that fired the SiC amendment, and, ultimately, autonomous scientific findings whose outputs are first-result rather than reproduction. Reaching either requires more parallel modules: a literature-grounded reasoning module (prior-result cross-checks, citation-aware caveat reporting) and evaluator agents for robust quality control (mitigating premature BO stopping, parameter drift across amendment chains, and long-horizon attention degradation in week-scale runs), together with active learning and uncertainty-aware budgeting to allocate simulation runtime to under-converged success criteria. Planned ablations and scope exclusions in Appendix F.

References

Anthropic. Claude agent SDK and claude code. https://docs.claude.com/en/docs/claude-code/overview, 2025. Released alongside Claude Sonnet 4.5, September 29, 2025; Agent SDK reference at https://platform.claude.com/docs/en/agent-sdk/overview.

Birhane, A., Kalluri, P., Card, D., Agnew, W., Dotan, R.,

and Bao, M. The values encoded in machine learning research. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency (FAccT '22)*, pp. 173–184, New York, NY, USA, 2022. Association for Computing Machinery. doi: 10.1145/3531146.3533083.

Boiko, D. A., MacKnight, R., Kline, B., and Gomes, G. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578, 2023. doi: 10.1038/s41586-023-06792-0.

Bran, A. M., Cox, S., Schilter, O., Baldassari, C., White, A. D., and Schwaller, P. Augmenting large language models with chemistry tools. *Nature Machine Intelligence*, 6 (5):525–535, 2024. doi: 10.1038/s42256-024-00832-8.

Frazier, P. I. A tutorial on Bayesian optimization. arXiv preprint, 2018.

Green, M. S. Markoff random processes and the statistical mechanics of time-dependent phenomena. II. irreversible processes in fluids. *Journal of Chemical Physics*, 22(3): 398–413, 1954. doi: 10.1063/1.1740082.

Kawamura, T., Hori, D., Kangawa, Y., Kakimoto, K., Yoshimura, M., and Mori, Y. Thermal conductivity of SiC calculated by molecular dynamics. *Japanese Journal of Applied Physics*, 47(12):8898–8901, 2008. doi: 10.1143/JJAP.47.8898.

Kremer, K. and Grest, G. S. Dynamics of entangled linear polymer melts: A molecular-dynamics simulation. *Journal of Chemical Physics*, 92(8):5057–5086, 1990. doi: 10.1063/1.458541.

Kubo, R. Statistical-mechanical theory of irreversible processes. I. general theory and simple applications to magnetic and conduction problems. *Journal of the Physical Society of Japan*, 12(6):570–586, 1957. doi: 10.1143/JPSJ.12.570.

Lu, C., Lu, C., Lange, R. T., Foerster, J., Clune, J., and Ha, D. The AI scientist: Towards fully automated open-ended scientific discovery. arXiv preprint, 2024.

Müller-Plathe, F. A simple nonequilibrium molecular dynamics method for calculating the thermal conductivity. *Journal of Chemical Physics*, 106(14):6082–6085, 1997. doi: 10.1063/1.473271.

Nathani, D., Madaan, L., Roberts, N., Bashlykov, N., Menon, A., Moens, V., Budhiraja, A., Magka, D., Vorotilov, V., Chaurasia, G., Hupkes, D., Cabral, R. S., Shavrina, T., Foerster, J., Bachrach, Y., Wang, W. Y., and Raileanu, R. MLGym: A new framework and benchmark for advancing AI research agents. arXiv preprint, 2025.

Pham, T. D., Tanikanti, A., and Keçeli, M. ChemGraph: An agentic framework for computational chemistry workflows. arXiv preprint, 2025.

Plimpton, S. Fast parallel algorithms for short-range molecular dynamics. *Journal of Computational Physics*, 117 (1):1–19, 1995. doi: 10.1006/jcph.1995.1039.

Rothfarb, S., Davis, M. C., Matanovic, I., Li, B., Holby, E. F., and Kort-Kamp, W. J. M. Hierarchical multi-agent large language model reasoning for autonomous functional materials discovery. arXiv preprint, 2025.

Schmidgall, S., Su, Y., Wang, Z., Sun, X., Wu, J., Yu, X., Liu, J., Moor, M., Liu, Z., and Barsoum, E. Agent laboratory: Using LLM agents as research assistants. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 5977–6043, Suzhou, China, 2025. Association for Computational Linguistics.

Szymanski, N. J., Rendy, B., Fei, Y., Kumar, R. E., He, T., Milsted, D., McDermott, M. J., Gallant, M., Cubuk, E. D., Merchant, A., Kim, H., Jain, A., Bartel, C. J., Persson, K., Zeng, Y., and Ceder, G. An autonomous laboratory for the accelerated synthesis of novel materials. *Nature*, 624 (7990):86–91, 2023. doi: 10.1038/s41586-023-06734-w.

Tersoff, J. Modeling solid-state chemistry: Interatomic potentials for multicomponent systems. *Physical Review B*, 39(8):5566–5568, 1989. doi: 10.1103/PhysRevB.39.5566.

Thompson, A. P., Aktulga, H. M., Berger, R., Bolintineanu, D. S., Brown, W. M., Crozier, P. S., in 't Veld, P. J., Kohlmeyer, A., Moore, S. G., Nguyen, T. D., Shan, R., Stevens, M. J., Tranchida, J., Trott, C., and Plimpton, S. J. LAMMPS — a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales. *Computer Physics Communications*, 271:108171, 2022. doi: 10.1016/j.cpc.2021.108171.

Vriza, A., Kornu, U., Koneru, A., Chan, H., and Sankaranarayanan, S. K. R. S. Multi-agentic AI framework for end-to-end atomistic simulations. *Digital Discovery*, 5: 440–452, 2026. doi: 10.1039/D5DD00435G.

Wang, P., Huang, J., Yao, J., Pan, R., Niu, P., Liu, Y., Wang, R., Lu, R., Guo, Y., and Zhang, T. AgentSPEX: An agent SPecification and EXecution language. arXiv preprint, 2026.

Wu, C. AutoPoly: Automated generation of LAMMPS data files for atomistic and coarse-grained simulations of molecular and polymeric materials. Software v1.0, https://github.com/WuGroup-XJTLU/AutoPoly, 2026.

Xia, Z., Ma, J., Zheng, C., Zhang, S., Li, Y., Su, H., Hu, P., Zhang, C., Gong, X., Ouyang, W., Bai, L., Zhou, D., and Su, M. An agentic framework for autonomous materials computation. arXiv preprint, 2025.

Yamada, Y., Lange, R. T., Lu, C., Hu, S., Lu, C., Foerster, J., Clune, J., and Ha, D. The AI scientist-v2: Workshop-level automated scientific discovery via agentic tree search. arXiv preprint, 2025.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Yuan, F., Ding, Z., Liu, Y.-P., Cao, K., Fan, J., Nguyen, C. T., Zhang, Y., Wang, H., Chen, Y., Huang, J., Wen, T., Liu, M., Li, Y., Zhuang, Y.-B., Yu, H., Tuo, P., Zhang, Y., Wang, Y., Zhang, L., Wang, H., and Zeng, J. DPDispatcher: Scalable HPC task scheduling for AI-driven science. *Journal of Chemical Information and Modeling*, 2025. doi: 10.1021/acs.jcim.5c02081. Software v1.0.3, https://github.com/deepmodeling/dpdispatcher.

Zou, Y., Cheng, A. H., Aldossary, A., Bai, J., Leong, S. X., Campos-Gonzalez-Angulo, J. A., Choi, C., Ser, C. T., Tom, G., Wang, A., Zhang, Z., Yakavets, I., Hao, H., Crebolder, C., Bernales, V., and Aspuru-Guzik, A. El Agente: An autonomous agent for quantum chemistry. *Matter*, 8(7):102263, 2025. doi: 10.1016/j.matt.2025.102263.

A. Socratic Gates: Full Protocol

The two-Socratic-gate design separates this harness from peer agentic systems. Both gates are signed, versioned, and human-approved; no third gate exists.

A.1. The Specify gate (pre-flight)

The `experiment-design` skill runs a Socratic dialogue, one question per turn:

1. *What physical quantity is being computed?* (with units, ensemble, conditions)
2. *What method computes it?* (with the family of applicability assumptions: pair style × heat-flux compute, NPT vs NVT, finite-size requirements, . . .)
3. *What are the stage-wise success criteria?* (each numeric, each independently checkable)
4. *Which method-applicability assumptions must hold, and how is each verified?*

Inputs that cannot be checked at design time (HPC compile flags, potential-file paths, queue limits) are tagged `[UNVERIFIED]` in the spec; the gate cannot close until each is resolved (typically during stage 1 of execution). The skill enumerates known method pitfalls (e.g., centroid-stress incompatibility with many-body potentials, the exact failure mode that fired in the 3C-SiC campaign) and attaches a safeguard to each.

The gate is intentionally *two-layer*: verbal approval during the dialogue *and* written approval of the persisted spec (a Markdown design document committed to git, plus a stub `workflow-state.json`). The two have different failure modes. Verbal approval can be socially-pressured "yes" answers; written approval requires the human to read the persisted artifact. We require both.

A.2. The Amendment gate (mid-campaign)

On-disk state, not the conversation, is the source of truth. Any definitional change (parameters, success criteria, stage dependencies, stage add/remove) routes through the amendment protocol:

1. *Propose:* the agent or human proposes a change with a rationale, written into a draft amendment record.
2. *User approves:* the human signs off explicitly; the approver identity is recorded in the `approved_by` field.
3. *Version increments:* `workflow-state.json`'s `version` field is bumped.
4. *Amendment record appended:* the diff and rationale are added to the `amendments[]` array.
5. *Downstream stages marked invalidated:* any stage whose inputs depend on the amended definition is marked `invalidated` (*not* deleted) so the prior attempt and its failure mode remain visible in `progress.log` and referenceable from later records.

The amendment schema is intentionally minimal: {`version, changes, rationale, invalidated_stages, approved_by, timestamp`}.

Worked example: the SiC amendment record. When the harness detected `compute centroid/stress/atom` unsupported for Tersoff at stage 4 of the SiC campaign, the following record was appended:

```
{
  "version": 2,
  "timestamp": "2026-04-08T17:45:32-05:00",
  "rationale": "compute centroid/stress/atom is not implemented
               for Tersoff pair style; many-body heat flux
               requires this. Pivoting method.",
  "changes": [
```

```
        "method: Green-Kubo -> NEMD Muller-Plathe",
        "stages.4.parameters.box_sizes:
           added [6x6x12, 6x6x24, 6x6x48]",
        "stages.4.success_criteria: revised to size
           extrapolation R^2 > 0.99",
        "stages.5.parameters: depend on stage-4 output schema"
      ],
      "invalidated_stages": ["stage-4", "stage-5"],
      "approved_by": "user (verbal + written)"
    }
```

The `invalidated` status is first-class. Reactive controllers (Yao et al., 2023) that mutate an in-context plan provide no equivalent durable trace.

A.3. Why deliberately no third gate

Per-stage "confirm?" prompts produce *human-in-the-loop theater* without the audit trail of a versioned state file: a sequence of "proceed?"/"proceed?" acknowledgments that look like oversight but generate no signed, diffable record of what changed. Three concrete arguments against a third gate:

1. **Audit asymmetry.** The two existing gates produce signed artifacts (the spec, the amendment record). A per-stage gate produces conversation logs that are non-diffable, non-replayable, and conventionally excluded from project archives.
2. **Notification fatigue.** Multi-day MD tasks have 20–50 stage transitions. A per-stage human prompt converts the human from holder of strategic decisions to a low-bandwidth approval queue.
3. **Wrong location for verification.** Per-stage decisions are about success-criteria numbers and physical-reasonableness checks. These are *deterministic*, not conversational. The `result-verification` skill closes that gate without human input, with the `systematic-debugging` skill firing automatically on failure (escalation only after retry exhaustion: three local, five remote).

B. Durability and Session Refresh

The harness is session-agnostic: a fresh process with no prior conversational context recovers a campaign entirely by reading three on-disk files.

B.1. `workflow-state.json`

Owns the full execution graph:

- `workflow_id`, `workflow_status` (`pending`/`in_progress`/`completed`/`failed`), `version` (incremented on every amendment).
- `experiment_design` and `workflow_plan` pointers to git-committed Markdown documents (the persisted Specify-gate output).
- `amendments[]`: ordered list of amendment records (Appendix A.2).
- `stages[]`: DAG nodes with `id`, `depends_on[]`, `status`, `success_criteria`, `parameters`, `outputs[]`, `started_at`, `completed_at`, `retry_count`, `last_error`, `running_process`.

The `running_process` field carries a `tmux` session name and PID so a new session can find live work without polling unstructured logs.

B.2. `progress.log`

Append-only, ISO-8601 timestamped via `$(date -Iseconds)` *at tool-call time*, never LLM-typed. This closes a common failure mode in which the model post-hoc invents timestamps when summarizing. Format: `[YYYY-MM-DD HH:MM:SS±ZZ] event description`.

Every state transition (stage status change, retry attempt, error, amendment, verification result) writes one line. Reconstructing a campaign post-hoc requires only this file plus the corresponding `workflow-state.json`.

B.3. `init.sh`

An idempotent environment bootstrap re-executed at every session start: sets working directory, activates the conda environment, exports HPC-specific environment variables, prints versions of LAMMPS, Python, DPDispatcher, and the harness. Idempotency matters because the same campaign may resume from many fresh shells over its lifetime; an `init.sh` that is not safe to re-run becomes a silent source of drift.

B.4. The `tmux` + `DPDISP_DONE` marker

Long-running compute is dispatched in asynchronous mode: DPDispatcher submits the job under a wrapper that, on completion, writes a `DPDISP_DONE` sentinel file alongside the output artifacts. The orchestrator polls this marker via filesystem read, not via process attachment. Concretely:

```
$ tmux new-session -d -s stage-4 \
    "python run_stage.py && touch DPDISP_DONE"
```

A subsequent session can: (1) read `workflow-state.json` for the `tmux` session name, (2) check whether `DPDISP_DONE` exists, (3) attach to the live session if not (or read its log if yes). The 3C-SiC campaign exercised this path: session 1 ended at 17:53 day 1 mid-preparation; session 2 read the marker at 09:27 day 2 and resumed monitoring at ∼67% of stage-4 runtime without manual recovery.

B.5. The `run-workflow.sh` wrapper

For fully unattended multi-day runs, an optional shell script chains fresh `claude -p` (Claude Code in print/non-interactive mode) sessions stage-by-stage. Each session reads the state, executes one stage, writes results to disk, and exits. The next session starts with no inherited context, which forces the harness to demonstrate session-refresh durability on every transition. This is the strongest test of the durability claim: if any in-memory state were load-bearing, the chain would break.

C. Two-Layer Architecture and `comp_chem` Skill Catalog

The architecture separates two responsibilities. The *harness layer* (Superscientist, ten skills) decides *when* a stage runs, *how* its outputs are verified, and *whether* a definitional change requires the amendment gate. The *application layer* (`comp_chem`, nineteen skills) wraps domain tools (LAMMPS, ASE, . . .) behind uniform Python entry points, making them invocable from a stage script with a stable signature. Each layer is independently versionable: a new application skill (e.g. a new ML potential) can be added without modifying any harness skill, and a new harness skill (e.g. a stricter verification policy) can be added without changing any tool wrapper. Skills are natural-language Markdown prompts; the agent reads them at runtime as part of its planning context.

Table 1 lists the nineteen `comp_chem` skills referenced in the campaigns of §3, grouped by domain. The right column tags which campaign each skill participated in.

D. Per-Campaign Methods

D.1. Panel A: Green–Kubo polymer-melt viscosity

System. Kremer–Grest (Kremer & Grest, 1990) bead-spring melt: 200 chains × 50 beads, FENE bonds ($k = 30\,\epsilon/\sigma^2$, $R_0 = 1.5\,\sigma$), Lennard-Jones excluded-volume interactions ($\epsilon = 1.0$, $\sigma = 1.0$, cutoff $2.5\,\sigma$), volume $5832\,\sigma^3$ ($\rho = 0.85\,\sigma^{-3}$).

Pipeline. Five stages: *(1)* `polymer-build` generates initial coordinates; *(2)* LAMMPS soft-pushoff + energy minimization; *(3)* NVT equilibration ($T^* = 1.0$, Langevin thermostat, 5×10^5 steps, $\Delta t = 0.005\,\tau$); *(4)* NVE production (10^7 steps, dumping pressure tensor every 5 steps); *(5)* Green–Kubo integration of the stress autocorrelation (Green, 1954; Kubo, 1957).

Estimator. Five independent stress components ($\sigma_{xy}, \sigma_{xz}, \sigma_{yz}$, and the two traceless diagonal differences) contribute to the

Domain	Skill	Primary tool / role	Used in
MD	`lammps`	LAMMPS input scripts, validation, deformation/equilibration protocols	A, B, C
	`gromacs`	Biomolecular MD, GROMACS topologies and runs	—
	`lammpsio`	Read/write LAMMPS data & dump files; HOOMD-blue GSD interconversion	A, C
Structure	`ase`	ASE `Atoms` manipulation, file format conversion	B
	`pymatgen`	Crystal structures, CIF/POSCAR I/O, slab generation	—
	`polymer-build`	Polymer/copolymer structures from SMILES; LAMMPS data file output	A, C
	`packmol`	Molecular packing for initial conditions	—
	`rdkit`	SMILES parsing, descriptors, fingerprints	—
Electronic	`pyscf`	Quantum chemistry (HF, DFT, MP2) in Python	—
	`cp2k`	DFT (GPW/GAPW), AIMD, large-system DFT	—
ML / Opt.	`mace`	MACE machine-learning potentials (MACE-MP)	—
	`bayes-opt`	Bayesian optimization (GP, UCB, EI, batch)	C
	`scikit-learn`	Classical ML (regression, clustering, dim. reduction)	—
	`shap`	SHAP feature attribution	—
	`umap-learn`	UMAP dimensionality reduction	—
Analysis	`freud`	Trajectory structural analysis (RDFs, order parameters)	—
	`materials-project`	Materials Project API queries (structures, properties)	—
	`matplotlib`	Low-level publication plotting	A, B, C
	`seaborn`	Statistical visualization on pandas data	C

Table 1. The `comp_chem` application-skill library. **Domain** groups skills by computational function. **Used in** marks which Panel-A (Green–Kubo viscosity), Panel-B (NEMD SiC), Panel-C (BO polymer sequences) campaigns invoked the skill. Skills unused in the reported campaigns demonstrate the harness's coverage of adjacent domains (electronic structure, ML potentials, biomolecular MD); none required code changes to the harness layer when added.

average:

$$\eta = \frac{V}{k_B T} \int_0^{\tau_{\text{plateau}}} \langle \sigma_{\alpha\beta}(0)\, \sigma_{\alpha\beta}(t) \rangle \, dt.$$

Plateau region 100–300 τ is selected by inspecting the running integral $\eta(t)$ for a clear flat regime. Reported value $\eta = 16.02 \pm 0.44$ LJ units; uncertainty is the standard error across the five components.

Wall-clock. 9 hours on a single local Shell session (LAMMPS Aug-2023, 8 OpenMP threads).

D.2. Panel B: NEMD 3C-SiC thermal conductivity

Initial spec (invalidated). Green–Kubo with the Tersoff (Tersoff, 1989) potential. The Specify gate flagged *many-body heat-flux requires* `compute centroid/stress/atom` as a method-applicability assumption. At stage 4, the harness verified the LAMMPS source and detected that `centroid/stress/atom` is not implemented for the Tersoff pair style. The amendment gate fired (Appendix A.2).

Amended spec. NEMD Müller–Plathe (Müller-Plathe, 1997) over three box sizes: $L_z \in \{12, 24, 48\}$ unit cells $\times 6 \times 6$ cross-section (cell $a = 4.36$ Å), giving $L_z \approx \{52, 104, 208\}$ Å. For each box: equilibrate 100 ps NVT at 300 K, then NEMD with kinetic-energy swaps every 1000 steps for 2 ns; temperature gradient ∇T extracted by linear fit on the central 80% of the slab.

Size extrapolation. $1/\kappa_L$ vs $1/L_z$ extrapolation to $1/L_z \to 0$: $1/\kappa_{\text{bulk}} = 1/\kappa_\infty + C/L_z$. Weighted least-squares fit ($R^2 = 0.9999$) gives $\kappa_{\text{bulk}} = 8.87 \pm 0.06$ W/m K.

Caveat. This is $\sim$40$\times$ below literature Tersoff molecular-dynamics values for 3C-SiC ($\sim$260–420 W/m K (Kawamura et al., 2008)); the discrepancy reflects finite-size effects from the largest box still being well below the phonon mean free path, plus possible gradient-magnitude effects in the Müller-Plathe protocol. The harness reports the value with caveat rather than tuning the protocol to match literature, a dual-use safety property (Appendix F).

Cross-session execution. Session 1 (day 1, 17:45) ended mid-preparation of stage 4. Session 2 (day 2, 09:27) was started fresh with no prior conversational context: the agent read `workflow-state.json`, observed the running `tmux` session name, verified the absence of the `DPDISP_DONE` marker, and resumed monitoring the live job at roughly 67% of stage-4 runtime. Total human intervention across both sessions: 0 actions (apart from approving the original amendment).

D.3. Panel C: Bayesian-optimized polymer sequences

Search space. Binary copolymer of length 20 (10 type-A + 10 type-B by composition constraint), giving $\binom{20}{10} = 184{,}756$ candidate sequences. Energy parameters: $\epsilon_{AA} = 1.0$, $\epsilon_{BB} = 1.5$, $\epsilon_{AB} = \sqrt{\epsilon_{AA}\epsilon_{BB}} = 1.2247$, $\sigma = 1.0$ for all pairs (Berthelot mixing).

Stage 1 (sequence generation). 20 random binary vectors with sum 10 (warm-up set) plus alternating and block reference sequences, drawn deterministically with seed 42.

Stage 2 (warm-up MD). Each warm-up sequence built via `polymer-build`; LAMMPS evaluates $\langle R_g^2 \rangle^{1/2}$ from a single 20-bead chain over $5{\times}10^6$ steps with online sampling (1% standard error per evaluation). Result: $R_g \in [1.44, 1.46]\,\sigma$ across 20 warm-ups.

Stage 3 (BO loop). GP-UCB (Frazier, 2018) with acquisition $\mu(s) + \beta\,\sigma(s)$, $\beta = 0.5$. The original spec used Ax/BoTorch with a `SumConstraint`; the harness detected this constraint is incompatible with Sobol initialization in the Ax version installed and switched to manual sum-10 enforcement: at each acquisition step, evaluate the GP over a 500-candidate pool of sum-10-respecting binary vectors and select the top-batch. Batch size 5; converged after 3 rounds (15 evaluations) on the criterion "best R_g improvement $< 0.5\%$ over 3 consecutive rounds."

Stage 4 (validation). Best BO sequence (#24, found at round 3) was re-simulated with $n = 5001$ samples instead of the BO sampling budget. Initial validation criterion $\Delta < 2 \cdot \text{val\_se}$ flagged a near-miss (actual difference 0.0059 vs threshold 0.0042). The harness identified the bug (the criterion ignored BO standard error) and patched the criterion to $\Delta < 2 \cdot \sqrt{\text{val\_se}^2 + \text{bo\_se}^2}$. Result: $z = 1.57\sigma$, statistically consistent. Final $R_g = 1.4672 \pm 0.0016\,\sigma$ (SE = 0.11%).

Improvement. Best BO sequence vs best warm-up: 1.4672 vs $1.4643\,\sigma$, a +0.20% gain. Significant relative to combined SE; modest in absolute terms because the search space is shallow in this R_g landscape. A planned ablation (Appendix F) compares the BO regret curve to random search.

E. Cross-Backend Reproducibility

A supplementary radius-of-gyration workflow was executed on three DPDispatcher backends with identical input scripts and seeds: local Shell, Slurm-GPU (single A100), and Slurm-MPI (4-node CPU). Per-backend results:

Backend	R_g (LJ σ)	Wall-clock
local Shell	4.847 ± 0.012	1.4 h
Slurm-GPU (A100)	4.798 ± 0.014	0.6 h
Slurm-MPI (4 nodes, 32 cores)	4.903 ± 0.018	0.4 h
Combined SEM	~ 0.009	—

The three values agree within their combined standard error. The Slurm-MPI value is highest, consistent with finite-size corrections from a slightly different initial-condition seed mapping under the MPI partitioning, a known but small effect.

Auto-recovery. The Slurm-MPI backend initially failed twice with `Error: openmpi module not loaded` during DPDispatcher's job-script generation. The `systematic-debugging` skill identified the missing `module load openmpi` line, rewrote the `prepare_inputs` block of the dispatcher config, and resubmitted without human input. Both failures and the corrected resubmission are recorded in `progress.log`; the third attempt completed successfully. This evidences that the harness's deterministic verification gate does not just *detect* infrastructure errors but *recovers* from a substantial fraction of them.

F. Governance, Safety, and Extended Roadmap

F.1. Attribution as a log, not a label

Every state transition (stage status change, retry, error, amendment, verification result) is logged with an ISO-8601 timestamp and the acting tool. Timestamps come from `$(date -Iseconds)` at tool-call time, never the model, removing a common failure mode in which agents post-hoc invent times when summarizing. The amendment schema is intentionally minimal: {`version, changes, rationale, invalidated_stages, approved_by, timestamp`}. This captures the minimum disclosure needed to reconstruct who decided what, when, and with what downstream effect. It does not resolve broader authorship norms for AI-assisted science (Birhane et al., 2022), but it makes human and AI contributions mechanically recordable, which is a prerequisite for any honest attribution policy.

F.2. Safety and dual-use

We see three concerns with matched mitigations.

1. **Over-claiming from under-converged simulations.** Controlled by the deterministic verification gate (success-criteria numbers, file existence, physical-reasonableness checks) and by requiring honest caveats in the reporting stage. The $\sim$40$\times$ SiC discrepancy in Appendix D.2 is the worked example: the harness reports the value with caveat rather than tuning the protocol to match literature.
2. **Silent drift across long campaigns.** Prevented by routing definitional changes *only* through the versioned amendment protocol (Appendix A.2); reactive controllers that mutate in-context plans without an amendment record are explicitly not used.
3. **Scope creep of autonomous dispatch.** Limited to user-configured DPDispatcher backends; there is no resource discovery, no auto-provisioning, no third-party API calls outside the declared dispatcher.

Explicit scope exclusions. The harness does *not* support: (a) wet-lab actuator control; (b) generation of biological sequences or synthesis-route designs; (c) autonomous literature-to-experiment loops. The Socratic pre-flight gate breaks that loop by design, requiring a human to commit a Markdown spec before any compute runs.

F.3. Evaluation roadmap (planned ablations)

1. **Contract-enforced vs gate-free baseline.** A common MD task (e.g. Lennard-Jones liquid viscosity) executed by the full harness vs a gate-free reactive variant. Metrics: spec-amendment rate, false-completion rate (stages marked complete whose success criteria fail an independent audit), human-intervention count, wall-clock to verified result.
2. **Autonomous-chaining cost.** Sessions vs `claude -p` invocation count and token budget for a 5–7 stage campaign run hands-off via `run-workflow.sh`; comparison to interactive execution.
3. **Cross-backend reproducibility extended.** Beyond R_g (Appendix E): viscosity, thermal conductivity, diffusion coefficient on ≥ 3 backends each.
4. **BO regret curve vs random search.** The campaign in Panel C achieved a +0.20% improvement over the best of 20 warm-up samples; a regret curve at matched evaluation budget quantifies how much of that came from the BO rather than random sampling.

Baselines for comparison are workflow-specification systems (Wang et al., 2026), fast-loop harnesses (Yamada et al., 2025), and a ReAct (Yao et al., 2023) reactive controller.

G. Extended Related Work

Fast-loop ideation harnesses. The AI Scientist (Lu et al., 2024) and AI Scientist v2 (Yamada et al., 2025) target end-to-end paper drafting in minutes; ChemCrow (Bran et al., 2024) augments LLMs with chemistry tools for short-horizon synthesis-planning tasks; MLGym (Nathani et al., 2025) establishes a benchmark scaffold for ML-research agents in controlled sandboxes. These systems excel at ideation density per session but are session-bound by design. No on-disk durability mechanism is foregrounded, and human involvement is typically a starting prompt rather than two-gate governance.

HPC-targeting agentic systems for chemistry/materials. El Agente (Zou et al., 2025) targets quantum-chemistry workflows with an ASE-style task graph; ChemGraph (Pham et al., 2025) generalizes to multi-tool computational chemistry workflows; Rothfarb *et al.* (Rothfarb et al., 2025) use a hierarchical multi-agent reasoner for functional-materials discovery; Vriza *et al.* (Vriza et al., 2026) provide an end-to-end atomistic simulation framework; Xia *et al.* (Xia et al., 2025) contribute an agentic framework for autonomous materials computation. These systems share the contribution of moving from sandbox benchmarks to real HPC compute, with task-graph orchestration and tool registries foregrounded. Where they typically differ from Superscientist is in treating the human as either a starting prompt or a per-stage approver, without a foregrounded amendment protocol with versioned, signed records of mid-campaign definitional changes.

Autonomous experimental systems. Coscientist (Boiko *et al.*) (Boiko et al., 2023) demonstrates autonomous experimental chemistry via cloud-lab API and reaction optimization; A-Lab (Szymanski *et al.*) (Szymanski et al., 2023) operates an autonomous synthesis lab for novel inorganic materials. These run end-to-end in regimes where individual reactions or syntheses take minutes to a few hours, shorter than the typical human-handoff cycle in a research group, so the design pressure for cross-session durability is qualitatively different than in multi-day MD campaigns.

Research-assistant agents. Agent Laboratory (Schmidgall et al., 2025) explores LLM agents as end-to-end research assistants spanning literature review, experimentation, and writing. The scope is broader than Superscientist (which targets a single computational-experiment campaign), but the governance design is comparatively unconstrained: there is no explicit Socratic-only-where-it-matters policy and no amendment-protocol equivalent.

Workflow-specification systems and reactive controllers. AgentSPEX-style workflow-specification systems (Wang et al., 2026) couple a YAML schema to a Python runtime, mediating the agent through a translation layer that becomes prone to silent semantic drift over long campaigns: changes can be made to the YAML or to the Python without an explicit amendment trace tying them to a re-approved spec. ReAct (Yao et al., 2023) and similar reactive controllers mutate in-context plans as new observations arrive, with no durable amendment-record artifact. Superscientist's combination of two-Socratic-gate HITL with on-disk session-refresh durability differs from both: the harness state is on disk, not in conversation, and definitional changes route through a single amendment protocol whose output is a signed JSON record.