
PhysMent: An Interactive Approach For LLM Reasoning In Physics Problems

Joseph Chan ^{†1} Utkarsh Jha ^{†1} Xiyin Yang ^{†1} Abhinav Jarajapu ¹ Anik Sahai ¹ Eddie Hu ¹
Robin Jeshua Deepak ¹ Stefano Saravalle ^{◦1} Aditya Shah ^{◦1}

Abstract

Large language models (LLMs) perform strongly on static science benchmarks, yet their ability to reason about the physical world through *active experimentation* remains poorly understood. We introduce **PhysMent**, a benchmark that evaluates LLM physical reasoning via iterative, tool-mediated interaction with a MuJoCo physics simulator. Unlike static benchmarks that supply all quantities upfront, PhysMent requires models to *discover* information by applying forces, querying object states, advancing time, and modifying scene geometry before answering. The benchmark comprises 105 scenes of classical mechanics, organized across four difficulty regimes (Easy/Hard \times Single/Multi), three scene modalities (standard, object creation, hidden objects), and a scene-manipulation category, evaluated with a six-dimensional scoring framework. Results show that current models perform reasonably well on qualitative single-concept tasks (up to 80% accuracy) but degrade substantially on quantitative tasks that demand precise, multi-step experimental procedures: most models fall below 30% on the hardest single-concept category, where the bottleneck is procedural (adaptive multi-step tool use) rather than conceptual load. Across the seven models, accuracy ranges from 25% to 67%, with failures due to premature answer submission, inefficient exploration, and inconsistent grounding in simulator feedback rather than conceptual gaps. Our code is available [here](#).

[†]Equal contribution [◦]Equal senior authorship
¹AlgoVerse AI Research. Correspondence to:
Joseph Chan <josephc3610@berkeley.edu>, Utkarsh
Jha <utkarsh.jha.inbox@gmail.com>, Xiyin Yang
<xy553@cornell.edu>.

The Third AI for MATH Workshop at the 43rd International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

1. Introduction

Large language models have reached strong performance on reading comprehension (Hendrycks et al., 2021a), graduate-level science (Rein et al., 2024), and mathematical problem solving (Hendrycks et al., 2021b). These capabilities are, however, fundamentally declarative: the model reads a fully specified problem and produces a text answer. Physical reasoning in the real world is different. A scientist confronting an unknown system must design experiments, apply interventions, observe outcomes, revise hypotheses, and iterate.

This distinction is what existing benchmarks miss. PIQA (Bisk et al., 2020), ScienceQA (Lu et al., 2022), GPQA (Rein et al., 2024), and ARC (Clark et al., 2018) all test static recall of physical principles expressed in natural language. PHYRE (Bakhtin et al., 2019) and IntPhys (Riochet et al., 2020) involve physical environments but evaluate learned visual or RL policies, not language-mediated experimental reasoning. The closest prior work, Mind’s Eye (Liu et al., 2023), augments LLMs with a MuJoCo simulation backend, but uses it as a *one-shot oracle*: the model invokes the simulator once to retrieve a fact, rather than designing a multi-step experimental protocol. PhysMent addresses this gap directly: it requires a *sustained, multi-turn experimental dialogue* where the model adaptively chooses interventions based on prior observations, contrasting fundamentally with Mind’s Eye’s single invocation.

We argue that a benchmark targeting this capability, using LLMs as agents for experimental physical inquiry, requires three properties: (1) **information incompleteness**: the model must choose which quantities to measure rather than reading them from the prompt; (2) **causal interventionism**: the model must apply forces, modify the scene, or advance time to observe consequent state changes; and (3) **grounded evaluation**: answers are verified by the physics engine, not a human-authored key. We stress that these are functional requirements for *this* evaluation goal, not deficiencies of static benchmarks, which remain well suited to their own goal of measuring declarative physics knowledge.

We present **PhysMent**, a benchmark built on these three principles. Models are placed inside MuJoCo environments (Todorov et al., 2012) and given a structured API of 26 simulation tools. Each of 105 scenes presents a task, such

as identifying the hollow sphere from its rolling behavior, or computing a coefficient of kinetic friction - neither of which can be solved by applying a formula alone.

Contributions.

- PhysMent is, to our knowledge, the first benchmark to evaluate LLMs through sustained multi-turn experimental dialogue with a 3D rigid-body simulator, in contrast to single-shot oracle invocations (Mind’s Eye) or non-LLM puzzle-solving agents (PHYRE).
- We design 105 scenes across 13 difficulty-modality categories covering the core topics of undergraduate classical mechanics.
- We define a six-dimensional scoring framework that goes beyond binary accuracy to diagnose the quality of experimental strategies.
- We establish baseline results across frontier models and multiple prompting strategies, exposing systematic failure modes in quantitative and multi-step physical reasoning.

2. Related Work

We situate PhysMent against four lines of work, each related to one facet of our contribution: physics/science reasoning benchmarks (what is measured), agentic tool use (how the model acts), LLM simulator coupling (how physics is grounded), and embodied AI (where prior interactive evaluation lives). PhysMent is distinguished by combining all four strands of assessment into a single benchmark suite. A full comparison of PhysMent to representative benchmarks can be found in Table 4 in Appendix A.

2.1. Physics and Science Reasoning Benchmarks

Static text-based benchmarks. PIQA (Bisk et al., 2020) tests physical commonsense with $\sim 20k$ binary-choice questions; every problem is self-contained and requires no interaction. ScienceQA (Lu et al., 2022) and ARC (Clark et al., 2018) cover elementary-to-high-school science in multiple-choice format. GPQA (Rein et al., 2024) extends the difficulty to expert-written, graduate-level questions in physics, biology, and chemistry that are resistant to lookup-based cheating, yet still present fully-specified problems requiring no experimental intervention. MATH (Hendrycks et al., 2021b) and MMLU (Hendrycks et al., 2021a) evaluate competition mathematics and broad multitask science knowledge respectively. A recent wave of physics-specific benchmarks sharpens this paradigm. PhysReason (Zhang et al., 2025a) introduces step-level evaluation across physics theorem application, process understanding, calculation,

and condition analysis; PHYBench (Qiu et al., 2025) curates 500 problems and proposes a graded Expression Edit Distance score in place of binary correctness; and UGPhysics (Xu et al., 2025) and ABench-Physics (Zhang et al., 2025b) target contamination-resistant undergraduate physics. All of these remain static: a fully specified problem is presented and the model must derive the answer. None of these benchmarks tests the ability to *design* measurements or reason from *observed* experimental outcomes.

Vision-based physical understanding. PhysBench (Chow et al., 2025) evaluates vision-language models on physical world understanding using over 10,000 video- and image-based entries, finding significant gaps across 75 VLMs. It evaluates *perception* of pre-recorded events, not active interaction with a live simulator. IntPhys (Riochet et al., 2020) uses a violation-of-expectation paradigm from developmental psychology to test intuitive physics from rendered video; CoPhy (Baradel et al., 2020) targets counterfactual prediction of physical dynamics from visual input. Both evaluate passive prediction of pre-recorded physics — they do not permit agents to intervene in, modify, or query a live simulation.

Interactive physical environments. PHYRE (Bakhtin et al., 2019) is a 2D classical mechanics puzzle benchmark where a learned agent places objects to satisfy a goal. While interactive, it employs simplified 2D physics, does not evaluate LLMs, and does not require language-mediated reasoning about physical quantities. ScienceWorld (Wang et al., 2022) tests agents on elementary science tasks in a text-based interactive environment but uses a rule-based text simulator rather than a physics engine.

Mind’s Eye and UTOPIA. Mind’s Eye (Liu et al., 2023) is the most closely related prior work. It grounds LLM reasoning in physics by generating simulation code from text, executing it in MuJoCo, and feeding the output back as additional context. It introduces UTOPIA, a 39-subtask benchmark across 6 physical scenes (motion, friction, free fall, horizontal projectile, elastic collision, and inclined plane), where each subtask is a fixed observed/queried-concept pair. As established in Section 1, PhysMent differs from Mind’s Eye in this fundamental respect: Mind’s Eye invokes the simulator *once* as an oracle to retrieve a fact, whereas PhysMent requires a sustained multi-turn protocol in which interventions are chosen adaptively. A single-invocation oracle has no notion of iteration budget, exploration efficiency, or grounding across turns. In PhysMent, we subsume UTOPIA’s physical scope while adding $2.7\times$ more scenes, object-creation and hidden-object modalities absent in UTOPIA, and a broader 26-tool interactive API (vs. code generation).

2.2. Agentic LLMs and Tool-Augmented Reasoning

PhysMent places tool-augmented reasoning inside a physics simulation loop where tools have physically grounded semantics and answers are verified against the MuJoCo simulator’s internal state. Prior work on tool-augmented language models has shown that interleaving reasoning, planning, and external tool use can substantially improve interactive problem solving. ReAct (Yao et al., 2023) interleaves verbal reasoning traces with concrete tool actions, outperforming chain-of-thought and imitation baselines by wide margins on interactive benchmarks such as ALFWorld (Shridhar et al., 2021). Toolformer (Schick et al., 2023) teaches LLMs to self-supervise API call generation, enabling zero-shot use of calculators, search engines, and calendars. HuggingGPT (Shen et al., 2023) demonstrates that a large LLM can orchestrate specialist models through a planning-selection-execution pipeline. DEPS (Wang et al., 2023) uses structured description and self-explanation for zero-shot multi-task planning in Minecraft; Voyager (Wang et al., 2024) extends this to lifelong skill acquisition.

PAL (Gao et al., 2023) offloads computation to a Python interpreter via program-aided reasoning chains. Chameleon (Lu et al., 2023) composes heterogeneous tools under LLM orchestration. Chain-of-thought prompting (Wei et al., 2022) provides the foundational technique for eliciting multi-step reasoning via exemplar demonstrations. PhysMent inherits the interleaved reason-act loop of these methods but constrains the tool space to physically grounded operations whose outputs are dictated by a simulator rather than retrieved text. As such, tool feedback in PhysMent carries verifiable physical semantics.

2.3. LLMs Coupled with Physics Simulators

Code as Policies (Liang et al., 2023) and ProgPrompt (Singh et al., 2022) show that LLMs can generate executable robot task plans interfacing with physical APIs. Inner Monologue (Huang et al., 2023) improves long-horizon robot task completion by closing the reasoning loop with natural language environment feedback. Language to Rewards (Yu et al., 2023) couples GPT-4 with MuJoCo MPC to synthesize robotic skills; Eureka (Ma et al., 2024) uses evolutionary reward-code optimization with Isaac Gym, surpassing expert-designed rewards on 83% of tasks. RT-2 (Zitkovich et al., 2023) co-fine-tunes a vision-language model on robot trajectory data for emergent physical control. SayCan (brian ichter et al., 2022) grounds LLM outputs in robotic affordance functions for long-horizon task execution.

These systems integrate LLMs with physics engines for *robotic control*. PhysMent repurposes the LLM by treating the simulator as a measurement tool, using it as a reliable source of ground truth. The focus isn’t on controlling anything, but on observing how the LLM behaves over multiple

steps as it interacts with a 26-tool API.

2.4. Embodied AI Benchmarks

EmbodiedQA (Das et al., 2018) requires navigation and question answering in 3D environments. ALFRED (Shridhar et al., 2020) evaluates language grounding in household task execution. ALFWorld (Shridhar et al., 2021) aligns text-based and visually embodied environments to study policy transfer. Habitat (Savva et al., 2019) and AI2-THOR (Kolve et al., 2022) provide high-fidelity 3D platforms for visual navigation and manipulation. BEHAVIOR (Srivastava et al., 2021) covers 100 household activities with rich physical state changes; MineDojo (Fan et al., 2022) builds an open-ended agent framework on internet-scale Minecraft knowledge. These benchmarks target general embodied acting and do not provide access to the underlying physical quantities such as forces, momenta, torques, energies that are central to PhysMent.

3. Method

3.1. Motivation

Existing benchmarks for evaluating physical reasoning in LLMs rely predominantly on answering questions based on text, and all relevant information is provided in the prompt. While such evaluations measure a model’s ability to recall and apply physics formulas, they fail to capture a fundamental aspect of scientific reasoning: the capacity to design experiments, gather evidence through interaction, and iteratively refine hypotheses based on observed outcomes. In physics problem solving, the investigator often lacks immediate access to all necessary quantities and must instead probe the system to extract them. PhysMent evaluates LLMs under these conditions, where the model must act as an agent within a physical environment, deciding what to measure, when to intervene, and how to synthesize observations into a final answer.

3.2. Pipeline Formulation

We define the PhysMent evaluation pipeline as follows. Let $\mathcal{S} = \{s_1, s_2, \dots, s_N\}$ denote a set of N physics scenes, where each scene s_i is specified by an XML configuration file $\mathcal{X}_i \in \mathcal{L}_{\text{XML}}$ that defines the physical world, and a meta-data file \mathcal{M}_i that encodes the task description τ_i , the set of observable object identifiers $\mathcal{O}_i = \{o_1, o_2, \dots, o_{K_i}\}$, a permission matrix $\mathbf{P}_i \in \{0, 1\}^{K_i \times D}$ governing which of D physical attributes (e.g., mass, position, geometry type) are accessible per object, and the ground-truth answer a_i^* .

The simulator Φ maintains a physical state $\mathbf{x}_t \in \mathbb{R}^m$ at each timestep t , where m represents the positions, velocities, and orientations of all bodies. The simulator changes according

to:

$$\mathbf{x}_{t+1} = \Phi(\mathbf{x}_t, \mathbf{u}_t) \quad (1)$$

where $\mathbf{u}_t \in \mathcal{U}$ is an action drawn from the action space \mathcal{U} . The action space comprises a discrete set of tool calls $\mathcal{T} = \{f_1, f_2, \dots, f_T\}$, where each tool f_j maps from a parameter space to either a state modification or an observation:

$$f_j : \Theta_j \rightarrow \mathcal{A} \cup \mathcal{R} \quad (2)$$

Here Θ_j denotes the parameter space of tool f_j , \mathcal{A} denotes the set of state-modifying actions, and \mathcal{R} denotes the space of returned observations. Available tools include state queries (e.g., `get_position`, `get_velocity`), state modifications (e.g., `apply_force`, `move_object`, `step`), and a terminal action `answer` that submits the model’s response.

An LLM agent π interacts with the simulator over a bounded number of iterations I_{\max} . At each iteration $k \in \{1, \dots, I_{\max}\}$, the agent receives the cumulative context \mathbf{c}_k consisting of the initial prompt (derived from τ_i , \mathcal{O}_i , and \mathbf{P}_i) and all prior tool calls and their returned results. The agent then produces a response r_k containing reasoning in natural language and a sequence of tool calls:

$$r_k = \pi(\mathbf{c}_k), \quad r_k = (\text{reasoning}_k, [f_{j_1}(\theta_1), f_{j_2}(\theta_2), \dots]) \quad (3)$$

The tool calls are executed sequentially by the simulator, and the returned observations are appended to form \mathbf{c}_{k+1} . The interaction terminates when either the agent calls the `answer` tool with its prediction \hat{a}_i , or $k = I_{\max}$ is reached. In the latter case, the episode is scored as incorrect regardless of any partial reasoning produced.

3.3. Scene Taxonomy & Design

Each scene s_i is assigned a problem type p_i drawn from the set `{comparison, calculation, computation, observation}` that determines both the expected answer format and the evaluation criteria. Comparison problems require the agent to identify which object satisfies a given physical property (e.g., which sphere is hollow). Calculation and computation problems require the agent to derive a numerical quantity from simulator interaction (e.g., coefficient of friction, angular velocity). Observation problems require the agent to describe a qualitative physical phenomenon.

The scenes cover core topics in classical mechanics including kinematics, Newton’s laws, rotational dynamics, energy conservation, momentum conservation, oscillatory motion, and friction. Objects in each scene are parameterized by physical attributes such as mass $m \in \mathbb{R}^+$, position $\mathbf{p} \in \mathbb{R}^3$, orientation $\mathbf{q} \in \mathbb{H}$ (unit quaternions), geometry type, density $\rho \in \mathbb{R}^+$, and friction coefficients $\boldsymbol{\mu} \in \mathbb{R}^3$. Table 1 summarizes the full scene taxonomy across 105 scenes.

To make the taxonomy reproducible, we define each axis by an explicit construction rule applied at design time. **Single vs. Multi** is the number of distinct physical concepts whose correct application is required to reach the answer: *Single* scenes require exactly one concept (e.g., projectile kinematics), while *Multi* scenes require composing ≥ 2 concepts (e.g., friction followed by energy conservation). **Easy vs. Hard** is determined by the minimum number of distinct tool-call *types* and simulator interventions a reference solution requires: *Easy* scenes are solvable with a short fixed measurement sequence (≤ 3 intervention steps, e.g., `displace-step-read`), whereas *Hard* scenes require an experimental loop in which a later action depends on an earlier observation (> 3 steps, e.g., a binary search over incline angle to find the sliding threshold). **Object Creation** scenes require the agent to instantiate at least one new body via `create_objects` to construct a controlled comparison (e.g., adding a reference mass), so the answer is unreachable from the initial scene alone. **Hidden Objects** scenes contain at least one task-relevant body that is not exposed in the initial object list or is visually/parametrically masked, and must be revealed through `find_objects` before it can be measured. Per-axis definitions, a worked example for every cell, and the reference solution length used to assign Easy/Hard are provided in Appendix C, Table 6.

We note that *Hard Multi* scenes are not uniformly harder in measured accuracy than *Easy Multi* (Section 4). This is by design and reflects the two orthogonal axes above: *Easy/Hard* grades the length and adaptivity of the required *experimental procedure*, while *Single/Multi* grades *conceptual* load. We make this decoupling explicit because it is itself a finding: procedural difficulty and conceptual difficulty are distinct failure axes for current models (Section 5).

All scenes run under MuJoCo 3.3.3 (Todorov et al., 2012) with a fixed timestep of 0.005 s. We adopt MuJoCo for four reasons that bear directly on benchmark validity: deterministic rigid-body dynamics (reproducible ground truth), programmatic access to internal state (engine-verifiable answers), fixed-timestep control (well-defined `step` tool semantics), and direct comparability with Mind’s Eye (Liu et al., 2023). Detailed information regarding the 26 tools in PhysMent can be found in Table 7 in Appendix D.

3.4. Evaluation

Given a predicted answer \hat{a}_i and the ground-truth a_i^* , a response is marked correct if one of the following holds: (1) both answers are numerical and lie within a mixed absolute–relative tolerance, or (2) the predicted answer is contained

Table 1. PhysMent scene taxonomy (105 scenes). Scene IDs and folder names are **consecutive** 7–111 (Scenes/Scene{N}/scene{N}.json).

Scene ID	Category	N	Primary Topics
7–15	Easy Single Physics	9	Kinematics, free-fall, inclined planes
16–20	Easy Single + Object Creation	5	Same + object manipulation
21–25	Easy Single + Hidden Objects	5	Same + object discovery
26–40	Hard Single Physics	15	Ladders, equilibrium, torque, angular momentum
41–45	Hard Single + Object Creation	5	Hard + object manipulation
46–50	Hard Single + Hidden Objects	5	Hard + object discovery
51–65	Easy Multi-Physics	15	Combined concept problems
66–70	Easy Multi + Object Creation	5	Multi-concept + object creation
71–75	Easy Multi + Hidden Objects	5	Multi-concept + discovery
76–90	Hard Multi-Physics	15	Complex multi-concept
91–95	Hard Multi + Object Creation	5	Complex + object creation
96–100	Hard Multi + Hidden Objects	5	Complex + discovery
101–111	Create / Delete / Attach	11	Dynamic scene manipulation

in the ground-truth. Formally:

$$C(s_i) = \begin{cases} 1 & \text{if } |\hat{a}_i - a_i^*| \leq \epsilon_{\text{abs}} + \epsilon_{\text{rel}} |a_i^*| \\ 1 & \text{if } \hat{a}_i \subseteq a_i^* \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

We set $\epsilon_{\text{rel}} = 0.05$ (5% relative tolerance) to accommodate floating-point accumulation across multi-step tool calls, together with a small absolute floor ϵ_{abs} (in the units of the queried quantity) so that the criterion remains well defined when $a_i^* = 0$; pure relative tolerance would otherwise divide by zero for ground-truth-zero answers (e.g., a net force of 0 N at equilibrium). This mixed form follows the standard `isclose` convention (Barker, 2015). For categorical and comparison answers, exact string matching applies after normalization to lowercase (see Section 5.4 for the full answer-normalization protocol). The overall accuracy for an agent π across all N scenes is:

$$\text{Acc}(\pi) = \frac{1}{N} \sum_{i=1}^N C(s_i) \quad (5)$$

3.5. Scoring Framework

Beyond binary correctness, each scene yields a six-dimensional score normalized to $[0, 1]$ and combined into a weighted Final Score: Let $\mathbf{w} = (w_C, w_E, w_G, w_V, w_R, w_{G'})$ with $\sum_j w_j = 1$. Then:

$$\text{FS}_i = 100 \mathbf{w}^\top (C, E, G, V, R, G')^\top \quad (6)$$

The dimensions are defined as follows. **Correctness** (C) is the per-scene correctness from Eq. (4) (binary at the strict threshold; see Section 5.4 for the graded variant). **Efficiency** (E) is the budget utilization $1 - k/I_{\text{max}}$, where k is the iteration on which `answer` is called. We report E as a descriptive utilization measure rather than a quality signal: because it rewards early submission, it can in principle penalize an

agent that spends additional iterations verifying a correct answer, a tension we discuss in Section 5 and which motivates reporting E separately rather than folding it silently into a single score. **Groundedness** (G) is the fraction of quantitative assertions in the model’s reasoning that are traceable to a value returned by a prior tool call, following the faithfulness/attribution tradition in retrieval-augmented and agentic evaluation (Es et al., 2024). **Action Validity** (V) is the fraction of tool calls returning valid (non-error) results. **Reasoning Diversity** (R) is $1 -$ maximum bigram repetition across the reasoning text; this is a deliberately lightweight *lexical proxy* in the spirit of *distinct-n* (Li et al., 2016), intended to flag agents that loop on identical tool-call phrasings. **Generalization** (G') is held-out category accuracy. We set $\mathbf{w} = (0.4, 0.15, 0.15, 0.10, 0.10, 0.10)$ for $(w_C, w_E, w_G, w_V, w_R, w_{G'})$ respectively, weighting correctness the heaviest. Full per-dimension results are reported in Appendix B.

3.6. Design Hypotheses and Experimental Conditions

We believe that model performance will strongly correlate with the problem complexity. Specifically, we expect models to perform well on comparison and observation tasks that primarily require qualitative reasoning, such as identifying which object is heavier or which sphere is hollow based on rolling behavior. On the other hand, we expect significantly lower performance on calculation and computation tasks that require calculations, multi-step tool usage, and the correct application of physical formulas. Furthermore, we predict that the bounded iteration count I_{max} will be a constraint, as models that fail to plan their tool usage efficiently are likely to use up their iterations before submitting an answer, resulting in a failure mode that is different from incorrect reasoning. We also want to see the effect of iteration budget, so we will evaluate each agent across a range of iteration counts $I_{\text{max}} \in \{5, 10, 15, 20\}$ and report accuracy as a function of I_{max} , identifying the point at which additional

iterations yield diminishing returns. The agent is explicitly told its I_{\max} in the prompt, so planning is part of the task rather than a hidden constraint. The four budgets were chosen to bracket the regime of interest: $I_{\max} = 5$ is near the minimum that allows a measure-intervene-remeasure loop for our Easy scenes, $I_{\max} = 20$ is where added iterations stop changing accuracy for every model we tested (Section 4), and the intermediate values resolve the shape of the curve in between. The design of PhysMent is that it is a diagnostic tool that can help identify different failure reasons in physical reasoning.

4. Results

4.1. Model Performance

Table 2 shows the overall accuracy across all seven models and four iteration budgets. GPT-5.5 achieves its highest accuracy at 51.4% under $I_{\max} = 5$, while Gemini 3.1 Pro reaches its peak of 66.7% at $I_{\max} = 20$. DeepSeek R1 and GLM-5 fluctuate within the roughly 38–53% range across budgets. Gemini 3.1 Pro shows the largest improvement with iterations, rising from 29.5% to 66.7%, with Kimi K-2.5 close behind (27.4% to 56.2%). Qwen 3.5 consistently scores lowest, remaining between 24.8% and 35.2% across all budgets.

Figure 1 illustrates these different trajectories across the four major category groups, showing that the marginal decline pattern for GPT-5.5 and Claude Opus 4.7 is most noticeable in Easy Single and Easy Multi scenes.

Table 3 shows per-model metrics averaged across iteration budgets. GPT-5.5 has the highest accuracy (50.0%), while GLM-5 achieves the highest groundedness score (0.744) despite ranking fourth in accuracy. This shows that GLM-5 grounds its reasoning in simulator observations more consistently than it converts that grounding into correct answers. DeepSeek R1 shows the opposite pattern: high accuracy (48.5%) with the lowest groundedness (0.514), indicating heavier reliance on prior knowledge rather than obtaining results from the simulator.

Figure 2 breaks down accuracy by scene category and presents a heatmap. Easy Single Physics scenes are the strongest category overall, with GPT-5.5 reaching 80.6% and Gemini 3.1 Pro reaching 72.2%. Hard Single Physics is the hardest category despite involving only one physical concept, with most models scoring below 40%, which shows that precision in multi-step tool usage is the bottleneck rather than conceptual complexity. Easy Multi + Hidden scenes surprisingly outperform their non-hidden counterparts, with Claude Opus 4.7 reaching 30.0% and Gemini 80.0%, indicating that the `find_objects` tool is reliably invoked once models recognize it is needed.

5. Discussion

5.1. The Effect of Iteration Budget

Contrary to the expectation that more iterations always help, our results show three distinct patterns across models. First, some models improve steadily with more iterations: Gemini 3.1 Pro more than doubled its accuracy (29.5% \rightarrow 66.7%), and Kimi K-2.5 similarly doubled (27.4% \rightarrow 56.2%). Second, some models showed a gradual decrease in accuracy: GPT-5.5 dropped slightly from 51.4% at $I_{\max} = 5$ to 48.6% at $I_{\max} = 15$, and Claude Opus 4.7 similarly remained flat with a slight downward trend across budgets. Third, some models neither improved nor declined monotonically: DeepSeek R1 stayed in the upper-40s to low-50s, and GLM-5 fluctuated between 37.7% and 48.6% without a clear trend in iteration count.

This suggests that additional iterations are not always necessarily beneficial. For stronger models that already answer correctly in a few steps, extra iterations introduce opportunities to second guess, experiment an unnecessary number of times, or accumulate context that leads to worse reasoning. For weaker models, additional iterations provide more chances to explore and converge. The implication is that I_{\max} should be chosen per model rather than fixed across the benchmark.

5.2. Qualitative vs. Quantitative Reasoning

Consistent with our hypothesis in Section 3.6, the models performed significantly better on comparison and observation tasks compared to calculation and computation tasks.

Comparison problems, which ask which object satisfies a physical property, often demand only a single well-chosen experiment followed by qualitative interpretation. The models are able to do this even when the maximum iteration I_{\max} is low.

Calculation and computation tasks, by contrast, require precise numerical reasoning and multiple tool invocations. Some common failure modes included applying the wrong physics formula, and using incorrect values extracted from the simulator. Sometimes, the LLMs even submitted intermediate steps as their answer instead of the actual final answers.

5.3. Common Failure Modes

We identified six recurring failure modes: premature answer submission, insufficient simulation time, answer format mismatch, object ID confusion, wrong parameter substitution, and simulation trust vs. theoretical reasoning. Figure 3 in Appendix F illustrates the first of these on Scene 48, where the agent terminates after 2 of 5 available iterations by reading the incline angle off static geometry instead of ex-

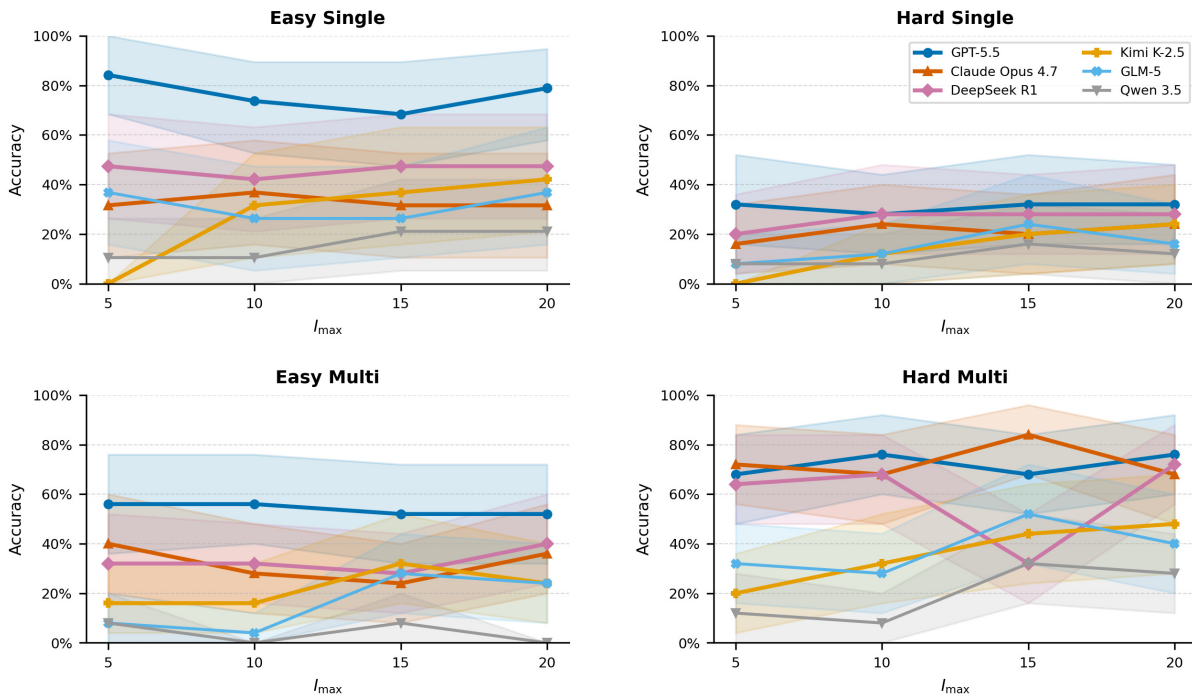


Figure 1. Accuracy vs. iteration budget I_{\max} across the four major scene categories. Models show three distinct patterns: steady improvement (Gemini 3.1 Pro, Kimi K-2.5), marginal decline with more iterations (GPT-5.5, Claude Opus 4.7), and plateauing (GLM-5, DeepSeek R1, Qwen 3.5).

Table 2. Overall accuracy (%) of each model on PhysMent across iteration budgets.

I_{\max}	Gemini 3.1 Pro	GPT-5.5	Claude Opus 4.7	Kimi K-2.5	GLM-5	DeepSeek R1	Qwen 3.5
5	29.5%	51.4%	40.5%	27.4%	46.7%	42.3%	26.7%
10	50.4%	49.5%	39.6%	37.1%	37.7%	51.4%	24.8%
15	51.4%	48.6%	42.3%	46.7%	48.6%	48.6%	35.2%
20	66.7%	50.5%	40.5%	56.2%	48.6%	53.3%	27.6%

perimentally locating the sliding threshold. Full interaction traces for each failure mode are provided in Appendix F.

5.4. Answer Normalization in Scoring

A recurring challenge in free-form benchmark evaluation is that an automatic matcher can mark a physically correct answer as wrong purely because of surface form. Similar free-form agentic benchmarks that suffer from this issue (Hendrycks et al., 2021b; Mialon et al., 2024; Chandak et al., 2025) address it with equivalence-aware classification. PhysMent’s default matcher is intentionally *strict*: it applies the numerical tolerance of Eq. (4) and otherwise requires normalized substring containment. To separate genuine reasoning failures from matcher false negatives, we additionally define an **answer-normalization protocol** consisting of four documented, rule-based equivalence classes. A model response is regraded from Fail to Pass if and only

if it falls into one of the following pre-registered classes verified against the scene’s ground truth:

- (1) **Equivalent object notation:** correct object under an alias the matcher missed (e.g. "2" vs "object_2"). (15 cases, 5 scenes.)
- (2) **Trivial rounding:** correct formula, off by rounding beyond the tolerance of Eq. (4). (35 cases, 10 scenes.)
- (3) **Conceptual description:** correct physical description equivalent to the keyword ground truth (e.g. “slows, stops, and reverses” for elastic collision). (63 cases, 11 scenes.)
- (4) **Partial answer format:** correct quantity embedded in extra or non-canonical formatting. (37 cases, 7 scenes.)

These rules account for 150 re-grades over 33 unique scenes. Two scenes dominate the conceptual class (Scenes 17 and 45 contribute 36 of 63), and Scene 52 contributes 17 of 37 partial-format cases.

Category	GPT-5.5	Gemini	Opus	DSR1	Qwen	Kimi	GLM
Easy Single Phys.	80.6	72.2	25.0	66.7	33.3	50.0	61.1
Easy Single + Obj.C.	50.0	55.0	45.0	40.0	5.0	20.0	25.0
Easy Single + Hid.	60.0	50.0	25.0	45.0	20.0	30.0	45.0
Hard Single Phys.	36.7	15.0	25.0	6.7	1.7	5.0	1.7
Hard Single + Obj.C.	0.0	50.0	0.0	25.0	30.0	30.0	20.0
Hard Single + Hid.	40.0	40.0	30.0	40.0	0.0	20.0	35.0
Easy Multi	41.7	45.0	21.7	31.7	5.0	30.0	25.0
Easy Multi + Obj.C.	40.0	50.0	65.0	25.0	10.0	35.0	35.0
Easy Multi + Hid.	85.0	80.0	30.0	60.0	25.0	55.0	50.0
Hard Multi	66.7	58.3	71.7	31.7	23.3	40.0	46.7
Hard Multi + Obj.C.	80.0	70.0	50.0	25.0	10.0	25.0	30.0
Hard Multi + Hid.	60.0	50.0	100.0	25.0	5.0	15.0	25.0
Create / Del. / Att.	18.2	56.8	38.6	27.3	13.6	25.0	18.2

< 20
20–39
40–59
60–79
≥ 80

Figure 2. Per-category accuracy heatmap across models, averaged over iteration budgets. Cell colors indicate accuracy buckets: red for < 20%, yellow for 20–39%, light green for 40–59%, mid green for 60–79%, and dark green for ≥ 80%. Several modality cells contain only 5 scenes (Table 1), so a single scene shifts that cell’s accuracy by roughly 20 percentage points; per-cell values should be read as coarse indicators rather than precise estimates.

Table 3. Per-model metrics averaged across available iteration budgets. Accuracy is the fraction of scenes answered correctly. Ground. = Grounded-ness Score, Act. Val. = Action Validity Score. All models cover $I_{\max} \in \{5, 10, 15, 20\}$. Full per-dimension breakdown in Appendix B.

Model	Acc. (%)	Avg. FS	Ground.	Act. Val.
Claude Opus 4.7	40.7	60.0	0.544	0.999
Gemini 3.1 Pro	49.5	58.4	0.598	1.000
GPT-5.5	50.0	64.1	0.729	1.000
DeepSeek R1	48.5	55.4	0.514	1.000
GLM-5	45.2	50.0	0.744	1.000
Kimi K-2.5	41.9	46.7	0.702	0.999
Qwen 3.5	28.6	44.9	0.712	1.000

5.5. Limitations

PhysMent has several limitations. The benchmark covers only classical mechanics and does not cover other physics topics. It is purely language-based and does not test visual perception of physical scenes. The iteration budget I_{\max} imposes an artificial ceiling on experimental complexity. Our strict default matcher could be made more flexible through fuzzy matching or semantic-equivalence checking. Finally, our answer normalization protocol can be improved with additional equivalence classes.

6. Future Work

Vision and embodiment. Because models frequently abandoned simulation after minimal tool calls (Section 5), a key open question is whether rendered RGB(D) observations alongside state readouts would improve temporal reasoning. Extending PhysMent with visual observations and vision language model interfaces would test perceptual grounding alongside tool-based reasoning.

Physics coverage and fidelity. Since all failure modes occurred within classical mechanics, it remains unknown whether the same patterns hold for fluids, deformation, or thermodynamics. Future versions could incorporate additional simulation substrates and calibration tasks closer to laboratory practice.

Models, budgets, and cost. Given that models such as GPT-5.5 and Claude Opus 4.7 showed marginal decline with more iterations while weaker models improved, adaptive stopping policies and think-mode ablations remain important open directions, as does standardizing the relationship between cost, latency, and accuracy reports across providers.

Dataset hygiene and community extensions. The 33 normalized scenes highlight the need for stricter answer format validation at construction time. Curating contributor guidelines and supporting community submitted scenes with automated validity checks would improve the long term benchmark stability.

7. Conclusion

We introduce PhysMent, a benchmark that tests how well language models reason about physics by interacting with a MuJoCo simulator. Unlike static benchmarks, PhysMent requires models to actively discover information through experimentation. Across 105 scenes and seven frontier models, we found that no model exceeded 67% overall accuracy, despite category-level peaks above 80%, and that increasing the iteration budget did not reliably improve performance, as some models plateaued, and others degraded with more iterations. These findings point toward the need for reasoning architectures that can plan experiments adaptively and maintain grounding across long multi-turn interactions, rather than simply extended context windows or additional tool calls.

References

- Bakhtin, A., van der Maaten, L., Johnson, J., Gustafson, L., and Girshick, R. Phyre: A new benchmark for physical reasoning. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alch'e-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/4191ef5f6c1576762869ac49281130c9-Paper.pdf.
- Baradel, F., Neverova, N., Mille, J., Mori, G., and Wolf, C. Cophy: Counterfactual learning of physical dynamics. In *International Conference on Learning Representations*, 2020. URL <https://openreview.net/forum?id=SkeyppEFvS>.
- Barker, C. PEP 485 – a function for testing approximate equality. Python Enhancement Proposal 485, 2015. <https://peps.python.org/pep-0485/>.
- Bisk, Y., Zellers, R., Gao, J., Choi, Y., et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.
- brian ichter, Brohan, A., Chebotar, Y., Finn, C., Hausman, K., Herzog, A., Ho, D., Ibarz, J., Irpan, A., Jang, E., Julian, R., Kalashnikov, D., Levine, S., Lu, Y., Parada, C., Rao, K., Sermanet, P., Toshev, A. T., Vanhoucke, V., Xia, F., Xiao, T., Xu, P., Yan, M., Brown, N., Ahn, M., Cortes, O., Sievers, N., Tan, C., Xu, S., Reyes, D., Rettinghouse, J., Quiambao, J., Pastor, P., Luu, L., Lee, K.-H., Kuang, Y., Jesmonth, S., Jeffrey, K., Ruano, R. J., Hsu, J., Gopalakrishnan, K., David, B., Zeng, A., and Fu, C. K. Do as i can, not as i say: Grounding language in robotic affordances. In *6th Annual Conference on Robot Learning*, 2022. URL https://openreview.net/forum?id=bdHkMjBJG_w.
- Chandak, N., Goel, S., Prabhu, A., Hardt, M., and Geiping, J. Answer matching outperforms multiple choice for language model evaluation. 2025. URL <https://arxiv.org/abs/2507.02856>.
- Chow, W., Mao, J., Li, B., Seita, D., Campagnolo Guizilini, V., and Wang, Y. Physbench: Benchmarking and enhancing vision-language models for physical world understanding. In *International Conference on Learning Representations*, volume 2025, pp. 97959–98108, 2025.
- Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try arc, the ai2 reasoning challenge. 2018. URL <https://arxiv.org/abs/1803.05457>.
- Das, A., Datta, S., Gkioxari, G., Lee, S., Parikh, D., and Batra, D. Embodied question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.
- Es, S., James, J., Espinosa Anke, L., and Schockaert, S. RAGAs: Automated evaluation of retrieval augmented generation. In Aletras, N. and De Clercq, O. (eds.), *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pp. 150–158, St. Julians, Malta, March 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.eacl-demo.16. URL <https://aclanthology.org/2024.eacl-demo.16/>.
- Fan, L., Wang, G., Jiang, Y., Mandlkar, A., Yang, Y., Zhu, H., Tang, A., Huang, D.-A., Zhu, Y., and Anandkumar, A. Minedojo: Building open-ended embodied agents with internet-scale knowledge. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. URL https://openreview.net/forum?id=rc8o_j8I8PX.
- Gao, L., Madaan, A., Zhou, S., Alon, U., Liu, P., Yang, Y., Callan, J., and Neubig, G. Pal: Program-aided language models. In *International conference on machine learning*, pp. 10764–10799. PMLR, 2023.
- Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. 2021a.
- Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*,

- 2021b. URL <https://openreview.net/forum?id=7Bywt2mQsCe>.
- Huang, W., Xia, F., Xiao, T., Chan, H., Liang, J., Florence, P., Zeng, A., Tompson, J., Mordatch, I., Chebotar, Y., Sermanet, P., Jackson, T., Brown, N., Luu, L., Levine, S., Hausman, K., and Ichter, B. Inner monologue: Embodied reasoning through planning with language models. In Liu, K., Kulic, D., and Ichnowski, J. (eds.), *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 1769–1782. PMLR, 14–18 Dec 2023. URL <https://proceedings.mlr.press/v205/huang23c.html>.
- Kolve, E., Mottaghi, R., Han, W., VanderBilt, E., Weihs, L., Herrasti, A., Deitke, M., Ehsani, K., Gordon, D., Zhu, Y., Kembhavi, A., Gupta, A., and Farhadi, A. Ai2-thor: An interactive 3d environment for visual ai. 2022. URL <https://arxiv.org/abs/1712.05474>.
- Li, J., Galley, M., Brockett, C., Gao, J., and Dolan, B. A diversity-promoting objective function for neural conversation models. In Knight, K., Nenkova, A., and Rambow, O. (eds.), *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 110–119, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1014. URL <https://aclanthology.org/N16-1014/>.
- Liang, J., Huang, W., Xia, F., Xu, P., Hausman, K., Ichter, B., Florence, P., and Zeng, A. Code as policies: Language model programs for embodied control. In *2023 IEEE International conference on robotics and automation (ICRA)*, pp. 9493–9500. IEEE, 2023.
- Liu, R., Wei, J., Gu, S. S., Wu, T.-Y., Vosoughi, S., Cui, C., Zhou, D., and Dai, A. M. Mind’s eye: Grounded language model reasoning through simulation. In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=4rXMRuoJlai>.
- Lu, P., Mishra, S., Xia, T., Qiu, L., Chang, K.-W., Zhu, S.-C., Tafjord, O., Clark, P., and Kalyan, A. Learn to explain: Multimodal reasoning via thought chains for science question answering. volume 35, pp. 2507–2521, 2022.
- Lu, P., Peng, B., Cheng, H., Galley, M., Chang, K.-W., Wu, Y. N., Zhu, S.-C., and Gao, J. Chameleon: Plug-and-play compositional reasoning with large language models. volume 36, pp. 43447–43478, 2023.
- Ma, Y. J., Liang, W., Wang, G., Huang, D.-A., Bastani, O., Jayaraman, D., Zhu, Y., Fan, J., and anandkumar, a. Eureka: Human-level reward design via coding large language models. In Kim, B., Yue, Y., Chaudhuri, S., Fragkiadaki, K., Khan, M., and Sun, Y. (eds.), *International Conference on Learning Representations*, volume 2024, pp. 26516–26560, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/70c26937fbf3d4600b69a129031b66ec-Paper-Conference.pdf.
- Mialon, G., Fourier, C., Wolf, T., LeCun, Y., and Scialom, T. Gaia: a benchmark for general ai assistants. 2024: 9025–9049, 2024.
- Qiu, S., Guo, S., Song, Z.-Y., Sun, Y., Cai, Z., Wei, J., Luo, T., Yin, Y., Haoxu, Z., Hu, Y., Wang, C., Tang, C., Chang, H., Liu, Q., Zhou, Z., Zhang, T., Zhang, J., Liu, Z., Li, M., Zhang, Y., Jing, B., Yin, X., Ren, Y., Fu, Z., Ji, J., Wang, W., Tian, X., Lv, A., Man, L., Li, J., Tao, F., Sun, Q., Liang, Z., Mu, Y., Li, Z., Zhang, J.-J., Zhang, S., Li, X., Xia, X., Lin, J., Shen, Z., Chen, J., Xiong, Q., Wang, B., Wang, F., Niziyang, Zhang, B., Cui, F., shaochangkun, Cao, Q.-H., xing Luo, M., Zhang, M., and Zhu, H. X. PHYBench: Holistic evaluation of physical perception and reasoning in large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025. URL <https://openreview.net/forum?id=brG8FPqlcf>.
- Rein, D., Hou, B. L., Stickland, A. C., Petty, J., Pang, R. Y., Dirani, J., Michael, J., and Bowman, S. R. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024. URL <https://openreview.net/forum?id=Ti67584b98>.
- Riochet, R., Castro, M. Y., Bernard, M., Lerer, A., Fergus, R., Izard, V., and Dupoux, E. Intphys: A framework and benchmark for visual intuitive physics reasoning. 2020. URL <https://arxiv.org/abs/1803.07616>.
- Savva, M., Kadian, A., Maksymets, O., Zhao, Y., Wijmans, E., Jain, B., Straub, J., Liu, J., Koltun, V., Malik, J., Parikh, D., and Batra, D. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.
- Schick, T., Dwivedi-Yu, J., Dessi, R., Raileanu, R., Lomeli, M., Hambro, E., Zettlemoyer, L., Cancedda, N., and Scialom, T. Toolformer: Language models can teach themselves to use tools. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 68539–68551. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/d842425e4

- bf79ba039352da0f658a906-Paper-Conference.pdf.
- Shen, Y., Song, K., Tan, X., Li, D., Lu, W., and Zhuang, Y. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 38154–38180. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/77c33e6a367922d003ff102ffb92b658-Paper-Conference.pdf.
- Shridhar, M., Thomason, J., Gordon, D., Bisk, Y., Han, W., Mottaghi, R., Zettlemoyer, L., and Fox, D. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
- Shridhar, M., Yuan, X., Cote, M.-A., Bisk, Y., Trischler, A., and Hausknecht, M. {ALFW}orld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=0IOX0YcCdTn>.
- Singh, I., Blukis, V., Mousavian, A., Goyal, A., Xu, D., Tremblay, J., Fox, D., Thomason, J., and Garg, A. Prog-prompt: Generating situated robot task plans using large language models. In *Workshop on Language and Robotics at CoRL 2022*, 2022. URL https://openreview.net/forum?id=3K4-U_5cRw.
- Srivastava, S., Li, C., Lingelbach, M., Martín-Martín, R., Xia, F., Vainio, K., Lian, Z., Gokmen, C., Buch, S., Liu, C. K., Savarese, S., Gweon, H., Wu, J., and Fei-Fei, L. Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments, 2021. URL <https://arxiv.org/abs/2108.03332>.
- Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.
- Wang, G., Xie, Y., Jiang, Y., Mandlkar, A., Xiao, C., Zhu, Y., Fan, L., and Anandkumar, A. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL <https://openreview.net/forum?id=ehfRiF0R3a>.
- Wang, R., Jansen, P., Côté, M.-A., and Ammanabrolu, P. Scienceworld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 11279–11298, 2022.
- Wang, Z., Cai, S., Chen, G., Liu, A., Ma, X. S., and Liang, Y. Describe, explain, plan and select: Interactive planning with llms enables open-world multi-task agents. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 34153–34189. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/6b8dfb8c0c12e6fafc6c256cb08a5ca7-Paper-Conference.pdf.
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.
- Xu, X., Xu, Q., Xiao, T., Chen, T., Yan, Y., Zhang, J., Diao, S., Yang, C., and Wang, Y. UGPhysics: A comprehensive benchmark for undergraduate physics reasoning with large language models. In Singh, A., Fazel, M., Hsu, D., Lacoste-Julien, S., Berkenkamp, F., Maharaj, T., Wagstaff, K., and Zhu, J. (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 69849–69877. PMLR, 13–19 Jul 2025. URL <https://proceedings.mlr.press/v267/xu25ai.html>.
- 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.
- Yu, W., Gileadi, N., Fu, C., Kirmani, S., Lee, K.-H., Arenas, M. G., Chiang, H.-T. L., Erez, T., Hasenclever, L., Humplik, J., Ichter, B., Xiao, T., Xu, P., Zeng, A., Zhang, T., Heess, N., Sadigh, D., Tan, J., Tassa, Y., and Xia, F. Language to rewards for robotic skill synthesis. In Tan, J., Toussaint, M., and Darvish, K. (eds.), *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 374–404. PMLR, 06–09 Nov 2023. URL <https://proceedings.mlr.press/v229/yu23a.html>.
- Zhang, X., Dong, Y., Wu, Y., Huang, J., Jia, C., Fernando, B., Shou, M. Z., Zhang, L., and Liu, J. PhysReason: A comprehensive benchmark towards physics-based reasoning. In Che, W., Nabende, J., Shutova, E., and Pilehvar, M. T. (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16593–16615, Vienna, Austria, July 2025a. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.811. URL <https://aclanthology.org/2025.acl-long.811/>.

Zhang, Y., Ma, Y., Gu, Y., Yang, Z., Zhuang, Y., Wang, F., Huang, Z., Wang, Y., Huang, C., Song, B., Lin, C., and Zhao, J. Abench-physics: Benchmarking physical reasoning in llms via high-difficulty and dynamic physics problems. 2025b. URL <https://arxiv.org/abs/2507.04766>.

Zitkovich, B., Yu, T., Xu, S., Xu, P., Xiao, T., Xia, F., Wu, J., Wohlhart, P., Welker, S., Wahid, A., Vuong, Q., Vanhoucke, V., Tran, H., Soricut, R., Singh, A., Singh, J., Sermanet, P., Sanketi, P. R., Salazar, G., Ryoo, M. S., Reymann, K., Rao, K., Pertsch, K., Mordatch, I., Michalewski, H., Lu, Y., Levine, S., Lee, L., Lee, T.-W. E., Leal, I., Kuang, Y., Kalashnikov, D., Julian, R., Joshi, N. J., Irpan, A., Ichter, B., Hsu, J., Herzog, A., Hausman, K., Gopalakrishnan, K., Fu, C., Florence, P., Finn, C., Dubey, K. A., Driess, D., Ding, T., Chormanski, K. M., Chen, X., Chebotar, Y., Carbajal, J., Brown, N., Brohan, A., Arenas, M. G., and Han, K. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In Tan, J., Toussaint, M., and Darvish, K. (eds.), *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183. PMLR, 06–09 Nov 2023. URL <https://proceedings.mlr.press/v229/zitkovich23a.html>.

A. Structured Comparison of PhysMent to Key Benchmarks

Table 4 compares PhysMent to current representative benchmarks on physics-reasoning.

B. Per-Dimension Scores

Table 5 reports per-model scores across all six scoring dimensions.

Why the C column differs from Accuracy. The **Acc.** column is the strict pass/fail accuracy of Eq. (4) (a scene is either correct or not). The **Correctness dimension C** used inside the Final Score is a *graded* $[0, 1]$ quantity that awards partial credit on multi-part computation scenes (Section 3): a scene whose ground truth is a tuple (e.g., “angular velocity *and* rotational energy after 2s, then again after a second torque”) contributes the fraction of sub-answers within tolerance, rather than collapsing to 0 on any single miss. This is why, for example, GPT-5.5 has $\text{Acc.} = 50.0\%$ but $C = 0.583$ (partial credit raises C above the all-or-nothing pass rate), whereas GLM-5 has $\text{Acc.} = 45.2\%$ but $C = 0.351$ (it more often produces a correct *final* value while missing intermediate sub-answers, so its graded score falls below its strict pass rate). The two columns are therefore expected to differ; they are not an inconsistency.

C. Scene Construction Guidelines

Each scene is defined by two files under `Scenes/Scene{N}/` with matching filenames `scene{N}.json` and `scene{N}.xml` for $N \in [7, 111]$.

Operational category definitions. Table 6 gives the design-time rule used to assign each scene to a difficulty/modality cell, together with a representative scene. These are the rules referenced in Section 3; they are applied at construction time so that category membership is reproducible rather than subjective.

Metadata file (`scene{N}.json`). Specifies: scene name, task, and `problem_type`; ground-truth answer; `expected_behavior`; `reasoning`; object list with IDs; and `object_permissions` per attribute.

MJCF file (`scene{N}.xml`). A valid MuJoCo model. Each dynamic object requires a `<body name="object_{id}">` wrapper, a `<geom>` defining shape, size, density, and friction, and `<joint name="object_{id}-joint" type="free"/>` for unconstrained rigid-body motion.

D. Complete Tool API

Table 7 lists all 26 simulation tools exposed to agents, organized by function category as described in Section 3.

E. Prompting Strategies

Zero-shot. The prompt contains the task description, visible object attributes (filtered by P_i), full tool API documentation, coordinate-frame conventions, and answer-format instructions. No worked examples are provided.

One-shot / few-shot CoT. One or three worked examples are attached to the prompt: a dropped ball, a pushed box, and a sphere on a ramp, each with a reasoning trace and tool calls showing how to interact with the simulator.

F. Failure Mode Interaction Traces

This appendix presents the raw prompts and model responses for the six failure modes discussed in Section 5. Each example shows the initial task prompt, key model responses, submitted answer, and correct answer.

Scene 48: Premature Answer Submission

Model: DeepSeekAgent | **Budget:** $I_{\max} = 5$

Prompt: *Determine the coefficient of static friction using the critical angle at which the block begins to slide.* Objects: block at position $[-2.0, -0.6, 0.5]$, ramp at origin.

PhysMent

Table 4. Structural comparison of PhysMent with representative physical-reasoning benchmarks. *Inter.* = interactive; *Sim.* = grounded in a physics engine; *Multi-turn* = sustained adaptive interaction (vs. single invocation); *Proc. metrics* = reports process/trajectory metrics beyond correctness; *Budget* = studies an iteration/interaction budget. “-” denotes not applicable.

Benchmark	LLM	Inter.	Sim.	Multi-turn	Proc. metrics	Budget	# Tasks
PIQA (Bisk et al., 2020)	✓	×	×	×	×	–	~20k
GPQA (Rein et al., 2024)	✓	×	×	×	×	–	448
PhysReason (Zhang et al., 2025a)	✓	×	×	×	✓	–	1200
PHYBench (Qiu et al., 2025)	✓	×	×	×	✓	–	500
PHYRE (Bakhtin et al., 2019)	×	✓	✓	×	×	–	50 templates
ScienceWorld (Wang et al., 2022)	✓	✓	rule-based	✓	×	–	30 tasks
Mind’s Eye / UTOPIA (Liu et al., 2023)	✓	✓	✓	×	×	–	39 subtasks
PhysMent (ours)	✓	✓	✓	✓	✓	✓	105 scenes

Table 5. Per-model scores across all six scoring dimensions, averaged over available iteration budgets. *C* = Correctness, *E* = Efficiency, *G* = Groundedness, *V* = Action Validity, *R* = Reasoning Diversity, *G'* = Generalization, FS = Final Score (0–100). Final Score recomputed uniformly using fixed weights ($w_C, w_E, w_G, w_V, w_R, w_{G'}$) = (0.40, 0.15, 0.15, 0.10, 0.10, 0.10) for comparability across runs.

Model	Acc. (%)	<i>C</i>	<i>E</i>	<i>G</i>	<i>V</i>	<i>R</i>	<i>G'</i>	FS
GPT-5.5	50.0	0.583	0.537	0.729	1.000	0.701	0.476	64.1
Gemini 3.1 Pro	49.5	0.555	0.420	0.598	1.000	0.657	0.437	58.4
Claude Opus 4.7	40.7	0.503	0.655	0.544	0.999	0.613	0.580	60.0
DeepSeek R1	48.5	0.480	0.418	0.514	1.000	0.633	0.588	55.4
GLM-5	45.2	0.351	0.238	0.744	1.000	0.700	0.426	50.0
Kimi K-2.5	41.9	0.308	0.201	0.702	0.999	0.708	0.381	46.7
Qwen 3.5	28.6	0.224	0.280	0.712	1.000	0.678	0.432	44.9

Table 6. Operational definitions of the PhysMent difficulty/modality axes. *Ref. steps* = number of distinct simulator-intervention steps in the reference solution used to assign Easy (≤ 3) vs. Hard (> 3). *Concepts* = number of distinct physical concepts required (Single = 1, Multi ≥ 2).

Category	Concepts	Ref. steps	Modality rule	Representative scene
Easy Single Phys.	1	≤ 3	standard	free-fall velocity readout (Scene 7)
Easy Single + Obj.C.	1	≤ 3	must <code>create_objects</code> for a reference body	add reference mass (Scene 16)
Easy Single + Hid.	1	≤ 3	must <code>find_objects</code> to reveal a masked body	hidden ball discovery (Scene 21)
Hard Single Phys.	1	> 3	standard, adaptive loop required	friction threshold via angle search (Scene 26)
Hard Single + Obj.C.	1	> 3	object creation + adaptive loop	Scene 41
Hard Single + Hid.	1	> 3	hidden object + adaptive loop	Scene 46
Easy Multi	≥ 2	≤ 3	standard	combined kinematics+energy (Scene 51)
Easy Multi + Obj.C.	≥ 2	≤ 3	object creation	Scene 66
Easy Multi + Hid.	≥ 2	≤ 3	hidden object	Scene 71
Hard Multi	≥ 2	> 3	standard, adaptive loop	Scene 76
Hard Multi + Obj.C.	≥ 2	> 3	object creation + adaptive loop	Scene 91
Hard Multi + Hid.	≥ 2	> 3	hidden object + adaptive loop	Scene 96
Create / Del. / Att.	–	varies	scene-graph manipulation is the task	attach two bodies (Scene 101)

Iteration 1 response: The model computed the incline angle from static block coordinates: rise = 0.5 m, run = 2.0

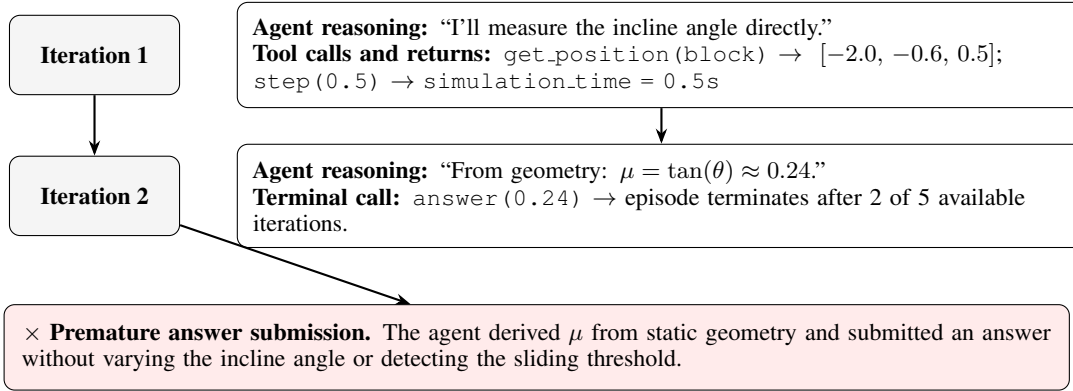


Figure 3. One PhysMent episode from Scene 48. The agent submits its answer after only 2 of 5 available iterations, illustrating premature answer submission: it derives μ from static geometry rather than experimentally varying the incline angle to detect sliding onset. See Appendix F for the full trace.

m , $\mu_s = \tan(\theta) = 0.25$. It then stepped the simulation once for 0.5 s to “verify”.

Iteration 2 (final): After observing displacement, the model refined the geometry calculation: $\text{run} = \sqrt{(-2.0)^2 + (-0.6)^2} \approx 2.088$ m, submitted $\mu_s = 0.240$.

Submitted: 0.24 **Correct:** $\mu_s = \tan(\theta_c)$ via experiment.

The model never varied the incline angle or detected the onset of sliding. It derived μ purely from static geometry in 2 of 5 available iterations.

Scene 89: Insufficient Simulation Time

Model: QwenAgent | **Budget:** $I_{\max} = 15$

Prompt: Determine the period of oscillation for a simple pendulum with a length of 2 meters.

Iteration 1–6: The model attempted to displace the pendulum via `move_object` (shape error), `apply_force`, and `step(0.1)`. After 0.1 s of simulation, position and velocity remained $[0, 0, 0]$ — the pendulum did not move.

Iteration 7 (final): Seeing no oscillation, the model abandoned simulation and applied the formula: $T = 2\pi\sqrt{L/g} = 2\pi\sqrt{2/9.81} \approx 2.843$ s.

Submitted: 2.843 s **Correct:** 2.837 s.

Only 0.1 s of simulation was run, which is 3.5% of one full period, and the pendulum never oscillated due to a failed displacement step. The model abandoned the simulator and fell back to the textbook closed-form formula rather than measuring the period experimentally. Although its hand-computed value happens to fall within the 5% tolerance of Eq. (4), the episode exemplifies the failure mode this category targets—abandoning the experimental loop—rather than a numerical scoring error.

Scene 63: Answer Format Mismatch

Model: GLMAgent | **Budget:** $I_{\max} = 20$

Prompt: Observe how angular velocity changes when masses are moved closer to the center. Objects: rotating platform, two masses at $r = 0.4$ m.

Iterations 1–18: The model made 45 tool calls over 18 iterations, attempting to spin the platform via `apply_torque`, `apply_force`, and `set_velocity`. All attempts produced near-zero angular momentum due to scene configuration issues.

Final response: The model fell back to theory, computing $I_{\text{initial}} = 0.00271$ kg·m², $I_{\text{final}} = 0.00258$ kg·m², and $\omega_{\text{final}}/\omega_{\text{initial}} = 1.05$. Submitted the ratio 1.05.

Submitted: 1.05 **Correct:** 2 (object ID of the faster-spinning mass).

The scene expected a comparison object ID, not a numerical ratio. The model’s physics was correct but it misread the answer format entirely.

Scene 68: Object ID Confusion

Model: GLMAgent | **Budget:** $I_{\max} = 20$

Prompt: Identify which rope has higher tension based on the mass it supports. Objects: `object_1` (mass_1, density 1.0), `object_2` (mass_2, density 2.0), `object_3` (rope_1), `object_4` (rope_2).

Iterations 1–7: The model correctly measured masses (0.00419 kg vs 0.00838 kg), identified `rope_2` above `mass_2`, and concluded `rope_2` has twice the tension.

Final response: “`rope_2` (`object_4`) has higher tension.” Submitted ID 4.

Submitted: 4 (`rope_2`) **Correct:** 2 (`mass_2`).

Table 7. All 26 simulation tools exposed to agents in PhysMent.

Tool	Description
step(dur)	Advance simulation by dur seconds
get_position(id)	Return $[x, y, z]$ and current simulation time
move_object(id, x, y, z)	Teleport object to absolute position
change_position(id, d, fr)	Translate by $[dx, dy, dz]$ in world or local frame
get_velocity(id)	Return $[v_x, v_y, v_z]$ linear velocity
set_velocity(id, v)	Set linear velocity vector
get_acceleration(id)	Estimate acceleration from velocity difference
apply_force(id, f)	Apply force $[f_x, f_y, f_z]$
apply_torque(id, τ)	Apply torque $[\tau_x, \tau_y, \tau_z]$
get_torque(id)	Return currently applied torque
get_parameters(id)	Return mass, bounding box, geometry type
get_displacement(id)	Euclidean distance from initial position
compute_force(id, m)	Compute $F = ma$
get_kinetic_energy(id, m)	Compute $\frac{1}{2}mv^2$
get_potential_energy(id, m, g)	Compute mgh
get_rotational_energy(id, m)	Compute $\frac{1}{2}I\omega^2$
get_momentum(id, m)	Compute $p = mv$
get_angular_momentum(id, m)	Compute $L = I\omega$
get_center_of_mass()	Scene-wide center of mass
detect_collision(id1, id2)	Check contact; apply elastic response
quat_to_rot_matrix(q)	Quaternion to 3×3 rotation matrix
create_objects(n, p, ρ , c)	Add sphere (modifies XML, reloads)
delete_objects(id)	Remove object (modifies XML, reloads)
find_objects()	Reveal hidden bodies by resetting colors
attach_objects(id1, id2)	Create joint between two bodies
answer(ans)	Submit final answer; terminate episode

The answer key expected the ID of the heavier mass, not the rope. The model’s physics reasoning was entirely correct but it submitted the rope’s ID instead of the mass’s ID.

Scene 91: Wrong Parameter Substitution

Model: GPT-5.5 | **Budget:** $I_{\max} = 15$

Prompt: Determine the rotational kinetic energy of a disk with radius 0.5 m, mass 2 kg, and angular velocity 10 rad/s.

Iteration 1: The model called get_parameters and received simulator mass = $\pi \approx 3.14159$ kg (computed internally from geometry and density).

Iteration 2 (final): “The simulator shows the actual disk mass is 3.14159 kg, so I should use that verified value.” Applied $KE = \frac{1}{4}mr^2\omega^2$ with $m = \pi$: result = 19.635 J.

Submitted: 19.635 J **Correct:** 12.5 J ($= \frac{1}{4} \times 2 \times 0.25 \times 100$).

The formula structure was correct but the model explicitly discarded the problem’s stated mass in favour of the simulator’s geometry-derived mass, treating the simulator as more authoritative than the problem parameters.

Scene 14: Simulation Trust vs. Theoretical Reasoning

Model: KimiAgent | **Budget:** $I_{\max} = 10$

Prompt: Determine which block will travel further when pushed with the same force. Objects: object_1 (low friction), object_2 (high friction), equal mass.

Iterations 1–9: The model applied equal forces to both blocks and stepped for 2 s. Measured displacements: object_1 = 0.100 m, object_2 = 4.001 m — a 40× difference inconsistent with friction physics.

Final response: “Physically, the low friction block should travel further ... However, my simulation results indicate that object_2 (high friction) traveled significantly further.” Submitted object_2.

Submitted: 2 (high friction) **Correct:** 1 (low friction).

The model explicitly identified the physical contradiction in its own reasoning but deferred to the anomalous simulator output. The displacement anomaly was caused by incorrectly initialized friction coefficients in the scene XML.

G. Examples of JSON/XML Files, Initial Scene Prompt for Experiment Logs

Example Scene JSON and XML Specifications: We present the corresponding JSON metadata file and XML scene specification for the example rotation and kinetic-energy task. The JSON file defines the task-level structure exposed to the evaluation pipeline, including the scene name, task description, problem type, ground-truth answer, expected behavior, object list, and object-level permissions. The XML file encodes the underlying MuJoCo scene, specifying the physical object geometry, pose, density, and joint configuration used by the simulator during tool-mediated experimentation.

JSON Metadata: Scene-Level Task Definition

```
{
  "metadata": {
    "scene_name": "Rotation and Kinetic Energy",
    "task": "A disk is set in motion by applying a torque of 20 N  $\cdot$  m. First, calculate its angular velocity and rotational energy after 2 seconds of simulation. Then, apply a torque of 10 N*m and calculate the new angular velocity and rotational energy after 2 seconds.",
    "problem_type": "computation"
  },
  "answer": "4.0, 79.6, 6.0, 179.0",
  "expected_behavior": "The LLM should apply a torque to rotate the object and calculate the angular velocity and rotational energy after 2 seconds of simulation.",
  "reasoning": "By applying torque, the object will rotate and accumulate angular momentum. The rotational energy can be calculated using the formula for kinetic energy of rotation.",
  "number_of_objects": "1"
}
```

JSON Object Registry

```
{
  "objects": {
    "object_1": {
      "name": "disk",
      "object_id": "1"
    }
  }
}
```

JSON Object-Permission Mask

```
{
  "object_permissions": {
    "object_1_permissions": {
      "geom_type": true,
      "geom_density": true,
      "body_mass": true,
      "geom_size": true,
      "geom_radius": true,
      "body_pos": true,
      "body_quat": true
    }
  }
}
```

XML Scene Header and Global Physics Options

```
<mujoco model="
  rotation_kinetic_energy_test">
  <option gravity="0 0 -9.81"/>
  <worldbody>
```

XML Dynamic Object Definition

```
  <!-- Disk -->
  <body name="object_1" pos="0 0
1">
    <geom type="cylinder" size
="0.2 0.05" density="40000" rgba="0 1
0 1"/>
    <joint name="object_1_joint"
type="free"/>
  </body>
```

XML Scene Closure

```
</worldbody>
</mujoco>
```

Specification structure: The JSON file separates semantic task information from simulator state. The metadata field specifies the natural-language task, scene name, and problem type; the answer field stores the target output used for evaluation; and the expected_behavior and reasoning fields document the intended experimental strategy. The object registry maps symbolic object identifiers, such as object_1, to named physical entities, while the permission mask determines which attributes are exposed to the LLM through the prompt and tool interface.

Simulator encoding: The XML file provides the executable MuJoCo scene loaded by the evaluation runtime. In this example, the scene contains a single disk represented

as a cylindrical geometry with radius and half-height specified by `size="0.2 0.05"`. The body is initialized at position `pos="0 0 1"` and assigned a free joint, allowing the simulator to update its translational and rotational state under applied forces and torques. This separation between JSON metadata and XML physics specification allows the benchmark to expose controlled task information to the LLM while preserving a grounded simulation backend for tool execution.

Initial Scene Prompt for Experiment Logs: We present an example initial scene prompt given to the LLM at the beginning of an experimental episode. The prompt provides task-level metadata extracted from the scene JSON file, while the associated XML scene specification is loaded by the evaluation pipeline to enable valid simulator interactions through tool calls. To improve readability, the prompt is divided into the following components: Scene Description and Task, Available Objects and Parameters, Tool API Listing, Tool Return Format and Answer Submission, Example Assistant–Environment Interaction (for one-shot or few-shot chain-of-thought prompting), Answer Format Requirements, Final Submission Guidelines, and JSON Tool-Call Formatting Rules.

Scene Description and Task

Scene Description: Rotation and Kinetic Energy.

Task: A disk is set in motion by applying a torque of 20 N*m. First, calculate its angular velocity and rotational energy after 2 seconds of simulation. Then, apply a torque of 10 N*m and calculate the new angular velocity and rotational energy after 2 seconds.

Available Objects and Parameters

Available Objects and Parameters:
Object id: `object_1`, Object name: `disk`

Quaternion: A quaternion is used to represent rotation in 3D space. The four numbers represent rotations along the X, Y, Z axes and a scalar component.

If an attribute has 'n/a' right beside it, that means you CANNOT access that attribute's value, so keep that in mind when running through the experiment.

Tool API Listing

You may use the following tools along with their description to interact with the scene. These functions accept parameters given below, and return data or perform simulation updates:

```
[
  {
    "name": "step",
    "description": "keeps on moving the simulator forward in time",
    "arguments": {"duration": "float"},
    "return type": {"results": null}
  },
  {
    "name": "apply_force",
    "description": "applies a force vector to an object",
    "arguments": {"object_id": "str", "force_vector": "list[float]"},
    "return type": {"status": "str", "object_id": "int", "force": "list[float]"}
  },
  {
    "name": "get_velocity",
    "description": "retrieves the velocity vector of an object",
    "arguments": {"object_id": "str"},
    "return type": {"velocity": "array"}
  },
  ...
  {
    "name": "answer",
    "description": "submits an answer back to the system for checking or logging",
    "arguments": {"answer": "str or float"},
    "return type": {"acknowledged": "bool"}
  }
]
```

Tool Return Format and Answer Submission

Every time you call a tool, you will receive a dictionary containing the outputs. For example, if you call `'get_velocity'` on `'object_1'`, the return might be:

```
{"vx": 0.0, "vy": -3.2, "vz": 0.0}
```

You only have **one chance** to answer the question. When you're confident, submit your final answer using:

```
{"tool": "answer", "parameters": {"answer": "<your_answer>"}}
```

Example Assistant–Environment Interaction

<THIS IS AN EXAMPLE PROBLEM OF THE INPUTS (ASSISTANT) AND OUTPUTS (ENVIRONMENT) THAT SHOULD TAKE PLACE>

Problem: You are given a ball and a ground surface for reference. Drop the ball from a height of 10 units and figure out the velocity of the object after 0.5 seconds.

```
<assistant>
I see that I have to move the ball up 10
units so I will do that.
json
[{"tool": "move_object", "parameters": {"
  object_id": "object_1", "x": 0, "y":
  0, "z": 10}}]
```

```
<environment>
Results: [{"tool": "move_object", "
  parameters": {...}, "result": {"
  position": [0, 0, 10]}, "sim_time":
  0}] What will you do next
```

```
<assistant>
Now I will simulate by using the step
function to go 0.5 seconds forward.
json
[{"tool": "step", "parameters": {"
  duration": 0.5}}]
```

```
<environment>
Results: [{"tool": "step", "parameters":
  {...}, "result": null, "sim_time":
  0.5}] What will you do next
```

```
<assistant>
Now I will use the get velocity function
to figure out what I should output as
my answer.
json
[{"tool": "get_velocity", "parameters":
  {"object_id": "object_1"}}]
```

```
<environment>
Results: [{"tool": "get_velocity", "
  parameters": {...}, "result": {"
  velocity": [0, 0, -4.9]}, "sim_time":
  0.5}] What will you do next
```

```
<assistant>
Now I will call back the answer.
json
[{"tool": "answer", "parameters": {"
  answer": "-4.9"}}]
```

<END EXAMPLE>

Answer Format Requirements

Since the problem is a computation problem, your answer should be the calculated number that satisfies the task rounded to the nearest thousandths place if applicable.

Final Simulation Guidelines

FINAL GUIDELINES

You must call 'step' to simulate time progression.

Do not make any assumptions about the positions or states of objects; if you are unsure, you can use tools to get this information.

The z plane is the vertical plane, and the x and y planes are the horizontal planes.

Understand that you can use the tools to gather necessary information pertaining to all objects in the scene, not just the one you are trying to analyze.

If your json format is incorrect, the environment will tell you and the simulator will remain in the same state. If one of your tool calls has incorrect formatting, the previous tool calls will successfully execute but the incorrect tool and subsequent tools will not execute.

JSON Tool-Call Formatting Rules

Remember that YOU MUST PROVIDE YOUR REASONING FOR EVERY ACTION you take, and then make sure to add a valid JSON format of an array of tool calls.

When you are trying to call functions, use a string that goes as object_{object_id} for the object id, and use the name of the function as the tool name.

For example, if you were to try and call the function get_velocity on object_1, you would write it as:

```
json
[{"tool": "get_velocity", "parameters":
  {"object_id": "object_1"}}]
```

RATHER THAN WRITING IT AS:

```
json
[{"tool": "get_velocity", "parameters":
  {"object_id": 1}}]
```

Remember that when giving multiple tool calls, you must give them in ONE SINGLE GO, as shown below.

```
json
[{"tool": "get_velocity", "parameters":
  {"object_id": "object_1"}},
{"tool": "get_velocity", "parameters":
  {"object_id": "object_2"}}]
```

DO NOT GIVE MORE THAN ONE SET OF TOOL CALLS AT A TIME, OR THE SIMULATOR WILL NOT WORK, AND END UP GIVING YOU AN ERROR FOR JSON SYNTAX!!!