COMPMAT-BENCH: BENCHMARKING AI AGENTS FOR COMPUTATIONAL MATERIALS SCIENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Evaluating AI agents on scientific research tasks is constrained by the time and resources required for the underlying experiments or calculations. In computational materials research, repeating the same expensive simulations across agents and trials can make evaluation impractical. We introduce CompMat-Bench, a benchmark of 94 tasks derived from recently published computational materials studies, each asking agents to complete a step toward achieving the study's scientific goal. We reproduce the research steps in advance and assess agents on preparing inputs and analyzing outputs for expensive simulations, so expensive simulations can be avoided during evaluation. The reproduced inputs and results serve as ground truth for grading agents with fixed rules, without an LLM judge. The benchmark supports four evaluation conditions: single tasks and workflows composed of related tasks, each with full or reduced methodological guidance. With full guidance on single tasks, agents based on three LLMs demonstrate the ability to complete individual materials research steps, with pass rates of 66.0–90.4% across 94 tasks. Both longer workflows and reduced guidance can limit agent performance, but in different ways for different agents: they lower the pass rates of the weaker agents, whereas the strongest agent falls only when a long workflow is combined with reduced guidance. Failure analysis attributes most failures to scientific errors rather than to errors in software usage. CompMat-Bench provides a basis for comparing agents on the steps of real materials research and for analyzing agent failure modes.

1 INTRODUCTION

Large language models (LLMs) have demonstrated strong coding capabilities, including solving programming problems and fixing bugs in software repositories (Chen et al., 2021; Jimenez et al., 2024; Yang et al., 2024; Tian et al., 2024). They have also demonstrated strong scientific understanding (Rein et al., 2024; Hendrycks et al., 2021). Computational materials research brings these capabilities together: scientific understanding guides the choice of models and methods, while code implements calculations and analyzes results. Yet separate assessments of coding and scientific understanding do not establish how well agents can combine them in materials research. We therefore need benchmarks that directly quantify LLM agents' capabilities in computational materials research and help assess their potential to accelerate scientific research (Lu et al., 2024; Boiko et al., 2023; Ghafarollahi & Buehler, 2025).

However, evaluating agents on computational materials research is challenging because simulations can take hours, or even days. Repeating these simulations across agents and trials is costly and makes rapid evaluation impractical (Cao et al., 2026). Most existing benchmarks either exclude computationally demanding tasks, as in ScienceAgentBench (Chen et al., 2025), or provide substantial computing resources (e.g., up to 72 hours per run in AutoMat (Huang et al., 2026a)). The former narrows the range of research represented, while the latter retains the cost of running simulations for each attempt. Our aim is to evaluate LLM agents on **diverse research problems** in computational materials research, **without expensive simulations** during evaluation.

We introduce **CompMat-Bench**, a benchmark of 94 tasks derived from 15 recently published, peer-reviewed studies in computational materials research. We remove expensive simulations from task evaluation by (1) identifying the research steps that require them and reproducing these simulations

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.

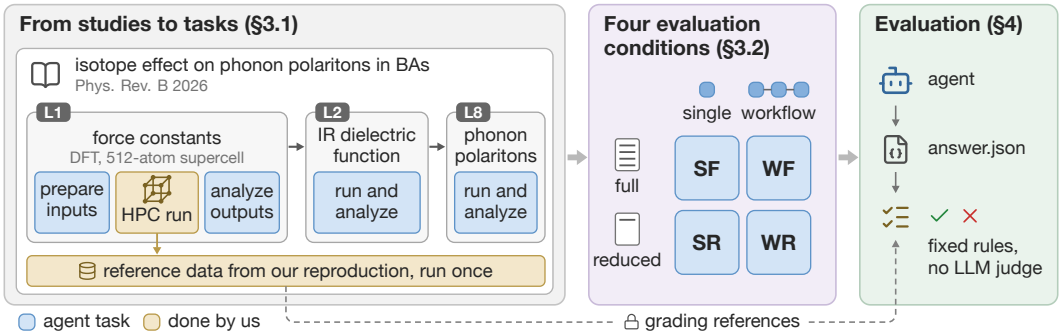


Figure 1: Overview of CompMat-Bench, illustrated with research steps from a study of isotope effects in boron arsenide (Wu, 2026). For an expensive step, input preparation and output analysis are separate tasks; prepared inputs are graded without running the simulation, and analysis starts from our precomputed outputs. Lighter calculations are executed by the agent. Tasks are evaluated singly or as a workflow of connected steps, with full or reduced method guidance. Fixed rules grade submissions against hidden references from our reproduction.

in advance, (2) saving their inputs and results as grading references, and (3) assessing agents on preparing simulation inputs and analyzing the resulting data (Fig. 1). Most computational materials tools are standard software packages with fixed interfaces. These interfaces, together with our reproduced inputs and results, allow us to grade agents using fixed rules, including numerical tolerances for scientific outputs, without an LLM judge.

CompMat-Bench also provides a protocol for controlling task difficulty through methodological guidance and task span. For each task, we provide full- and reduced-guidance prompts that preserve the scientific target and grading criteria. We combine scientifically dependent tasks from the same paper into eight workflows, each requiring an agent to reach the final result from the initial inputs. Crossing guidance with task span yields four evaluation conditions: single tasks with full (SF) or reduced (SR) guidance, and workflows with full (WF) or reduced (WR) guidance.

We evaluate four agents built on three LLMs under the four conditions and analyze their failures. Our contributions are:

(1) **Computational materials benchmark.** We introduce CompMat-Bench, with 94 tasks for fixed-rule evaluation of computational materials research without expensive simulations.

(2) **Difficulty control.** We vary task difficulty along two axes, methodological guidance and task span, while keeping the scientific target and grading of each task fixed, which quantifies their individual contributions to agent performance.

(3) **Failure analysis.** We identify common failure modes of agents in computational materials research: agents do not recognize anomalies in their own results that an experienced researcher would notice, and agents based on different LLMs lack the same implicit knowledge of software interfaces.

2 Related Work

Evaluation cost. Computational materials research relies on simulations, such as density functional theory (DFT) and molecular dynamics (MD), that can run for hours or even days on high-performance computing (HPC) clusters (Camps et al., 2025). A benchmark that includes these simulations must repeat them for every agent, trial and evaluation condition. Simulation time is an evaluation cost that token counts (Kapoor et al., 2025) do not capture (Cao et al., 2026). Most existing benchmarks either avoid this cost or pay it in every attempt. The first approach, *exclusion*, keeps expensive computation out of the tasks (Bogin et al., 2024). FIRE-Bench selects low-cost machine-learning studies (most attempts finish within one hour) (Wang et al., 2026b). CORE-Bench selects papers whose code runs in under 45 minutes (Siegel et al., 2024). ReplicationBench caps the core contributions of each paper at six hours (with intermediate data products for costlier steps) (Ye et al.,

Table 1: CompMat-Bench and prior agent benchmarks on three properties: tasks derived from published studies, tasks involving expensive computation, and expensive computation excluded from evaluation; the last column gives the grading method. ✓ yes, ✗ no, (✓) in part, – no expensive computation in the tasks.

Benchmark	Domain	Published studies	Expensive computation	Excluded from evaluation	Grading
FIRE-Bench (Wang et al., 2026b)	ML	✓	✗	–	LLM judge
ReplicationBench (Ye et al., 2025)	astrophysics	✓	(✓)	(✓)	fixed rules
PaperBench (Starace et al., 2025)	ML	✓	(✓)	✗	LLM judge
AutoMat (Huang et al., 2026a)	materials	✓	✓	(✓)	LLM judge
SimulCost (Cao et al., 2026)	physics sim.	✗	(✓)	✗	fixed rules
CompMat-Bench	materials	✓	✓	✓	fixed rules

2025). Exclusion keeps attempts cheap but limits the research to light computation. The second approach, *rerunning*, executes the computation in every attempt (Chan et al., 2025; Faroughy et al., 2026). AutoMat allows each run up to 96 CPU cores and 72 hours (Huang et al., 2026a). PaperBench executes each submission on a fresh GPU machine (Starace et al., 2025). Rerunning retains the research but repeats its cost for every agent and trial. Recent benchmarks take a third approach, *precomputation*, and supply outputs computed in advance. gwBenchmarks builds its tasks on public gravitational-wave simulations and catalogs (Islam et al., 2026). MDArena evaluates the stages before and after MD runs, which lack a ground truth and are stochastic (Anand et al., 2026).

CompMat-Bench runs each expensive simulation once in advance and evaluates agents on preparing its inputs and analyzing its outputs. This suits computational materials research because most simulations run in standard software packages with fixed interfaces. Input files that agents prepare can therefore be graded by fixed rules without running them, and the outputs of one reproduction serve every agent and trial. Table 1 compares benchmarks on three properties: tasks derived from published studies, tasks involving expensive computation, and expensive computation excluded from evaluation. Each prior benchmark lacks at least one property or holds it only in part, whereas CompMat-Bench holds all three.

Real research tasks. A benchmark based on real research tasks better reflects the research capability of agents. However, collecting and converting real research tasks is difficult, and some benchmarks define their own tasks (Zhu et al., 2025). SimulCost asks agents to tune the parameters of standard physics solvers (Cao et al., 2026). gwBenchmarks defines eight gravitational-wave research tasks, from regression to physics derivation (Islam et al., 2026). A better approach is to select published, peer-reviewed research and convert it into benchmark tasks (Qiu et al., 2026; Majumder et al., 2025; Deng et al., 2026). ScienceAgentBench adapts 102 tasks from 44 peer-reviewed publications (Chen et al., 2025). CORE-Bench builds tasks from the released code of 90 papers (Siegel et al., 2024). PaperBench asks agents to replicate 20 ICML 2024 papers (Starace et al., 2025). ReplicationBench selects 20 peer-reviewed astrophysics papers with their authors' collaboration (Ye et al., 2025). FIRE-Bench further decomposes each paper into a tree of research problems (Wang et al., 2026b). CompMat-Bench converts peer-reviewed materials studies, mostly published between January and March 2026, into research steps that keep their scientific dependencies. It reproduces the expensive simulations of these studies and uses the results as ground truth, so that the ground truth is consistent with the task settings and includes the details that papers omit.

Materials science benchmarks. Although the number of materials science benchmarks keeps growing (Zaki et al., 2024; Niyongabo Rubungo et al., 2025; Yang et al., 2026), most of them focus on a single type of computational approach or tool. MatTools tests knowledge and use of the pymatgen library (Liu et al., 2025). INCARBench tests the preparation of VASP INCAR files against Materials Project records (Shao et al., 2026). QMBench tests DFT knowledge, structure-file generation and parameter choice without running calculations (free text graded by an LLM) (Wang et al., 2025). DFTBench runs Quantum ESPRESSO workflows in every attempt (Hu et al., 2026). MDGym reruns LAMMPS and GROMACS simulations in every attempt (for both materials and biomolecular systems) (Kumar et al., 2026). AutoMat, the closest work, asks agents to reproduce 85

claims from 10 materials papers across several methods and scores them with an LLM judge (Huang et al., 2026a). Its claims are unevenly distributed across these papers, with at least 31 from a single paper (counted from the released dataset, Huang et al., 2026b). CompMat-Bench draws its 94 tasks from 15 published studies that use 12 scientific codes, with no study contributing more than 9 tasks, and grades them with fixed rules instead of an LLM judge.

3 Benchmark Construction

Here, we describe how CompMat-Bench is constructed: how tasks are derived from published studies (Section 3.1), how guidance and task span are controlled (Section 3.2), and what the resulting tasks cover (Section 3.3).

3.1 From studies to tasks

Study selection. We draw studies from four journals: npj Computational Materials, Physical Review B, ACS Nano and Nano Letters. We collected the 1,912 articles published from January to March 2026 whose titles and abstracts were available through the OpenAlex API (Priem et al., 2022). We keep studies whose core is a computation on a real material with a checkable result, and exclude experiment-led, dataset and review papers. We then require a part of the study that we can afford to reproduce and calculations that are light on the agent's side but demand scientific reasoning. We also exclude studies made public before August 2025 (e.g., a study published in January 2026 whose preprint appeared in March 2025) (Jain et al., 2025) and studies that rely on closed-source codes. These filters leave 72 studies, from which we select 14 by hand, favoring methods and codes not yet covered. A fifteenth study, published in September 2025, served as the pilot for these criteria.

Decomposition and reproduction. We convert each study into tasks in two steps. First, a language model guided by a structured prompt adapted from FIRE-Bench (Wang et al., 2026b) decomposes the study into a directed acyclic graph of research steps, which we check against the paper. Each step is one calculation that produces a new physical quantity shown in a main-text figure or table, and each edge records an upstream result that the step uses. These graphs have 6 to 18 steps per study. Second, we select a logically connected sub-graph in which every selected step keeps all of its upstream steps. Each of the 15 studies contributes 3 to 7 steps, with a median of 5 (Table A1). Of all 65 steps, 30 need HPC resources for calculations such as DFT, and 35 run within minutes on one machine. We reproduce these sub-graphs on HPC clusters and local workstations, and our results serve as the grading references. They sometimes differ from the published values, mostly because papers do not report some inputs and we have to set them ourselves.

Tasks and grading. We build three kinds of tasks. For a step that needs HPC, the agent either prepares the simulation inputs or analyzes our precomputed outputs (Fig. 1). For a step that does not, the agent runs the calculation and analyzes the result. Fig. 2 shows a concrete example from a study of magnons in CrN, published in Physical Review B (Kim et al., 2026). Its three stages correspond to three tasks: preparing the DFT inputs, deriving exchange constants from our precomputed DFT outputs, and computing the spin-wave dispersion, including the magnon splitting at the zone boundary. The left panel of Fig. 2 shows the prompt of the last task. Its full-guidance prompt states the scientific target, the method and the required output, and the reduced-guidance prompt omits how the magnon energies are obtained, the Holstein-Primakoff expansion and the Bogoliubov diagonalization (shaded). The hidden reference stores our value for each graded quantity with its tolerance, here the zone-boundary magnon splitting (6.1272 meV, within 10%). Input files are graded on the effective value of each required setting, including software defaults, and on structures equivalent to the reference. Analysis results must match our reference within numerical tolerances, 10% relative by default. A task scores 1 only when every check passes. Tasks have 1 to 55 checks, with a median of 12. Prompts are written in our own words and structures are regenerated from public databases, so no text, figures or values are copied from the source papers.

Quality control. Each task is reviewed by a computational materials researcher with a PhD in a related field, who grades it on four questions: (1) answer leakage, whether any file given to the agent reveals the answer, the method or a shortcut; (2) prompt clarity, whether the prompt defines the task

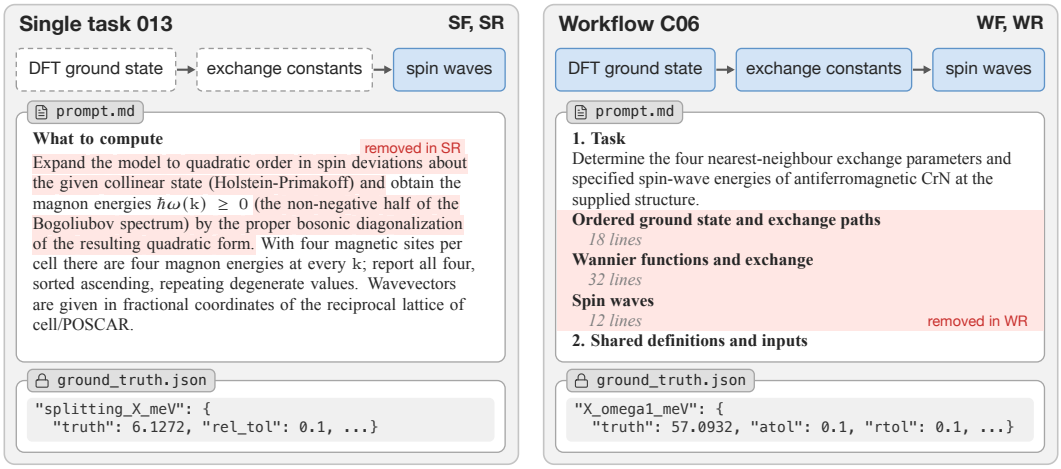


Figure 2: A single task and a workflow from a study of magnons in antiferromagnetic CrN (Kim et al., 2026). Both panels show the same three stages. Left: in single task 013 (SF, SR), the first two stages (dashed) are precomputed, and the agent computes the spin waves from their outputs. Right: in workflow C06 (WF, WR), the agent runs all three stages. Shaded prompt text is removed in the reduced-guidance version, and the rest of the prompt is identical. Bottom: the hidden reference value and tolerance for one graded quantity.

and its valid answers without ambiguity; (3) reference quality, whether the reference answer or input set is physically reasonable; and (4) scoring fairness, whether an essentially correct answer could fail.

3.2 Guidance and task span

Our construction lets us assemble variants of the benchmark. We vary methodological guidance through the level of detail in single-task prompts, and task span by combining single tasks into workflows. Crossing them gives the four conditions SF, SR, WF and WR introduced in Section 1.

Full and reduced guidance. The full-guidance prompt states the scientific target, the inputs, the method and the required output. The reduced-guidance prompt removes method guidance that the grader does not check (Fig. 2, shaded) and keeps everything else, so both prompts share one grader. In 42 tasks the full prompt contains no such guidance, and the two prompts are identical. Of the other 52 tasks, 34 lose explicit method or implementation guidance and 18 lose explanations or hints.

Single tasks and workflows. Because the selected steps keep their scientific dependencies, we can chain single tasks from one study into a workflow that takes the agent from the initial inputs to the final result (Fig. 2, right). We build eight workflows from eight studies (Table C1). Seven have three stages, and one has four. Unlike single tasks, which cover one step each, a workflow requires the agent to run every calculation itself, including simulations in seven of the eight workflows, within a five-hour limit on one node. Each workflow has full- and reduced-guidance prompts, and fixed rules grade its final quantities against our reference.

3.3 Composition

CompMat-Bench covers different journals, computational methods and scientific codes. The 15 source studies appeared in Physical Review B (11), Nano Letters (2) and npj Computational Materials (2). Their 94 tasks span 11 method families and 12 scientific codes (Fig. 3). VASP (Kresse & Furthmüller, 1996) (28 tasks), Quantum ESPRESSO (Giannozzi et al., 2009) (11) and LAMMPS (Thompson et al., 2022) (10) are the most common codes, and in 30 tasks the agent writes the calculation in Python without a named code. The tasks comprise 30 input-preparation, 29 output-analysis and 35 run-and-analyze tasks, and no study contributes more than 9.

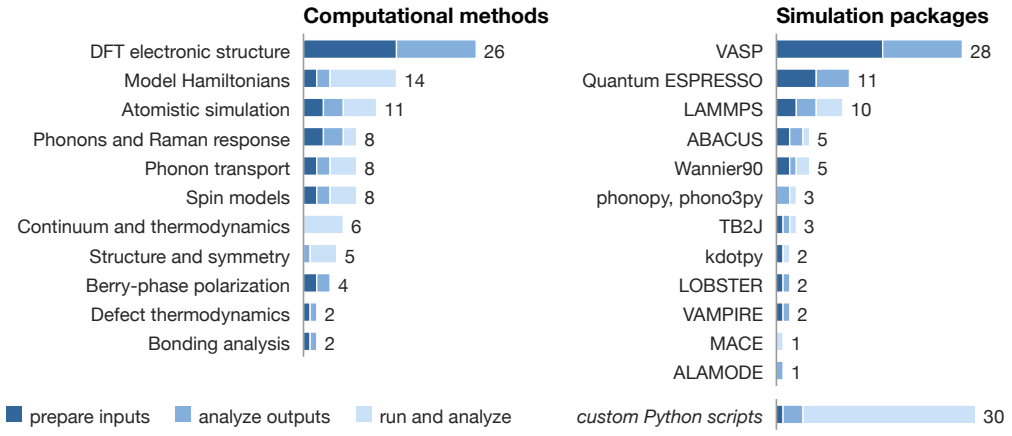


Figure 3: Composition of the 94 tasks. Left: tasks by computational method. Right: tasks by the scientific code they involve. Shading gives the kind of task, as in Fig. 1.

4 Evaluation Setup

Agents. We evaluate four agents built on three LLMs. DeepSeek V4.1 Flash (DeepSeek-AI, 2026), Gemini 3.8 Flash (Google DeepMind, 2026) and GPT-5.6 Sol (OpenAI, 2026) each run in the MatClaw harness (Zhang & Yakobson, 2026), and GPT-5.6 Sol also runs in OpenAI's Codex CLI (OpenAI, 2025). In MatClaw, all three models use high reasoning effort and, on single tasks, at most 500 agent steps (100 for DeepSeek V4.1 Flash), with an 800k-token context for DeepSeek V4.1 Flash and 256k tokens for Gemini 3.8 Flash and GPT-5.6 Sol. Codex CLI also uses high reasoning effort and a 256k-token context, with no step limit (Table B1).

Run conditions. The agent works in an isolated container, with a new session for every run and no internet access except the model API. It gets the task files and offline user manuals for the simulation codes the task uses, and cannot see reference answers or graders. For a single task, it has 4 CPU cores, 16 GiB of memory (6 GiB for 85 of the 94 DeepSeek V4.1 Flash runs under SF) and one hour, and expensive simulations are precomputed. For a workflow, the agent runs all calculations itself on one compute node with two 64-core AMD EPYC 7763 processors and 239 GiB of memory, and must finish within five hours and 5,000 steps.

Evaluations and scoring. We run the following evaluations: (1) SF: each of the four agents runs all 94 tasks once, and each MatClaw agent runs a selected subset of 20 tasks five more times to measure the reliability of its performance; (2) SR: each MatClaw agent runs once the 52 tasks whose reduced prompt removes method guidance or explanations; (3) WF and WR: each MatClaw agent runs each of the eight workflows under both guidance levels in three rounds, 24 runs per agent and condition. We report the pass rate, the fraction of runs whose submission meets every check of the task's grading rules (Section 3.1); for a workflow, the checks apply to its final result.

5 Results and Analysis

5.1 All agents pass most single tasks under full guidance

All four agents pass most single tasks under full guidance, with pass rates ranging from 66.0 to 91.5% (Table 2 and Fig. C1). GPT-5.6 Sol passes 90.4% in MatClaw and 91.5% in Codex CLI, so the two harnesses give nearly the same result for the same model. Input preparation is the hardest kind of task for every agent, with pass rates of 36.7 to 76.7%, whereas the three stronger agents pass 96.6 to 100% of the output-analysis and run-and-analyze tasks. At standard vendor prices, one run over the 94 tasks costs 3.86 USD for DeepSeek V4.1 Flash, 27.00 USD for GPT-5.6 Sol and 106.43 USD for Gemini 3.8 Flash. GPT-5.6 Sol thus matches Gemini 3.8 Flash in pass rate at a quarter of the cost, despite a higher price per token. The high cost of Gemini 3.8 Flash comes from

Table 2: Pass rates of four agents on the 94 single tasks with full guidance, one run per task. API cost is the cost of the 94 runs at each vendor's standard list price; the Codex CLI run was billed by subscription and has no recorded cost.

Model	Harness	Pass rate (%)				API cost (USD)
		Prepare inputs $n = 30$	Analyze outputs $n = 29$	Run and analyze $n = 35$	All tasks $n = 94$	
DeepSeek V4.1 Flash	MatClaw	36.7	79.3	80.0	66.0 (62/94)	3.86
Gemini 3.8 Flash	MatClaw	70.0	96.6	100.0	89.4 (84/94)	106.43
GPT-5.6 Sol	MatClaw	76.7	96.6	97.1	90.4 (85/94)	27.00
GPT-5.6 Sol	Codex CLI	76.7	100.0	97.1	91.5 (86/94)	–

Table 3: Repeatability of three agents on a selected subset of 20 single tasks with full guidance, five runs per task. The pass rate is over the 100 runs, and the remaining columns count the tasks by how many of the five runs passed.

Model	Pass rate (%)	Tasks by passing runs					
		5/5	4/5	3/5	2/5	1/5	0/5
DeepSeek V4.1 Flash	62.0	7	3	2	3	3	2
Gemini 3.8 Flash	88.0	16	1	1	0	1	1
GPT-5.6 Sol	93.0	17	2	0	0	0	1

its long runs: it takes a median of 56 steps per task, against 4 for GPT-5.6 Sol, and uses 48 times as many input tokens.

To test whether one run per task is enough, we also run repeated tests with the three MatClaw agents on a selected subset of tasks (five runs on 20 tasks; Table 3). The ranking of Table 2 is robust to repeated runs: DeepSeek V4.1 Flash, Gemini 3.8 Flash and GPT-5.6 Sol pass 62, 88 and 93 of the 100 runs, in the same order as in their single runs. The failures of Gemini 3.8 Flash and GPT-5.6 Sol are mostly tied to specific tasks rather than to chance between runs, since only 2 to 3 of the 20 tasks give mixed results, passing some but not all five runs, whereas most failures of DeepSeek V4.1 Flash vary between runs, with mixed results on 11 tasks.

5.2 Longer workflows limit the weaker agents, and the stronger agent falls only on workflows with reduced guidance

Fig. 4 compares the pass rates of three agents based on different LLMs under four conditions: SF, SR, WF and WR. In WF and WR, each of the eight workflows is run three times (24 runs per condition). In SF and SR, we cover the 52 of the 94 tasks that have a reduced prompt, so that the two guidance levels are compared on the same tasks. From Fig. 4, we find two conclusions common to all agents. (1) Longer task span and reduced guidance do not raise the pass rate beyond noise, and mostly lower it. Compared with SF, WF lowers the pass rates of DeepSeek V4.1 Flash and Gemini 3.8 Flash by 21 and 11 points, and SR lowers them by 9 and 7 points, whereas GPT-5.6 Sol stays within 4 points of SF under both. (2) WR is the hardest condition, with pass rates of 50, 54 and 75% for DeepSeek V4.1 Flash, Gemini 3.8 Flash and GPT-5.6 Sol (for DeepSeek V4.1 Flash, the WR pass rate exceeds the WF pass rate by two runs, within statistical noise).

All three agents perform worse under WR than under SF, but for different reasons: the weaker agents lose on longer workflows, and the stronger agent only on the combination with reduced guidance. For DeepSeek V4.1 Flash, longer workflows alone reduce its performance markedly: the pass rate decreases by 21 points from SF to WF, both under full guidance. We speculate that this agent has difficulty maintaining consistency over long trajectories (11 of its 26 failed workflow runs reach the five-hour limit). In contrast, for GPT-5.6 Sol, the pass rates under WF and SR are nearly identical to that under SF, and the pass rate decreases, by 13 points, only when a long task span is combined with reduced guidance, which requires the agent to plan and carry out the multi-step calculation itself.

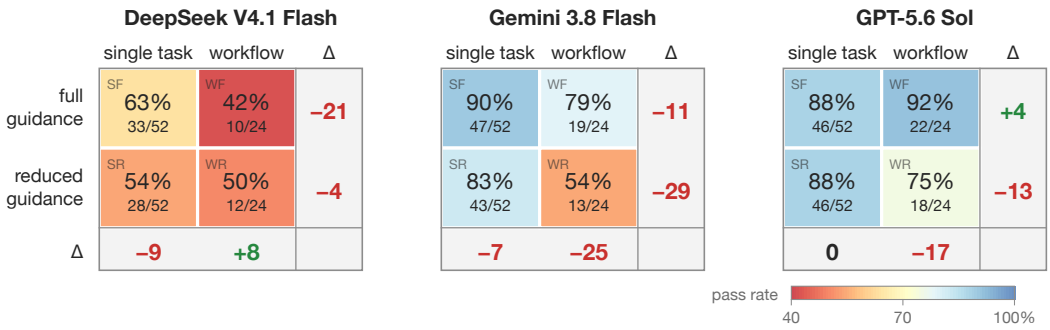


Figure 4: Pass rates of three agents under the four evaluation conditions of Section 3.2. Each cell gives the pass rate and the count behind it. The single-task column covers the 52 tasks whose reduced prompt differs from the full prompt, and the workflow column covers the eight workflows with three runs each. The Δ cells give differences in percentage points: reduced minus full guidance (bottom row) and workflow minus single task (right column, different task sets).

		scientific		*operational*			
		wrong physics	**wrong implementation**	**wrong software usage**	**format or rule violation**	**time limit or unclassified**	**failed runs**
SF 94 tasks	DeepSeek	**7** 7%	**6** 6%	**9** 10%	**8** 9%	**2** 2%	32/94
	Gemini	**3** 3%	**1** 1%	**1** 1%	**5** 5%	0	10/94
	GPT	**4** 4%	0	**3** 3%	**2** 2%	0	9/94
SR 52 tasks	DeepSeek	**5** 10%	**8** 15%	**4** 8%	**5** 10%	**2** 4%	24/52
	Gemini	**1** 2%	**4** 8%	0	**4** 8%	0	9/52
	GPT	**3** 6%	0	**1** 2%	**2** 4%	0	6/52
WF 24 runs	DeepSeek	**4** 17%	**4** 17%	0	0	**6** 25%	14/24
	Gemini	**2** 8%	**3** 12%	0	0	0	5/24
	GPT	**2** 8%	0	0	0	0	2/24
WR 24 runs	DeepSeek	**2** 8%	**5** 21%	0	0	**5** 21%	12/24
	Gemini	**4** 17%	**7** 29%	0	0	0	11/24
	GPT	**2** 8%	**3** 12%	0	0	**1** 4%	6/24

Figure 5: Failure analysis: failed runs of three agents by cause under the four evaluation conditions of Section 3.2. DeepSeek, Gemini and GPT denote DeepSeek V4.1 Flash, Gemini 3.8 Flash and GPT-5.6 Sol. Each cell gives the number of failed runs and their share of all runs in that row.

Gemini 3.8 Flash falls between these two cases: longer workflows and reduced guidance lower its pass rate by 11 and 7 points, respectively. Their combination lowers it by 36 points, far more than either alone, and its WR pass rate (54%) is close to that of DeepSeek V4.1 Flash (50%).

5.3 Workflow failures are mostly scientific, and all agents share some interface errors

We classify each failed run into three groups (Fig. 5): scientific errors, either a wrong physical choice (for example, a missing dipole correction for a polar slab) or a correct choice implemented wrongly (for example, a sign error in a Bloch phase); operational errors, either wrong software usage (for example, an input the code cannot parse) or a format or rule violation (for example, a missing answer field); and runs that reach the time limit or remain unclassified. In workflows, 38 of the 50 failed runs are scientific errors, and 11 of the other 12 are time limits of DeepSeek V4.1 Flash. In single tasks, scientific and operational errors are about equal, 42 and 44 of 90. All 18 software-usage errors occur in input-preparation tasks, where the agent writes the inputs without running the software and therefore receives no error message, whereas in a workflow the agent runs each code and can correct an input that fails.

In-depth analysis suggests that agents fail to diagnose their errors because they do not recognize anomalies that an experienced researcher would notice. In the CrN workflow of Fig. 2, 9 of the 15 failing runs use inappropriate settings for the Wannier construction. Their Wannier spreads reach 105 to 215 $Å^2$, four to eight times the 25 to 28 $Å^2$ of the passing runs, or the disentanglement does not converge, and the exchange constants computed from them deviate by up to 116%. In each of these nine runs, the anomaly is printed in the agent's own Wannier90 output, yet the agent continues to the exchange calculation without correcting the settings. We speculate that such diagnostic knowledge is implicit: it is rarely documented in the literature and is instead acquired through research practice, so it is poorly represented in the training data of language models.

Some software interfaces are difficult for all agents. For example, one task asks for the LAMMPS inputs of a WS_2/WSe_2 moiré bilayer with six atom types, and the line that sets the interaction between the two layers must give an element name for each of the six types. The offline LAMMPS manual is available to the agents, but it states no rule for the number of names and its example uses two atom types, and every agent keeps this form, with two element names instead of six. In the repeated runs, all 15 trials make the same mistake. Because the task asks for inputs without running LAMMPS, no error message reveals the mistake. Agents also fail when a calculation needs a setting beyond the code defaults, such as the van der Waals correction for a bilayer, which all four agents omit. Like the diagnostic knowledge above, the knowledge these interfaces require is implicit: the documentation available to the agents does not state it, so agents based on different LLMs fail in the same way.

6 Limitations

CompMat-Bench has the following limitations: the strongest current agents pass most single tasks under full guidance, which leaves little room to distinguish stronger models, and the conditions that still separate them, workflows and reduced guidance, have fewer runs; fixed-rule grading can wrongly reject a correct answer, and although our failure analysis found and fixed several such cases, others may remain; and the tasks do not cover all computational materials research, for example studies that propose new ideas or concepts, which are excluded at selection. We plan to add tasks to the workflow and reduced-guidance conditions in later versions.

7 Conclusion

We introduce CompMat-Bench, a benchmark of 94 tasks for evaluating AI agents on recently published computational materials research: agents are assessed on preparing the inputs and analyzing the outputs of expensive simulations, which we reproduced in advance, rather than on running them during evaluation. With full methodological guidance, agents based on three LLMs pass 66.0–90.4% of the single tasks. Varying the task span and the level of methodological guidance shows that both factors lower the pass rates of the weaker agents, whereas the strongest agent falls only when a long workflow is combined with reduced guidance. Most failures are scientific errors, and in-depth analysis reveals common failure scenarios: agents do not recognize anomalies in their own results that an experienced researcher would notice, and agents based on different LLMs lack the same implicit knowledge of software interfaces. CompMat-Bench provides a basis for comparing agents on these steps of real materials research and for analyzing their failure modes.

Reproducibility Statement

The tasks and workflows, with their prompts, input files, graders, reference answers and offline reference library, are provided as supplementary material. Appendices A to E list the source studies and tasks, the agent configuration and software versions, the workflows and complete results, and the prompts. The intermediate files of our reproductions, about 1 TB, are not released.

AI Use Statement

In this work, we used generative AI tools for decomposing the source studies, drafting task prompts and graders, and failure analysis. We have not used generative AI tools for grading or for the final

scientific decisions, including the acceptance of each task, its reference answer and the conclusions drawn; other tasks with required disclosure are not applicable to this work. Additionally, we used generative AI tools for drafting and editing the manuscript. We have reviewed all AI-assisted work and take responsibility for its final content, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Nithishwer Mouroug Anand, Wei-Tse Hsu, Kyle Vaccaro, Eden James Gage, Jonathan David Colburn, Linda Xi Phan, Minjoon Seo, Kevin Guan, and Philip C. Biggin. MDArena: Evaluating coding agents on realistic molecular dynamics workflows. *arXiv preprint arXiv:2608.02642*, 2026. URL https://arxiv.org/abs/2608.02642v1.

Ben Bogin, Kejuan Yang, Shashank Gupta, Kyle Richardson, Erin Bransom, Peter Clark, Ashish Sabharwal, and Tushar Khot. SUPER: Evaluating agents on setting up and executing tasks from research repositories. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 12622–12645, 2024. URL https://arxiv.org/abs/2409.07440v1.

Daniil A. Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. Autonomous chemical research with large language models. *Nature*, 624:570–578, 2023. doi: 10.1038/s41586-023-06792-0.

Daan Camps, Ermal Rrapaj, Katherine Klymko, Hyeongjin Kim, Kevin Gott, Siva Darbha, Jan Balewski, Brian Austin, and Nicholas J. Wright. Quantum computing technology roadmaps and capability assessment for scientific computing: An analysis of use cases from the NERSC workload. Technical Report LBNL-2001699, Lawrence Berkeley National Laboratory, September 2025. URL https://arxiv.org/abs/2509.09882v1.

Yadi Cao, Sicheng Lai, Jiahe Huang, Yang Zhang, Zach Lawrence, Rohan Bhakta, Izzy F. Thomas, Mingyun Cao, Chung-Hao Tsai, Zihao Zhou, Yidong Zhao, Hao Liu, Alessandro Marinoni, Alexey Arefiev, and Rose Yu. SimulCost: A cost-aware benchmark and toolkit for automating physics simulations with LLMs. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2603.20253v4.

Kyaw Hla Saing Chak, Bipin Bhattarai, Andrew C. Meng, and Yijia Gu. Integrated thermodynamic modeling of composition and strain tunable ferroelectricity in wurtzite $Zn_{1-x}Mg_xO$. *npj Computational Materials*, 12:154, 2026. doi: 10.1038/s41524-026-02021-0. URL https://doi.org/10.1038/s41524-026-02021-0.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng, and Aleksander Madry. MLE-bench: Evaluating machine learning agents on machine learning engineering. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.07095v6.

Mark Chen et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374.

Ziru Chen et al. ScienceAgentBench: Toward rigorous assessment of language agents for data-driven scientific discovery. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.05080v3.

DeepSeek-AI. DeepSeek-V4.1-Flash: Pushing the limits of KV cache compression. *arXiv preprint arXiv:2609.19969*, 2026. URL https://arxiv.org/abs/2609.19969v1.

Ken Deng, Xiangfei Wang, Guijing Duan, Chen Mo, Junkun Huang, Runqing Zhang, Ling Qian, Zhiguo Huang, Jize Han, and Di Luo. Towards verifiable and self-correcting AI physicists for quantum many-body simulations. *arXiv preprint arXiv:2604.00149*, 2026. URL https://arxiv.org/abs/2604.00149v2.

Nayana Devaraj, Anumita Bose, Arindom Das, Md Afsar Reja, Arijit Mandal, Awadhesh Narayan, and B. R. K. Nanda. Unlocking doping effects on altermagnetism in MnTe: Emergence of quasi-altermagnetism. *Physical Review B*, 113:104438, 2026. doi: 10.1103/k36v-91br. URL https://doi.org/10.1103/k36v-91br.

Darius A. Faroughy, Sofia Palacios Schweitzer, Ian Pang, Siddharth Mishra-Sharma, and David Shih. Collider-Bench: Benchmarking AI agents with particle physics analysis reproduction. *arXiv preprint arXiv:2605.13950*, 2026. URL https://arxiv.org/abs/2605.13950v1.

Alireza Ghafarollahi and Markus J. Buehler. Automating alloy design and discovery with physics-aware multimodal multiagent AI. *Proceedings of the National Academy of Sciences*, 122: e2414074122, 2025. doi: 10.1073/pnas.2414074122.

Paolo Giannozzi, Stefano Baroni, Nicola Bonini, Matteo Calandra, Roberto Car, Carlo Cavazzoni, Davide Ceresoli, Guido L. Chiarotti, Matteo Cococcioni, Ismaila Dabo, Andrea Dal Corso, Stefano de Gironcoli, Stefano Fabris, Guido Fratesi, Ralph Gebauer, Uwe Gerstmann, Christos Gougoussis, Anton Kokalj, Michele Lazzeri, Layla Martin-Samos, Nicola Marzari, Francesco Mauri, Riccardo Mazzarello, Stefano Paolini, Alfredo Pasquarello, Lorenzo Paulatto, Carlo Sbraccia, Sandro Scandolo, Gabriele Sclauzero, Ari P. Seitsonen, Alexander Smogunov, Paolo Umari, and Renata M. Wentzcovitch. QUANTUM ESPRESSO: A modular and open-source software project for quantum simulations of materials. *Journal of Physics: Condensed Matter*, 21:395502, 2009. doi: 10.1088/0953-8984/21/39/395502.

Google DeepMind. Gemini 3.8 Flash model card. https://deepmind.google/models/model-cards/gemini-3-8-flash/, 2026. Model card published 2026-09-02; PDF at https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-8-Flash-Model-Card.pdf.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2009.03300.

Maximilian Hofer, Christopher Fuchs, Moritz Siebert, Christian Berger, Lena Fürst, Martin P. Stehno, Steffen Schreyeck, Hartmut Buhmann, Tobias Kießling, Wouter Beugeling, and Laurens W. Molenkamp. Self-consistent electrostatic modeling of gated narrow-gap topological insulators. *Nano Letters*, 26:3042–3048, 2026. doi: 10.1021/acs.nanolett.5c05762. URL https://doi.org/10.1021/acs.nanolett.5c05762.

Zhengding Hu, Kuntal Talit, Zhen Wang, Haseeb Ahmad, Yichen Lin, Prabhleen Kaur, Christopher Lane, Elizabeth A. Peterson, Zhiting Hu, Elizabeth A. Nowadnick, and Yufei Ding. TritonDFT: Automating DFT with a multi-agent framework. *arXiv preprint arXiv:2603.03372*, 2026. URL https://arxiv.org/abs/2603.03372v2.

Ziyang Huang, Yi Cao, Ali K. Shargh, Jing Luo, Ruidong Mei, Mohd Zaki, Zhan Liu, Wyatt Bunstine, William Jurayj, Somdatta Goswami, Tyrel McQueen, Michael Shields, Jaafar El-Awady, Paulette Clancy, Benjamin Van Durme, Nicholas Andrews, William Walden, and Daniel Khashabi. Can coding agents reproduce findings in computational materials science? In *Conference on Language Modeling*, 2026a. URL https://arxiv.org/abs/2605.00803v2.

Ziyang Huang, Yi Cao, Ali K. Shargh, Jing Luo, Ruidong Mei, Mohd Zaki, Zhan Liu, Wyatt Bunstine, William Jurayj, Somdatta Goswami, Tyrel McQueen, Michael Shields, Jaafar El-Awady, Paulette Clancy, Benjamin Van Durme, Nicholas Andrews, William Walden, and Daniel Khashabi. AutoMat dataset. Hugging Face dataset jhu-clsp/AutoMat, revision 58d2fe48, https://huggingface.co/datasets/jhu-clsp/AutoMat, 2026b.

Tousif Islam, Digvijay Wadekar, and Zihan Zhou. gwBenchmarks: Stress-testing LLM agents on high-precision gravitational wave astronomy. *arXiv preprint arXiv:2605.11269*, 2026. URL https://arxiv.org/abs/2605.11269v2.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2403.07974v2.

Xuance Jiang, Sayed Ali Akbar Ghorashi, Deyu Lu, and Jennifer Cano. Altermagnetism induced surface Chern insulator. *Nano Letters*, 26:1327–1333, 2026. doi: 10.1021/acs.nanolett.5c05341. URL https://doi.org/10.1021/acs.nanolett.5c05341.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.06770.

Sayash Kapoor, Benedikt Stroebl, Zachary S. Siegel, Nitya Nadgir, and Arvind Narayanan. AI agents that matter. *Transactions on Machine Learning Research*, 2025. URL https://arxiv.org/abs/2407.01502v1.

Seo-Jin Kim, Zdeněk Jirák, Jiří Hejtmánek, Karel Knížek, Helge Rosner, and Kyo-Hoon Ahn. Semiclassical model of magnons in double-layered antiferromagnets. *Physical Review B*, 113:144429, 2026. doi: 10.1103/gj66-4yhj. URL https://doi.org/10.1103/gj66-4yhj.

G. Kresse and J. Furthmüller. Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set. *Physical Review B*, 54:11169–11186, 1996. doi: 10.1103/PhysRevB.54.11169.

Vinay Kumar, Satyendra Rajput, Mausam, and N. M. Anoop Krishnan. MDGYM: Benchmarking AI agents on molecular simulations. *arXiv preprint arXiv:2605.08941*, 2026. URL https://arxiv.org/abs/2605.08941v1.

Xingfu Li, Sunny Gupta, and Boris I. Yakobson. Undulation-induced moiré superlattices with one-dimensional polarization domains and flat bands in two-dimensional bilayer semiconductors. *Physical Review B*, 112:115308, 2025. doi: 10.1103/bxw4-7y5c. URL https://doi.org/10.1103/bxw4-7y5c.

Yifei Li, Yichong Chen, Guihua Tang, Livio Gibelli, and Matthew K. Borg. Scattering matrix formalism for interface heat conduction based on Monte Carlo evaluation of the phonon Boltzmann transport equation. *Physical Review B*, 113:205304, 2026. doi: 10.1103/kz9s-y611. URL https://doi.org/10.1103/kz9s-y611.

Siyu Liu, Bo Hu, Beilin Ye, Jiamin Xu, David J. Srolovitz, and Tongqi Wen. MatTools: Benchmarking large language models for materials science tools. *arXiv preprint arXiv:2505.10852*, 2025. URL https://arxiv.org/abs/2505.10852v2.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The AI scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024. URL https://arxiv.org/abs/2408.06292v3.

Bodhisattwa Prasad Majumder, Harshit Surana, Dhruv Agarwal, Bhavana Dalvi Mishra, Abhijeetsingh Meena, Aryan Prakhar, Tirth Vora, Tushar Khot, Ashish Sabharwal, and Peter Clark. DiscoveryBench: Towards data-driven discovery with large language models. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2407.01725v1.

Andre Niyongabo Rubungo, Kangming Li, Jason Hattrick-Simpers, and Adji Bousso Dieng. LLM4Mat-Bench: Benchmarking large language models for materials property prediction. *Machine Learning: Science and Technology*, 6(2):020501, 2025. doi: 10.1088/2632-2153/add3bb.

OpenAI. Codex CLI: A lightweight coding agent that runs in your terminal. https://github.com/openai/codex, 2025. Official GitHub repository; no CITATION.cff or paper is provided; initial public release 2025-04-16.

OpenAI. GPT-5.6 system card. https://deploymentsafety.openai.com/gpt-5-6, 2026. Published 2026-07-09; covers the Sol, Terra and Luna models of the GPT-5.6 family.

Selçuk Parlak, Abhishek Kumar, Runhan Li, Maia G. Vergniory, and Ion Garate. Tight-binding and density-functional study of the Raman tensor in two-dimensional massive Dirac fermion systems. *Physical Review B*, 113:085431, 2026. doi: 10.1103/m1p1-w3w4. URL https://doi.org/10.1103/m1p1-w3w4.

Jason Priem, Heather Piwowar, and Richard Orr. OpenAlex: A fully-open index of scholarly works, authors, venues, institutions, and concepts. *arXiv preprint arXiv:2205.01833*, 2022. URL https://arxiv.org/abs/2205.01833v2.

Shi Qiu, Junyi Deng, Yiwei Deng, Haoran Dong, Jieyu Fu, Mao Li, Zeyu Li, Zhaolong Zhang, Huiwen Zheng, Leidong Bao, Anqi Lv, Zihan Mo, Yadi Niu, Yiyang Peng, Yu Tian, Yili Wang, Ziyu Wang, Zi-Yu Wang, Jiashen Wei, Liuheng Wu, Aoran Xue, Leyi Yang, Guanglu Yuan, Xiarui Zhan, Jingjun Zhang, Zifan Zheng, Pengfei Liu, Linrui Zhen, Kaiyang Li, Qichang Li, Ziheng Zhou, Guo-En Nian, Yunwei Xiao, Qing-Hong Cao, Linjie Dai, Xu Feng, Peng Gao, Ying Gu, Chang Liu, Jia Liu, Ming-xing Luo, Yan-Qing Ma, Liang-You Peng, Huichao Song, Shufeng Wang, Chenxu Wang, Tao Wang, Yi-Nan Wang, Chengyin Wu, Pengwei Zhao, and Hua Xing Zhu. PRBench: End-to-end paper reproduction in physics research. *arXiv preprint arXiv:2603.27646*, 2026. URL https://arxiv.org/abs/2603.27646v1.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level Google-proof Q&A benchmark. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

Bin Shao, Jixiang Li, Xinyue Zhang, Baishun Yang, Zhiyang Liu, and Weichao Wang. INCAR-Bench: A benchmark for scientific configuration in VASP INCAR by large language models. *arXiv preprint arXiv:2606.23571*, 2026. URL https://arxiv.org/abs/2606.23571v2.

Zachary S. Siegel, Sayash Kapoor, Nitya Nadgir, Benedikt Stroebl, and Arvind Narayanan. CORE-Bench: Fostering the credibility of published research through a computational reproducibility agent benchmark. *Transactions on Machine Learning Research*, 2024. URL https://arxiv.org/abs/2409.11363v2.

Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, Johannes Heidecke, Amelia Glaese, and Tejal Patwardhan. PaperBench: Evaluating AI's ability to replicate AI research. In *International Conference on Machine Learning*, 2025. URL https://arxiv.org/abs/2504.01848v3.

Tsubasa Takagi, Hikaru Watanabe, Ryutaro Yoshimi, Yuki Sato, Shingo Toyoda, Atsushi Tsukazaki, Kei S. Takahashi, Masashi Kawasaki, Yoshinori Tokura, and Naoki Ogawa. Quantum geometry in low-energy linear and nonlinear optical responses of the magnetic Rashba semiconductor (Ge,Mn)Te. *Physical Review B*, 113:104417, 2026. doi: 10.1103/dp31-y358. URL https://doi.org/10.1103/dp31-y358.

Sunil Taper, Bo Cai, Nives Strkalj, Bartomeu Monserrat, and Giuliana Di Martino. Role of local cation environment on the formation energy of oxygen vacancies in ferroelectric $Hf_{1-x}Zr_xO_2$. *Physical Review B*, 113:134114, 2026. doi: 10.1103/vz64-n5xh. URL https://doi.org/10.1103/vz64-n5xh.

Aidan P. Thompson, H. Metin Aktulga, Richard Berger, Dan S. Bolintineanu, W. Michael Brown, Paul S. Crozier, Pieter J. in 't Veld, Axel Kohlmeyer, Stan G. Moore, Trung Dac Nguyen, Ray Shan, Mark J. Stevens, Julien Tranchida, Christian Trott, and Steven J. Plimpton. 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.

Minyang Tian, Luyu Gao, Shizhuo Dylan Zhang, Xinan Chen, Cunwei Fan, Xuefei Guo, Roland Haas, Pan Ji, Kittithat Krongchon, Yao Li, Shengyan Liu, Di Luo, Yutao Ma, Hao Tong, Kha Trinh, Chenyu Tian, Zihan Wang, Bohao Wu, Yanyu Xiong, Shengzhu Yin, Minhui Zhu, Kilian Lieret, Yanxin Lu, Genglin Liu, Yufeng Du, Tianhua Tao, Ofir Press, Jamie Callan, Eliu Huerta, and Hao Peng. SciCode: A research coding benchmark curated by scientists. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, 2024. URL https://arxiv.org/abs/2407.13168v1.

Yanzhen Wang, Yiyang Jiang, Diana Golovanova, Kamal Das, Hyeonhu Bae, Yufei Zhao, Huu-Thong Le, Abhinava Chatterjee, Yunzhe Liu, Chao-Xing Liu, Felipe H. da Jornada, Binghai Yan,

and Xiao-Liang Qi. QMBench: A research level benchmark for quantum materials research. *arXiv preprint arXiv:2512.19753*, 2025. URL https://arxiv.org/abs/2512.19753v1.

Youwen Wang, Nanya Gao, and Qingjun Tong. Origin of moiré potentials in WS_2/WSe_2 heterobilayers: Contributions from lattice reconstruction and interlayer charge transfer. *Physical Review B*, 113:085408, 2026a. doi: 10.1103/hhdc-yb5s. URL https://doi.org/10.1103/hhdc-yb5s.

Zhen Wang, Fan Bai, Zhongyan Luo, Jinyan Su, Kaiser Sun, Xinle Yu, Jieyuan Liu, Kun Zhou, Claire Cardie, Mark Dredze, Zhiting Hu, and Eric P. Xing. FIRE-Bench: Evaluating AI agents on the rediscovery of scientific insights. In *International Conference on Machine Learning*, 2026b. URL https://arxiv.org/abs/2602.02905v2.

Himal Wijekoon, Pierre Hirel, and Anna Grünebohm. Ferroelectric switching at edge dislocations in $BaTiO_3$ modeled at the atomic scale. *Physical Review B*, 113:144108, 2026. doi: 10.1103/jrql-63kl. URL https://doi.org/10.1103/jrql-63kl.

Huan Wu. Nonperturbative isotope effect on light-matter interaction in boron arsenide. *Physical Review B*, 113:134316, 2026. doi: 10.1103/yqg4-5gf7. URL https://doi.org/10.1103/yqg4-5gf7.

Zhenyang Xie, Jitong Song, Yuntao Wu, Yuanji Xu, and Fuyang Tian. Local magnetic structure in fully and partially ordered V_2XAl Heusler alloys (X = Cr, Mn, Fe, Co, Ni). *Physical Review B*, 113:064410, 2026. doi: 10.1103/4gsl-4lxv. URL https://doi.org/10.1103/4gsl-4lxv.

Qichen Xu and Anna Delin. A general optimization framework for mapping local transition-state networks. *npj Computational Materials*, 12:112, 2026. doi: 10.1038/s41524-026-01985-3. URL https://doi.org/10.1038/s41524-026-01985-3.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2405.15793v3.

Penghui Yang, Zhonghan Zhang, Yue Li, Xinrun Wang, Yanchen Deng, Yuhao Lu, Bijun Tang, Zheng Liu, and Bo An. AutoDFT: A closed-loop multi-agent framework for autonomous DFT calculations. *arXiv preprint arXiv:2605.26179*, 2026. URL https://arxiv.org/abs/2605.26179v2.

Christine Ye, Sihan Yuan, Suchetha Cooray, Steven Dillmann, Ian L. V. Roque, Dalya Baron, Philipp Frank, Sergio Martin-Alvarez, Nolan Koblischke, Frank J Qu, Diyi Yang, Risa Wechsler, and Ioana Ciuca. ReplicationBench: Can AI agents replicate astrophysics research papers? *arXiv preprint arXiv:2510.24591*, 2025. URL https://arxiv.org/abs/2510.24591v2.

Mohd Zaki, Jayadeva, Mausam, and N. M. Anoop Krishnan. MaScQA: Investigating materials science knowledge of large language models. *Digital Discovery*, 3:313–327, 2024. doi: 10.1039/D3DD00188A.

C. Zhang and B. I. Yakobson. MatClaw: An autonomous code-first LLM agent for end-to-end materials exploration. *arXiv preprint arXiv:2604.02688*, 2026.

Minhui Zhu, Minyang Tian, Xiaocheng Yang, Tianci Zhou, Lifan Yuan, Penghao Zhu, Eli Chertkov, Shengyan Liu, Yufeng Du, Ziming Ji, Indranil Das, Qingzhi Chen, Junyi Cao, Yufeng Du, Jiabin Yu, Peixue Wu, Jinchen He, Yifan Su, Yikun Jiang, Yujie Zhang, Chang Liu, Ze-Min Huang, Weizhen Jia, Yunkai Wang, Farshid Jafarpour, Yong Zhao, Xinan Chen, Jessie Shelton, Aaron W. Young, John Bartolotta, Wenchao Xu, Yue Sun, Anjun Chu, Victor Colussi, Chris Akers, Nathan Brooks, Wenbo Fu, Jinchao Zhao, Marvin Qi, Anqi Mu, Yubo Yang, Allen Zang, Yang Lyu, Peizhi Mai, Christopher Wilson, Xuefei Guo, Juntai Zhou, Daniel Inafuku, Chi Xue, Luyu Gao, Ze Yang, Yaïr Hein, Yonatan Kahn, Kevin Zhou, Di Luo, John Drew Wilson, Jarrod T. Reilly, Dmytro Bandak, Ofir Press, Liang Yang, Xueying Wang, Hao Tong, Nicolas Chia, Eliu Huerta, and Hao Peng. Probing the critical point (CritPt) of AI reasoning: a frontier physics research benchmark. *arXiv preprint arXiv:2509.26574*, 2025. URL https://arxiv.org/abs/2509.26574v4.

A SOURCE STUDIES AND TASK LIST

Table A1 lists the 15 source studies of Section 3.1, and Table A2 lists the 94 single tasks derived from them. The study numbers of Table A1 identify the source study of each task in Table A2. Each task is identified by the publication month of its source study and a sequence number within that month, followed by a short name. For example, single task 013 of Fig. 2 is 202603_013.

Table A1: The 15 source studies. Month is the publication month. Methods lists the method families of Fig. 3 for the tasks of each study, shortened to their first words. Graph steps counts the research steps in the full graph of each study, Selected counts the steps in its selected sub-graph, and Tasks counts the single tasks built from these steps (Section 3.1).

No.	Study	Journal	Month	Methods	Graph steps	Selected	Tasks
1	(Li et al., 2025)	Phys. Rev. B	2025-09	DFT, Structure	10	4	7
2	(Jiang et al., 2026)	Nano Lett.	2026-01	Model Hamiltonians, DFT	7	5	5
3	(Xie et al., 2026)	Phys. Rev. B	2026-01	DFT, Structure, Spin models	10	5	8
4	(Devaraj et al., 2026)	Phys. Rev. B	2026-02	DFT, Model Hamiltonians, Structure	12	5	7
5	(Hofer et al., 2026)	Nano Lett.	2026-02	Model Hamiltonians	6	3	4
6	(Parlak et al., 2026)	Phys. Rev. B	2026-02	Phonons, DFT	10	3	6
7	(Takagi et al., 2026)	Phys. Rev. B	2026-02	Model Hamiltonians	13	3	4
8	(Wang et al., 2026a)	Phys. Rev. B	2026-02	DFT, Atomistic, Continuum	18	5	8
9	(Xu & Delin, 2026)	npj Comput. Mater.	2026-02	Atomistic, Spin models	7	3	3
10	(Chak et al., 2026)	npj Comput. Mater.	2026-03	Berry phase, Continuum	7	5	7
11	(Kim et al., 2026)	Phys. Rev. B	2026-03	Spin models, DFT	7	5	8
12	(Li et al., 2026)	Phys. Rev. B	2026-03	Phonon transport, Structure	11	7	9
13	(Taper et al., 2026)	Phys. Rev. B	2026-03	Defects, Bonding, DFT, Structure	6	3	6
14	(Wijekoon et al., 2026)	Phys. Rev. B	2026-03	Atomistic	8	4	6
15	(Wu, 2026)	Phys. Rev. B	2026-03	Phonons, Continuum	9	5	6
Total					141	65	94

Table A2: The 94 single tasks. A dash in Codes marks a task with no named code, where the agent writes the calculation itself in Python. Checks counts the checks in the grading rules of the task. QE: Quantum ESPRESSO; phonopy includes phono3py.

Task	Short name	Study	Kind	Codes	Checks
202509_001	hbn-sliding-sweep-inputs	1	input preparation	VASP	14
202509_002	hbn-sliding-potential-step	1	output analysis	VASP	4
202509_003	hbn-undulated-sliding-ratio	1	run and analyze	–	8
202509_004	bn-nanotube-inputs	1	input preparation	VASP	15
202509_005	bn-nanotube-flexoelectric-constant	1	output analysis	VASP	1
202509_006	hbn-flexo-sliding-decomposition-inputs	1	input preparation	VASP	14
202509_007	hbn-undulated-step-decomposition	1	output analysis	VASP	14
202601_001	v2coal-heusler-structures	3	run and analyze	–	2
202601_002	v2coal-relax-energy-inputs	3	input preparation	VASP	14
202601_003	v2coal-formation-energies-ordering	3	output analysis	VASP	10

Task	Short name	Study	Kind	Codes	Checks
202601_004	v2coal-l21-abacus-inputs	3	input preparation	ABACUS	12
202601_005	v2coal-l21-magnetic-moment	3	output analysis	ABACUS	2
202601_006	v2coal-xa-abacus-inputs	3	input preparation	ABACUS	13
202601_007	v2coal-xa-site-moments	3	output analysis	ABACUS	3
202601_008	v2coal-heisenberg-exchange	3	run and analyze	ABACUS, TB2J	5
202601_009	snte-110-surface-dirac-cones	2	run and analyze	–	1
202601_010	ruo2-snte-interface-inputs	2	input preparation	VASP	20
202601_011	coupled-slab-surface-splittings	2	run and analyze	–	2
202601_012	surface-cone-hall-contributions	2	run and analyze	–	2
202601_013	slab-hall-interface-distribution	2	run and analyze	–	2
202602_001	hgte-well-hartree-inputs	5	input preparation	kdotpy	15
202602_002	hgte-well-hartree-analysis	5	output analysis	–	10
202602_003	hgte-well-surface-dispersion	5	run and analyze	kdotpy	5
202602_004	hgte-well-sdh-densities	5	run and analyze	–	10
202602_005	gete-wannier-inputs	7	input preparation	QE, Wannier90	29
202602_006	gete-fermi-levels	7	output analysis	Wannier90	5
202602_007	gete-interband-jdos	7	run and analyze	Wannier90	8
202602_008	gete-optical-conductivity	7	run and analyze	Wannier90	5
202602_009	rucl2-dftu-bands-inputs	6	input preparation	VASP	17
202602_010	rucl2-valley-gaps	6	output analysis	VASP	4
202602_011	rucl2-phonon-forces-inputs	6	input preparation	VASP	14
202602_012	rucl2-a1p-phonon-mode	6	output analysis	VASP, phonopy	5
202602_013	rucl2-frozen-phonon-optics-inputs	6	input preparation	VASP	17
202602_014	rucl2-magneto-raman-tensor	6	output analysis	VASP	6
202602_015	ws2-wse2-moire-lammps-inputs	8	input preparation	LAMMPS	17
202602_016	ws2-wse2-moire-reconstruction-fields	8	output analysis	LAMMPS	17
202602_017	ws2-wse2-bilayer-soc-band-inputs	8	input preparation	VASP	18
202602_018	ws2-wse2-band-offsets-at-k	8	output analysis	VASP	5
202602_019	tmd-monolayer-strain-sweep-inputs	8	input preparation	VASP	13
202602_020	tmd-strain-deformation-potentials	8	output analysis	VASP	6
202602_021	moire-piezoelectric-bound-charge	8	run and analyze	LAMMPS	7
202602_022	moire-screened-piezo-potential	8	run and analyze	–	6
202602_023	mnte-altermagnet-pipeline-inputs	4	input preparation	QE	17
202602_024	mnte-altermagnet-spin-splitting	4	output analysis	QE	6
202602_025	doped-mnte-space-group-classification	4	run and analyze	–	2
202602_026	doped-mnte-supercell-band-inputs	4	input preparation	QE	18
202602_027	doped-mnte-momentum-spin-splitting	4	output analysis	QE	16
202602_028	tb-chain-spin-splitting	4	run and analyze	–	9
202602_029	substituted-tb-chain-spectrum	4	run and analyze	–	13
202602_030	spin-lattice-hessian-minimum-modes	9	run and analyze	–	9
202602_031	ni-island-eam-saddle-points	9	run and analyze	–	12
202602_032	ni-island-mlp-saddle-certification	9	run and analyze	MACE	8
202603_001	zno-mgo-phase-boundaries	10	run and analyze	–	6
202603_002	zno-finite-strain-inputs	10	input preparation	QE	18
202603_003	znmgo-elastic-piezoelectric	10	output analysis	QE	28
202603_004	znmgo-berry-phase-inputs	10	input preparation	QE	16
202603_005	znmgo-spontaneous-polarization	10	output analysis	QE	7
202603_006	znmgo-landau-devonshire-model	10	run and analyze	–	6
202603_007	znmgo-strain-composition-phase-map	10	run and analyze	–	6
202603_008	afm-chain-magnon-spectrum	11	run and analyze	–	2
202603_009	crn-ldau-structure-inputs	11	input preparation	VASP	14
202603_010	crn-relaxed-structure-exchange-paths	11	output analysis	VASP	15
202603_011	crn-wannier-exchange-inputs	11	input preparation	VASP, Wannier90, TB2J	29
202603_012	crn-nearest-neighbour-exchange	11	output analysis	TB2J	12
202603_013	crn-linear-spin-wave-spectrum	11	run and analyze	–	2
202603_014	crn-monte-carlo-deck	11	input preparation	VAMPIRE	23
202603_015	crn-neel-temperature	11	output analysis	VAMPIRE	2
202603_016	si-ge-interface-cells	12	run and analyze	–	4
202603_017	si-ge-empirical-phonons	12	run and analyze	LAMMPS, phonopy	16

Task	Short name	Study	Kind	Codes	Checks
202603_018	si-ge-phonon-force-inputs	12	input preparation	VASP	8
202603_019	si-ge-abinitio-phonons	12	output analysis	VASP, phonopy	16
202603_020	si-ge-mode-resolved-transmission	12	run and analyze	–	45
202603_021	si-ge-diffuse-mismatch-transmission	12	run and analyze	–	4
202603_022	si-ge-reflection-specularity	12	run and analyze	–	4
202603_023	si-ge-phonon-mc-tables	12	input preparation	–	12
202603_024	si-ge-slab-thermal-conductivity	12	output analysis	–	8
202603_025	hzo-disordered-supercell-inputs	13	input preparation	VASP	15
202603_026	hzo-oxygen-environments	13	output analysis	–	17
202603_027	hzo-oxygen-vacancy-inputs	13	input preparation	VASP	14
202603_028	hzo-vacancy-charge-thermodynamics	13	output analysis	VASP	55
202603_029	hzo-bonding-analysis-inputs	13	input preparation	VASP, LOBSTER	17
202603_030	hzo-oxygen-sublattice-bonding	13	output analysis	LOBSTER	46
202603_031	batio3-hysteresis-lammps-inputs	14	input preparation	LAMMPS	35
202603_032	batio3-polarization-hysteresis	14	output analysis	LAMMPS	7
202603_033	batio3-dislocation-dipole-inputs	14	input preparation	LAMMPS	22
202603_034	batio3-dislocation-dipole-relaxed	14	output analysis	LAMMPS	16
202603_035	batio3-dislocation-strain-maps	14	run and analyze	LAMMPS	14
202603_036	batio3-dislocation-core-polarization	14	run and analyze	LAMMPS	36
202603_037	bas-force-constant-inputs	15	input preparation	QE	14
202603_038	bas-harmonic-force-constants	15	output analysis	QE, ALAMODE	21
202603_039	bas-natural-isotope-dielectric	15	run and analyze	–	11
202603_040	bas-isotope-dielectric-dependence	15	run and analyze	–	12
202603_041	bas-surface-phonon-polariton-gap	15	run and analyze	–	11
202603_042	bas-near-field-heat-flux	15	run and analyze	–	19

B AGENT CONFIGURATION AND RUN ENVIRONMENT

Table B1 gives the settings of each agent, and Table B2 the software available to it in the container.

Table B1: Agent configuration. Steps is the step limit per single task, Context the context window, Output the maximum output tokens per call, and Effort the reasoning effort. For Codex CLI, none means that its native tool loop has no step cap, and the dash means its default output limit.

Model	Harness	Steps	Context	Output	Effort
DeepSeek V4.1 Flash	MatClaw	100	800k	131,072	high
Gemini 3.8 Flash	MatClaw	500	256k	65,536	high
GPT-5.6 Sol	MatClaw	500	256k	128,000	high
GPT-5.6 Sol	Codex CLI 0.154.0	none	256k	–	high

MatClaw runs each attempt as a code-first loop: one step is one model reply, a JSON object whose code field the harness executes in a persistent Python kernel, returning the printed output as the next observation. From that code the agent can call file, search and shell tools; web and job-submission tools are disabled. A run ends when the agent calls `final_answer`, reaches its step limit, or reaches the time limit. Codex CLI receives the same task text and runs its native tool loop with web search disabled.

Each run sees only its workspace: the task files, the offline user manuals and read-only software. The graders and reference values are kept outside the container and are applied to the submission after the run. The container has no network route except to the model API. The manuals come from one reference library, so every run of every agent reads the same documentation.

Table B2: Software in the container. Single tasks and Workflows state whether the agent can run each simulation code: yes, no, or source when only a read-only source tree is staged. Where it cannot, the agent has the code's manual. The Python packages are shared by every run.

Simulation code	Version	Single tasks	Workflows
VASP	6.1.0; 6.4.1	no	yes
Quantum ESPRESSO	7.5	no	yes
ABACUS	3.10.1	no	yes
Wannier90	3.1.0	no	yes
LAMMPS	22 Jul 2025, update 4	yes	yes
phono3py, with phonopy	4.4.0	yes	yes
MACE, with PyTorch	0.3.16; 2.13.0 (CPU)	yes	yes
ALAMODE	1.4.2	yes	yes
TB2J	0.9.12.26	yes	yes
TB2J	0.9.20	source	yes
kdotpy	1.4.1	source	no
VAMPIRE	7.0.0	source	no
LOBSTER	–	no	no
Python package	**Version**	**Python package**	**Version**
Python	3.12.14	spglib	2.7.0
numpy	2.4.6	matplotlib	3.10.9
scipy	1.17.1	pandas	3.0.3
ase	3.27.0	h5py	3.16.0
pymatgen	2026.5.4	sympy	1.14.0

C WORKFLOWS AND COMPLETE RESULTS

Table C1 defines the eight workflows of Section 3.2: the source study, the steps in order, and the single task that tests the same step.

Table C1: The eight workflows. Study numbers refer to Table A1 and task numbers to Table A2; a range such as 202601_004 to 007 lists the input and analysis tasks of one step. A dash marks a step that has no single task.

Workflow	Study	Step	Description	Single task
C01	8	a	Build and relax the WS_2/WSe_2 moiré	202602_015 to 016
		b	Compute the piezoelectric bound charge	202602_021
		c	Compute the screened electron potential	202602_022
C02	4	a	Classify Se-pair substitutions by space group	202602_025
		b	Build a tight-binding model per family	–
		c	Compute the spin splittings	202602_028 to 029
C03	15	a	Compute BAs force constants from DFT forces	202603_037 to 038
		b	Compute the isotope-disordered dielectric function	202603_039
		c	Compute gap modes and near-field transmission	202603_041
C04	12	a	Relax the Si/Ge interface and compute force constants	–
		b	Compute the phonon reflection amplitudes	202603_020
		c	Compute the specularity of reflection	202603_022
C05	10	a	Relax five ordered $Zn_{1-x}Mg_xO$ cells	202603_002 to 003
		b	Compute Berry-phase polarization paths	202603_004 to 005
		c	Fit the Landau free-energy model	202603_006
C06	11	a	Compute the magnetic ground state and Wannier functions	202603_011
		b	Derive the four exchange constants	202603_012
		c	Compute the spin-wave energies	202603_013

Workflow	Study	Step	Description	Single task
C07	3	a	Build the two chemical orderings of V_2CoAl	202601_001
		b	Relax both orderings with DFT	202601_002 to 003
		c	Compute the magnetic states of both orderings	202601_004 to 007
		d	Compute nearest-neighbour exchange couplings	202601_008
C08	9	a	Find the Ni-island saddle point with EAM	202602_031
		b	Refine the saddle with a machine-learned potential	–
		c	Certify its curvature and connected minima	202602_032

Figure C1 gives every attempt behind Table 2, Table 3 and Fig. 4: the 94 single tasks under SF and SR, the eight workflows under WF and WR, and the 20 repeated tasks.

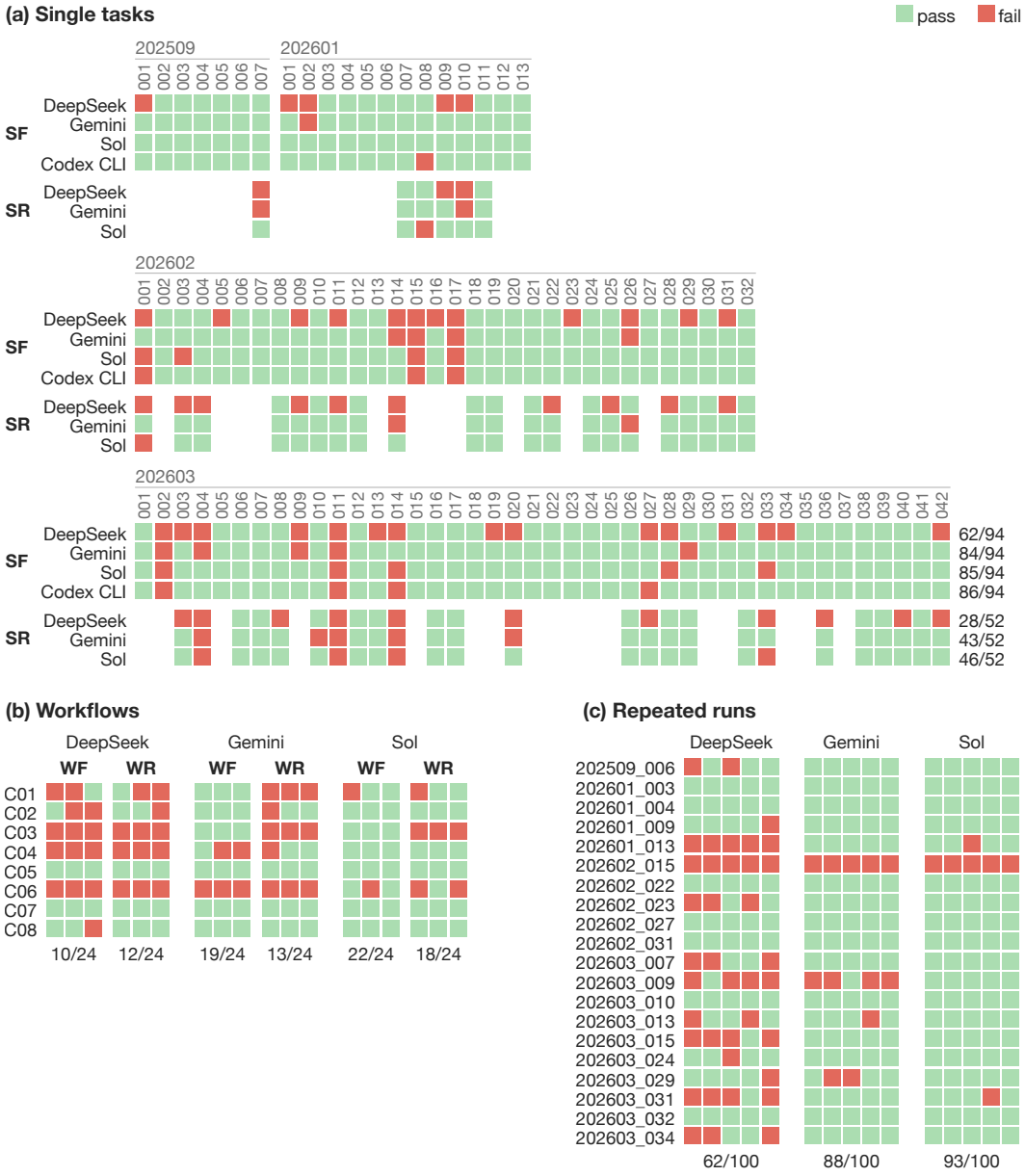


Figure C1: Complete results. Each square is one run. DeepSeek and Gemini are DeepSeek V4.1 Flash and Gemini 3.8 Flash; Sol and Codex CLI are GPT-5.6 Sol in the two harnesses of Table B1. (a) Columns are the 94 single tasks of Table A2, labelled by task number. SR rows are blank for the 42 tasks whose prompt has no reduced form.

D ERRORS SHARED ACROSS AGENTS

This appendix lists the errors that several agents make in the same way on the same task. Section 5.3 shows one of them, a LAMMPS line that all four agents write wrongly. Of the 94 single tasks under full guidance, 14 are failed by two or more agents. Table D1 lists six tasks on which the failing agents write the same wrong input. On five of these tasks the wrong input appears in every failing run; on the sixth it appears in three of the four.

Table D1: Errors shared across agents on single tasks under full guidance. Task numbers refer to Table A2. Agents names those that fail the task with the error shown; an agent not named passes the task or fails it in another way.

Task	Code	What the agents write	What the task needs	Agents
202602_015, WS_2/WSe_2 moiré bilayer, LAMMPS inputs	LAMMPS	Interlayer potential `kolmogorov/crespi/z` with two element names	One element name per atom type, six for this cell	all four
202602_017, WS_2/WSe_2 bilayer, band-structure inputs	VASP	Relaxation without a van der Waals correction (`IVDW` unset)	A dispersion correction for the van der Waals-bonded bilayer	all four
202603_011, CrN Wannier functions to exchange constants, inputs	VASP, Wannier90, TB2J	Spin-channel files named `wannier90.up` and `wannier90.dn`	The names VASP writes, `wannier90.1` and `wannier90.2`	DeepSeek, Sol, Codex CLI
202603_014, CrN Monte Carlo deck	VAMPIRE	No `unit-cell-category` in the material file, so only one Cr sublattice is built	Each sublattice assigned its own category	DeepSeek, Sol, Codex CLI
202601_002, V_2CoAl relaxation inputs	VASP	`LASPH` not set	`LASPH = .TRUE.` for 3d elements	DeepSeek, Gemini
202603_009, CrN DFT+U structure inputs	VASP	`LASPH` not set	`LASPH = .TRUE.` for 3d elements with DFT+U	DeepSeek, Gemini

Section 5.3 names the cause: agents based on different LLMs lack the same implicit knowledge of software interfaces. Each row of Table D1 is such a case. The correct form is known from practice with the code: an example in the manual does not generalise to more atom types, a default must be overridden, or a tag must be set for a class of elements. Given the same documentation, the agents converge on the same wrong form.

E PROMPTS

E.1 DECOMPOSITION PROMPT

The prompt below, given one source paper, produces the graph of research steps of Section 3.1.

```
You are a research-paper expert specializing in methodological analysis, problem
    decomposition, and computational-dependency analysis of scientific studies.

**GOAL**
Produce, in a SINGLE response, a directed acyclic graph (DAG) of the paper's
research-problem decomposition. Work in two INTERNAL phases, then output once.

PHASE 1 -- Construct the research-problem tree (do not output yet):
- Fully comprehend the paper; understand its core research problems and experiments.
- Build a hierarchical research-problem tree that mirrors the authors' logic:
  * Root node -- the single, broadest research problem tackled by the paper.
  * Intermediate nodes -- progressively narrower sub-problems / objectives.
  * Leaf nodes -- fully specified experimental tasks (datasets, models, metrics,
    protocols) that each map to a MAIN-TEXT Figure or Table.
- Continue decomposing until every branch ends in such a leaf. There is no depth limit.

PHASE 2 -- Annotate the COMPLETE tree into a DAG (only after Phase 1 is finished, so the
full set of leaves is already known):
- For each leaf node, append: `id`, `cost_class`, `consumes`.
- For each intermediate node (root included), append: `id`.
- Because the whole tree already exists before you annotate, every `consumes` reference
  must point to a leaf you have already identified in Phase 1.

Output ONLY the final annotated JSON (the Phase 1 tree is internal scratch work).

---

### Phase 1 -- Reading & Extraction Rules
1. **Locate the root** in the title, abstract, introduction, or discussion.
2. **Recursively decompose** each problem by following explicit textual cues (headings,
    "first... second...", "to this end...", method overviews, figure/table captions,
    bullet lists, etc.).
3. **Identify leaves**: a node is a leaf *only if* it describes a concrete experiment and
    its `evidence.figure` or `evidence.table` field cites a **main-text** Figure or Table.
    A section reference alone is not sufficient. Supplementary material (SI, SM, Appendix,
    Supplementary Figs./Tables) may appear only inside a leaf's `protocol_or_setup` field.
4. **Not every main-text figure is a leaf.** A leaf produces a new physical quantity by
    running a calculation, a measurement, or its own equation on upstream output. The
    following are never leaves; fold each into the leaf whose output it reads, by adding
    the figure or table to that leaf's `evidence` and the quantity to its `metrics` and
    `conclusion`:
    (1) a figure or panel that only re-plots, visualizes, or takes summary statistics of
        another leaf's output;
    (2) a comparison of the paper's own results with literature or experimental values
        that the paper did not produce;
    (3) computing time, cost, or software-performance reports.
5. **Capture all layers** -- do **not** skip intermediate hypotheses, objectives, or
    analysis steps the paper explicitly discusses.
6. **Stay faithful** to the paper's wording for technical terms; paraphrase only for brevity.
7. **No outside invention** -- derive every node from the paper alone. If information is
    missing, mark the node with [uncertain].

### Phase 2 -- Annotation Rules
1. **Assign stable IDs.** Leaves receive `Lk` numbered top-to-bottom in tree order
    (`L1`, `L2`, ...). Intermediate nodes receive `Ik` (`I1`, `I2`, ...). The root node
    receives `id = "root"`.
2. **Tag `cost_class` for each leaf.** Choose `"heavy"` if the leaf describes a first-
    principles calculation (DFT, GW/BSE, AIMD, TDDFT), large-scale classical MD (~10^4
    atoms or longer than ~1 ns), Monte Carlo with non-trivial sampling, or ML-potential
    training -- anything taking more than ~1 hour on a single GPU/CPU node. Choose
    `"light"` otherwise: post-processing, analytical fits, geometry construction,
    structure mapping, visualization, or 1D ODE/PDE solving.
3. **Write `consumes` for each leaf.** List the IDs of upstream leaves whose outputs this
    leaf needs, e.g. `["L1", "L3"]`. Empty list `[]` is allowed for foundational leaves
    with no upstream. Dependencies may cross intermediate branches; do not invent
    dependencies that are not stated or implied in the paper.
4. **Stay acyclic and faithful.** The graph must be acyclic; a `consumes` reference that
    would create a cycle is the result of a misread and must be corrected. If a cost class
    or dependency is genuinely unclear, append `[uncertain]` to the relevant field.

Strictly output the annotated tree in a JSON format. Extraction fields come first; the
three annotation fields are appended within the same node object:
```

```
{ "paper": { "title": "", "authors": [], "venue": "", "year": "" },
  "problem_tree": {
    "node": "Root: broadest research problem tackled by the paper",
    "type": "root node",
    "description": "a detailed description of the research problem in this node",
    "evidence": "references back to the original paper",
    "id": "root",
    "children": [
      {"node": "Intermediate sub-problem / objective 1",
        "type": "depth-1 node",
        "description": "a detailed description of the research problem in this node",
        "evidence": "references back to the original paper",
        "id": "I1",
        "children": [
          {"type": "leaf node",
            "task": "Concrete experimental task (as phrased by paper)",
            "dataset": ["..."],
            "model_or_method": ["..."],
            "metrics": ["..."],
            "protocol_or_setup": "key settings/splits/hyperparams if stated",
            "evidence": {"figure": "Fig. X", "table": "Table Y", "section": "Sec. Z"},
            "conclusion": "explicit, detailed conclusions from this leaf's experiment",
            "status": "",
            "id": "L1",
            "cost_class": "heavy",
            "consumes": []
          }
        ]
      }
    ]
  }
}
```

E.2 TIME NOTICE

In a single-task run, MatClaw appends the notice below to the task and to the observation returned after every step, with the remaining time updated; the example is taken with 1,008 seconds left. Codex CLI receives a similar notice.

```
[Time budget] Total task wall-clock budget: 3600 seconds (60 minutes),
including API requests, reasoning, tool execution and waiting. Remaining at
this observation: 1008 seconds (16m 48s). Complete the required submission
files and call final_answer before the deadline.
```

E.3 ONE TASK UNDER REDUCED AND FULL GUIDANCE

The two prompts below are the reduced (SR) and full (SF) forms of task 202603_020, the mode-resolved phonon transmission through a Si/Ge interface. The reduced prompt keeps the input file, the required accuracy, the definitions of the reported quantities and the output format. The full prompt adds the setting of the problem, the contents of the input file, and the method: the atomistic Green's function approach, the criterion for a propagating channel, and the conventions of the dynamical matrix and of the normalised scattering matrix.

Reduced guidance (SR)

```
# Task: mode-resolved phonon transmission through a Si/Ge interface from harmonic force
↪  constants

Compute the harmonic phonon transmission of a planar Si/Ge(100) interface
between a semi-infinite Si lead on the left and a semi-infinite Ge lead on
the right.

`device_force_constants.hdf5` contains the device's principal-layer
force-constant blocks and atomic masses. Its attributes specify units,
block ordering, coupling directions, layer period and transverse periodicity.
Use these blocks as given, without symmetrisation or sum-rule corrections.
No molecular-dynamics or electronic-structure calculation is required.
Python with numpy, scipy and h5py is available.

## Quantities and accuracy
```

```
Let $t$ be the flux-normalized Si-to-Ge transmission matrix, with incoming
propagating Si channels as rows and outgoing propagating Ge channels as columns.
For every incoming channel, transmission plus reflection must sum to one
with absolute error below $10^{-6}$.

At ordinary frequencies $\nu = 1.00, 2.00, 4.00, 5.05, 6.00, 8.00,
10.00, 11.50, 12.00$ THz, report:

- `n_Si`, `n_Ge`: the number of propagating channels in each lead travelling
  towards the interface;
- `Xi`: the sum of transmission probabilities over all incoming Si channels;
- `mean_T_Si`: the mean transmission probability over all incoming Si channels;
- `std_tau_Si`: the population standard deviation of the full spectrum of
  $t t^\dagger$ (all $n_{\mathrm{Si}}$ eigenvalues, including zeros).

## Output format

Write a single `answer.json` in the workspace root with exactly this layout:
one entry per requested frequency, in the order listed above, the frequency
echoed as given (counts are integers; the other three fields are dimensionless
floats):

```json
{
  "results": [
    {
      "frequency_THz": 1.00,
      "n_Si": <int>,
      "n_Ge": <int>,
      "Xi": <float>,
      "mean_T_Si": <float>,
      "std_tau_Si": <float>
    },
    ...
  ]
}
```

All values must be computed from the staged file; do not guess or hardcode.
```

Full guidance (SF)

```
# Task: mode-resolved phonon transmission through a Si/Ge interface from harmonic force
↪  constants

A planar (100) interface between crystalline Si and crystalline Ge is studied
as a phonon scatterer. The interface sits in a device made of a semi-infinite
Si lead on the left, a semi-infinite Ge lead on the right and a central region
that contains the junction. All three parts are described by harmonic force
constants in a principal-layer basis. The question is how many propagating
phonon channels each lead carries towards the interface at a given frequency,
how much of the incident flux the interface transmits, and how unevenly the
transmission is distributed over the channels of the Si side.

## What you are given

- `device_force_constants.hdf5` -- the harmonic force-constant blocks of the
  device and the per-atom masses: for each lead, the on-site block of one
  principal layer and the coupling block to the next layer; for the central
  region, the on-site block of each of its slices, the couplings between
  consecutive slices, and the couplings of the first and last slice to their
  leads. Units, the atom-major ordering inside the blocks and the direction
  convention are recorded in the file's attributes. For both leads the coupling
  block connects a principal layer to the next one along $+x$, the direction
  from the Si lead towards the Ge lead, and in every coupling block the row
  block precedes the column block along $+x$. Both leads share the one layer
  period recorded in the file. The blocks already contain the full transverse
  periodic cell of the device, so no transverse wave-vector sampling is
  needed. Use the blocks as given, without further symmetrisation or sum-rule
  corrections.

No molecular-dynamics or electronic-structure calculation is required. The
environment provides Python with numpy, scipy and h5py. Solving one frequency
takes of the order of a minute on one core; the frequencies are independent.

## Method

Use the atomistic Green's function method for a lead-device-lead geometry with
the leads treated as semi-infinite, extended to a mode-resolved scattering
matrix: at each frequency, the propagating Bloch channels of each lead are
obtained from the lead's own blocks, the scattering matrix between the incoming
```

````
and outgoing propagating channels of the two leads is formed from the device
Green's function, and each entry is normalised by the channel group velocities
so that its squared modulus is a probability. A lead channel counts as
propagating when its Bloch factor across one layer period has unit modulus
(real wave vector); evanescent channels are removed before the scattering
matrix is formed. At the requested frequencies no channel lies at a band edge,
so the propagating and evanescent sets are well separated.

Conventions (fixed for this task):

- Frequencies are ordinary frequencies $\nu$ in THz; the dynamical problem is
  posed in the angular frequency $\omega = 2\pi\nu$.
- The dynamical blocks are the mass-weighted force constants
  $\Phi_{ij}/\sqrt{m_i m_j}$, converted to SI angular-frequency-squared units.
- The small positive imaginary part added to $\omega^2$ for the retarded
  Green's functions must be chosen small enough that, for every incoming
  channel, transmission plus reflection sums to one to better than $10^{-6}$.
- Let $t$ be the normalised transmission block of the scattering matrix from
  the incoming propagating channels of the Si lead (rows) to the outgoing
  propagating channels of the Ge lead (columns). For an incoming Si channel
  $l$, its transmission is $\Gamma_l = \sum_m |t_{lm}|^2$, and the **total
  transmission** is $\Xi(\nu) = \sum_l \Gamma_l$ over all incoming propagating
  channels of the Si lead.
- The **transmission eigenvalues** $\tau_k$ are the eigenvalues of the Hermitian
  matrix $t\,t^\dagger$; there are as many as incoming Si channels, and their
  sum equals $\Xi$. Their standard deviation is the population one (the
  variance normalised by the number of channels). Unlike the individual
  $\Gamma_l$, the $\tau_k$ do not depend on how a degenerate multiplet of
  channels is resolved into a basis.

## What to determine

At each of the nine frequencies $\nu \in \{1.00, 2.00, 4.00, 5.05, 6.00, 8.00,
10.00, 11.50, 12.00\}$ THz:

(1) $n_{\mathrm{Si}}$ and $n_{\mathrm{Ge}}$, the number of propagating channels
    of each lead that travel towards the interface;

(2) $\Xi$, the total transmission from the Si side;

(3) the mean of $\Gamma_l$ over the incoming Si channels, and the standard
    deviation of the transmission eigenvalues $\tau_k$.

## Output format

Write a single `answer.json` in the workspace root with exactly this layout:
one entry per requested frequency, in the order listed above, the frequency
echoed as given (counts are integers; the other three fields are dimensionless
floats):

```json
{
  "results": [
    {
      "frequency_THz": 1.00,
      "n_Si": <int>,
      "n_Ge": <int>,
      "Xi": <float>,
      "mean_T_Si": <float>,
      "std_tau_Si": <float>
    },
    ...
  ]
}
```

- `n_Si`, `n_Ge`: item (1).
- `Xi`: item (2).
- `mean_T_Si`, `std_tau_Si`: item (3).

All values must be computed from the staged file; do not guess or hardcode.
````