Agentic Exploration of Physics Models

Maximilian Nägele* and Florian Marquardt

Max Planck Institute for the Science of Light, Staudtstraße 2, 91058 Erlangen, Germany and Department of Physics, Friedrich-Alexander Universität Erlangen-Nürnberg, Staudtstraße 5, 91058 Erlangen, Germany

(Received 12 November 2025; accepted 22 May 2026; published 8 July 2026)

The process of scientific discovery relies on an interplay of observations, analysis, and hypothesis generation. Machine learning is increasingly being adopted to address individual aspects of this process. However, it remains an open challenge to fully automate the heuristic, iterative loop required to discover the laws of an unknown system by exploring it through experiments and analysis, without tailoring the approach to the specifics of a given task. Here, we introduce SCIEXPLORER, an agent that leverages large language model tool-use capabilities to enable exploration of systems without any domain-specific blueprints and apply it to physical systems that are initially unknown to the agent. We test SCIEXPLORER on a broad set of models spanning mechanical dynamical systems, wave evolution, and quantum many-body physics. Despite using a minimal set of tools, primarily based on code execution, we observe impressive performance on tasks such as recovering equations of motion from observed dynamics and inferring Hamiltonians from expectation values. The demonstrated effectiveness of this setup opens the door toward similar scientific exploration in other domains, without the need for fine-tuning or task-specific instructions.

DOI: 10.1103/xnqc-q6nt

Subject Areas: Condensed Matter Physics, Nonlinear Dynamics, Optics

I. INTRODUCTION

In recent years, specialized machine learning techniques have been increasingly employed to address various individual aspects of the scientific discovery process, for example, by suggesting new experiments [1–3], predicting or analyzing experimental results [4–6], learning meaningful representations of data [7,8], or recovering governing equations as symbolic expressions [9]. However, these tools are typically designed to solve a specific task in a specific setting. Large-language models (LLMs) exceed the capabilities of the usual machine learning approaches in several ways. Famously, they can work on a large variety of tasks without any specialized training dataset, often in a zero-shot manner, i.e., without even providing a few exemplary solution templates [10]. They work well with arbitrarily structured input, including multimodal data mixing text and images [11] and can reason, producing a textual record of their arguments. As a result, LLMs can work well in situations where heuristic approaches and some level of creativity are required, all the time relying on their general factual knowledge garnered during training. For the tasks we will consider here, this includes, in particular, math and physics knowledge, supplemented by outstanding abilities in coding [12,13]. At the same time, the greatest weakness of LLMs is their propensity to hallucinate. One approach to overcome this to some extent, also relied on in the present work, is to focus on scenarios where some level of self-correction is possible, with independent ways to check and repudiate results [14].

In the past few years, the power of LLMs has been boosted by agentic behavior (tool use). By allowing the LLM to call up external software tools, one can combine the reliability of those tools with the heuristics and general knowledge of the LLM [15]. First examples of agentic behavior have recently been introduced into several scientific domains. In computer science, LLMs conduct complex coding tasks [16], and can, to some extent, also suggest new research ideas and summarize their findings [17]. In chemistry, they combine Web search, specialized simulation tools, and access to laboratory interfaces to, for example, optimize chemical reactions [18] and synthesize molecules [19,20]. In biology, agentic LLMs are employed to find gene perturbations resulting in specific phenotypes [21] and design gene-editing experiments [22].

*Contact author: maximilian.naegele@mpl.mpg.de

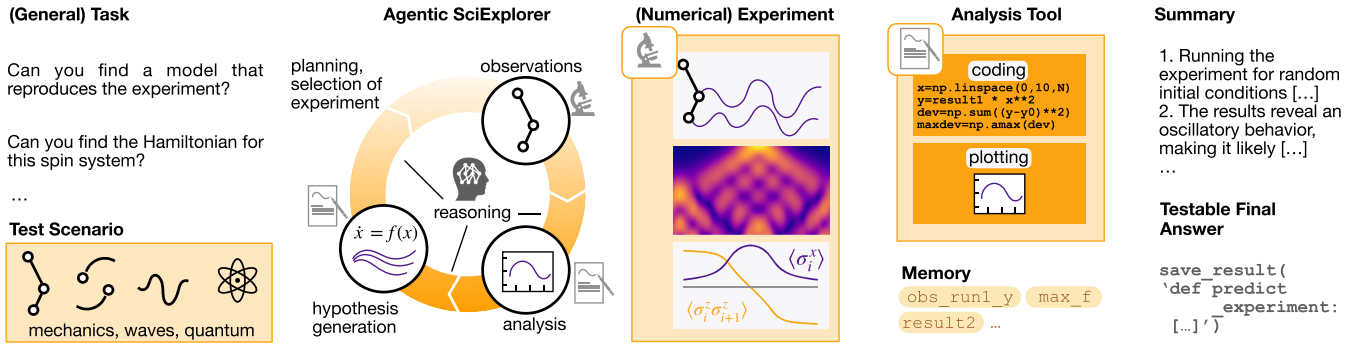

FIG. 1. Agentic SCIEXPLORER. The explorer is given a general task, which is then applied to a specific physics scenario. During the scientific exploration cycle, LLM-based reasoning is employed to select a (numerical) experiment to be carried out, calling the appropriate tool. The resulting observations are analyzed using one or several calls to analysis tools, often involving both the coding and the multimodal capabilities of the LLM. This cycle repeats until the LLM agent decides it has acquired sufficient information. Finally, a summary of the reasoning chain is produced for the human user. For benchmarking, the agent is asked to provide a testable final answer, e.g., in the form of code that can be run and evaluated.

In physics, nonagentic LLMs have been used to solve well-defined tasks such as calculating Hartree-Fock Hamiltonians [23] or answering questions about astrophysics [24]. Additionally, their performance has been tested on a suite of benchmarks containing typical exam questions ranging from high school to Olympiad-level difficulties [25–27], as well as on typical scientific coding tasks [28].

While exploring the capabilities of agentic LLMs in physics has been called for [29,30], previous work focuses on building specialist agents with specific tools for tasks that require adaptability but typically follow a structured blueprint. Examples include density functional calculations [31], cosmology calculations [32,33], calculating properties of topological materials [34], inverse design for metamaterials [35], designing integrated photonic circuits [36], cooling of trapped atom experiments [37], and calibrating superconducting quantum processors [38].

In this work, we focus on exploring the capabilities of a general AI physicist in a verifiable setting requiring heuristic, open exploration. Crucially, and in contrast to previous work, the artificial physicist is given only minimal task-specific instructions. Therefore, if it performs well on the benchmark problems, we can expect good performance also on other tasks. Specifically, we will explore how tool use can enable an LLM to automate the heuristic loop of scientific discovery, including selecting experiments, analyzing results, and forming hypotheses when exploring an unknown system.

The tools we consider include running experiments (here, numerical experiments), performing data analysis through a general coding tool, and visualizing results. Based on these tools, the LLM can use its reasoning to drive an active learning process, deciding which analysis or experiment to carry out next based on previous results. We will explore in prototypical physics scenarios, drawn from fields like dynamical systems, wave dynamics, and quantum many-body systems, how this approach can successfully solve research-oriented tasks requiring heuristics by mimicking essential aspects of the scientific process. Tasks we will consider include discovering the equations of motion of a system by observing its dynamics and discovering Hamiltonians of quantum many-body systems from measurements.

II. AGENTIC SCIEXPLORER

The agentic SCIEXPLORER is based on an LLM agent that autonomously selects and analyzes experiments (see Fig. 1), where "experimental" results here are obtained via numerical simulations. In a typical scenario, the agent specifies initial conditions, observes the time evolution of a system, and adaptively selects subsequent experiments based on its previous analysis. To support this process, the agent can access two generic analysis tools. Rather than providing narrowly defined, user-specified tools, we leverage the advanced coding capabilities of modern LLMs [28], giving the agent access to a coding tool that executes arbitrary PYTHON code. This choice maximizes flexibility and enables dynamic tool creation by the agent during runtime. Additionally, the agent can access a code-based plotting tool, which, paired with the agent's multimodal capabilities, enables the agent to quickly grasp qualitative insights about the system under study by visual inspection. We integrate an external memory that is accessible through the analysis tools, to offer reliable persistence of experimental and analysis results in the form of arrays of numerical values. When addressing a user-defined task, the agent proceeds in multiple steps. Each step involves verbal reasoning and may include multiple tool calls for controlling the experiment and analyzing results. The tool results are then communicated to the agent and added to the memory (see Fig. 2 and Appendix A for a more complete description).

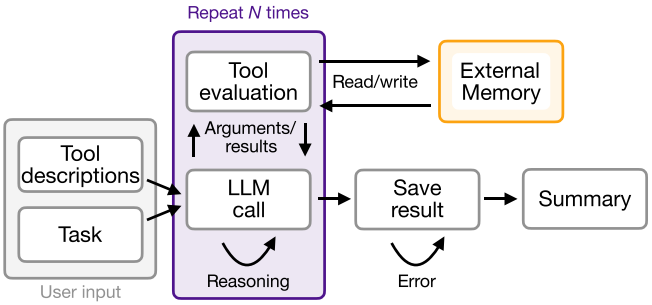

FIG. 2. SCIEXPLORER scheme. Given task and tool descriptions supplied by the user, SCIEXPLORER executes up to N (in our case 100) steps fully autonomously. In each step, SCIEXPLORER can reason verbally and specify arguments for tools it wants to evaluate. The tools are then executed, reading and writing to an external memory containing all previous tool results. The results are typically returned to SCIEXPLORER in plain text. However, for large numerical arrays, SCIEXPLORER receives a description of their type and shape. When SCIEXPLORER calls the `save_result` tool, the exploration stops. In case of a syntax error, SCIEXPLORER can try again, otherwise it is asked to summarize the exploration.

To explore the intrinsic capabilities of state-of-the-art LLMs in successful agentic exploration, we deliberately keep any domain-specific input to a minimum. The relatively short system prompt provides only general advice for scientific exploration (like systematically formulating hypotheses) but does not contain task-specific or even domain-specific information and is applied across all domains explored in the following (see Appendix F). When providing information to the agent through the system description or the documentation of the tools executing experimental runs, we follow a simple philosophy: We provide only information that an experimentalist investigating such a system would have access to. This includes, for example, the dimensionality, topology (e.g., 1D vs 2D), and constituents (e.g., particles vs spins) of the system but not the general shape of its governing laws. Therefore, we do not even tell the agent that mechanical systems are governed by ordinary differential equations or wave systems by partial differential equations.

We evaluate SCIEXPLORER on two qualitatively distinct classes of tasks. In the first scenario, the agent is asked to reproduce the experimental results by implementing a PYTHON function. This setting involves not only model discovery but is closer to the even more challenging task of program discovery, as the solution space consists of the vast set of all valid functions. In this case, correct solutions typically include both a simulator of the system and a description of the system's governing laws. In the second scenario, the agent is asked to identify the model of the system within a broad but predefined model class. The advantage of this approach is that the resulting models tend to be more interpretable. However, it requires some limited prior knowledge about the class of admissible models. To evaluate SCIEXPLORER on both task types, we consider the first scenario for mechanical and field systems, and the second scenario for quantum systems, where the agent is asked to determine the Hamiltonian governing the experiment.

If not otherwise specified, we use GPT 5 [39] as the state-of-the-art LLM backbone of the SCIEXPLORER.

III. RESULTS

SCIEXPLORER is able to recover accurate models even when starting from minimal information about the system under consideration. Below, we explicitly present all information provided for a representative example: the damped double pendulum (see Supplemental Material [40] for details on all systems). The system and task description reads: "You are investigating a dynamical physical system. Can you find a model that reproduces the `observe_experiment` function? After your exploration, save it using the `save_result` function." The `observe_experiment` tool provides only the additional information, "The system has 2 generalized coordinates." It also includes a description of the format of its arguments, namely, the initial generalized coordinates, initial generalized velocities, the start and end times of the experiment, and the time resolution.

A. Mechanical systems

Mechanical systems provide an ideal test bed for the SCIEXPLORER due to their wide diversity spanning conservative, dissipative, and driven regimes, and the wide range of accessible complexity levels, selected via choosing the equations of motion or their dimensionality. An example of an illustrative exploration process is summarized in Fig. 3.

For test purposes, we considered generic dynamical systems with up to three generalized coordinates and systems with particles moving in 2D (with up to ten particles) subject to fixed potentials or pairwise interactions (for the full equations of motion, see Supplemental Material, Table S1 [40]). Figure 4 summarizes the agent's performance across these systems when asked to infer their equations of motion. The agent receives information only about the number of generalized coordinates (or number of particles moving in 2D) and is asked to produce a PYTHON function that reproduces the experimental results. To quantify the agent's performance, we compute the coefficient of determination R^2 between the velocities and accelerations predicted by the agent and the ground-truth dynamics.

As an expanded challenge, we also consider dynamical systems where the agent is allowed to observe only a subset of particles, and is not even told the number of unobserved particles. In those cases, we additionally require the agent to infer the initial conditions of the hidden particles. We then report the mean R^2 value across trajectories with

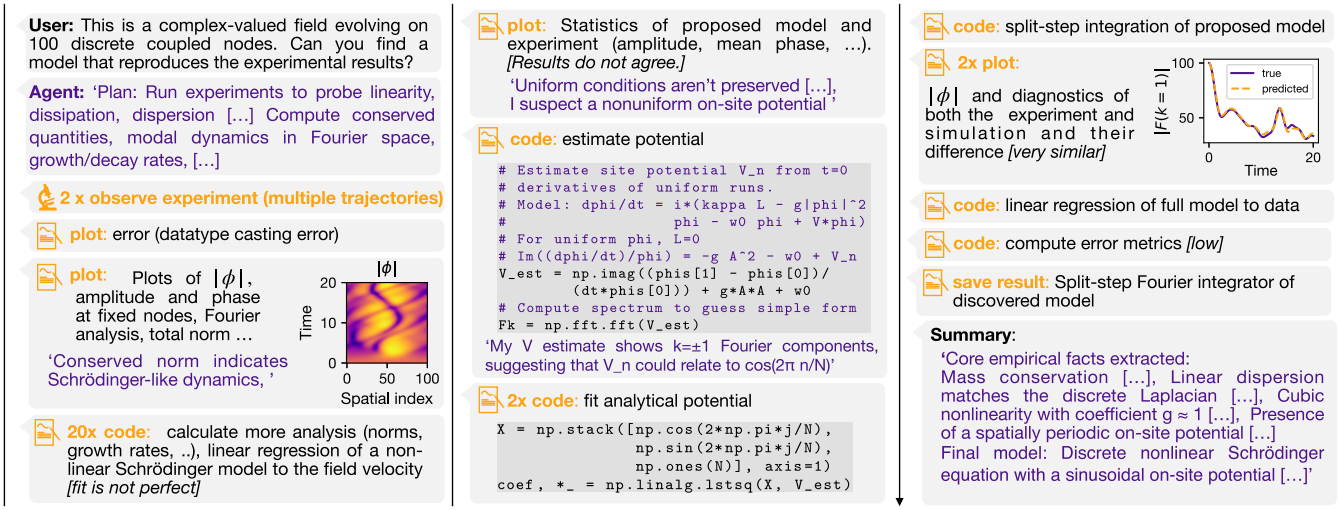

FIG. 3. Example exploration. The agent is tasked with discovering the unknown model of a complex field system by observing and analyzing its dynamics. In this figure, we sketch in a compressed fashion some of the steps in the resulting extended autonomous exploration. The agent first runs several experiments with varying initial conditions. Then, it infers the qualitative model (here, a nonlinear Schrödinger equation with external potential) from visualizations and analysis based on the experimental results. It subsequently estimates the external potential and discovers its analytical cosine structure. To validate the proposed model, the agent simulates time evolution using a (self-coded) split-step integrator and compares the results to experimental data. Finally, it saves its result as a PYTHON function that reproduces the experimental results and contains both a simulator of the system and the underlying partial differential equation. The sequence of steps depends entirely on the choices of the agent, which are adapted to the given problem, reacting to its own observations and conclusions.

random initial conditions of the observable particle (see Appendix B for details).

For a large subset of systems, the agent recovers the governing model both qualitatively and quantitatively, often achieving perfect fits ($R^2 \approx 1$) using only observations of the system's time evolution under initial conditions selected by the agent. Performance degrades for more complex or atypical systems, and the agent only sometimes recovers the correct model. Systems that are not close to any known models (e.g., the "arbitrary 2D potential") are particularly challenging, indicating that the LLM agent relies on its broad knowledge base to be successful for other systems. To identify systems correctly in most cases, the agent autonomously implements a broad repertoire of analysis techniques by generating and executing PYTHON code, without requiring specific tools. Typically, the agent first extracts qualitative signatures from visualizations. These include, among others, linearity, mirror symmetry, limit cycles, attractors, and decay. It then constructs candidate basis terms for the governing differential equations and fits their coefficients to the observed accelerations. When applicable, it uses sparse linear regression [41]. Otherwise, it fits numerical solutions of the proposed differential equations directly to the experimental data. Depending on the task, the agent uses a large set of additional strategies, such as locating centers of attraction in 2D by maximizing the radial acceleration with respect to a candidate center, testing for angular momentum and energy conservation, and using Fourier transforms to find initial parameter guesses for oscillator systems or to infer the number of hidden particles in partially observed systems (for example explorations, see Supplemental Material, S3A and S3B; for the models discovered by the agent, see Supplemental Material, Table S2 [40]).

Furthermore, we test the agent's ability to self-critique by asking it, for each mechanical system, to identify in which of multiple conversations it performed best (stars in Fig. 4). When the correct model was found at least once, the agent reliably selected that conversation. However, when none of the proposed models was exactly correct, the agent often did not choose the conversation corresponding to the highest R^2 value. Instead, it tends to prioritize qualitative agreement, such as the existence of bound and unbound trajectories, over purely quantitative fit quality.

It is important to note that even in a real experimental setting, one can verify the quality of the solution discovered by the agent without knowing the ground truth by comparing the agent's theory with new experimental data. Therefore, one can boost the effective success rate of the agent by running several explorations and picking the best-performing model (in all but three of the scenarios of Fig. 4, this leads to the correct solution, as it is among the five runs).

To understand which components of the SCIEXPLORER are essential, we perform ablation studies. Depending on the physical system, both the plotting and coding tools are crucial for consistent performance. To test the importance of tool access, we compare a given exploration run of our

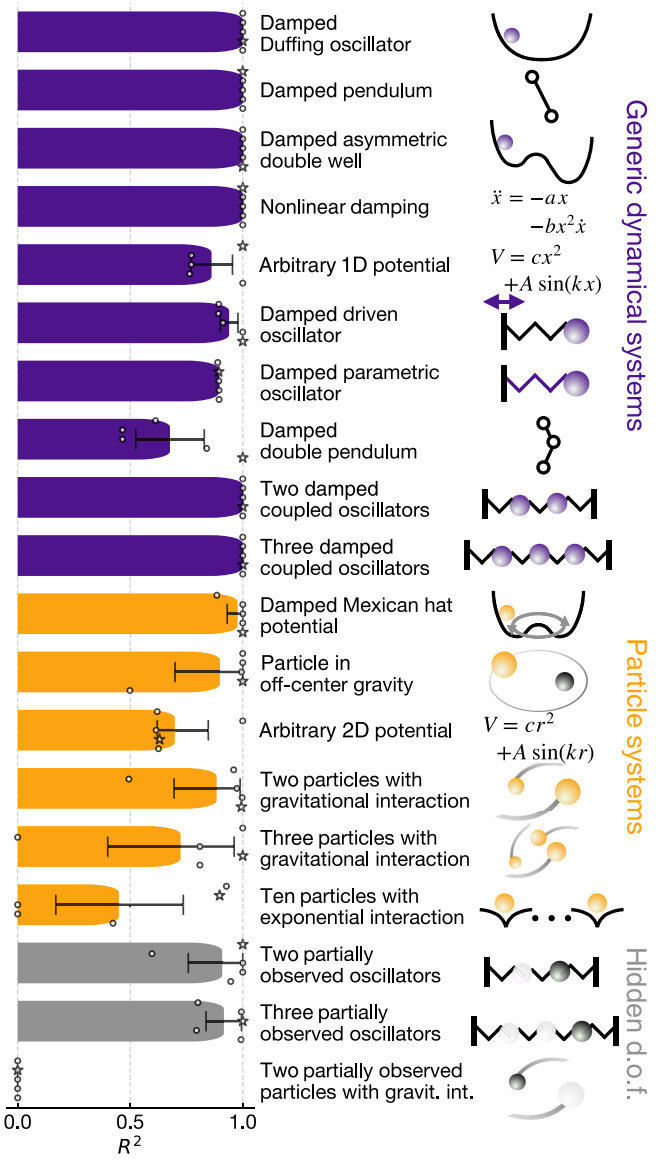

FIG. 4. Mechanical systems. The agent discovers the equations of motion of black-box physical systems by specifying initial conditions, observing dynamics, and analyzing the results. We consider generic dynamical systems with 1–3 generalized coordinates, particles moving in 2D, and systems where an observable particle interacts with an unknown number of hidden degrees of freedom (d.o.f.). We run 5 independent attempts per system and show the mean coefficient of determination R^2 of the agent's proposed model with the true system with 95% bootstrap confidence intervals. The agent can recover the true model in a large subset of systems. Stars indicate the conversation the agent considers best when asked to rank all attempts.

agent against a conversation where an LLM without access to tools is initially provided with visualizations for the same number of experiments, but now with randomly chosen initial conditions. In this case, the LLM cannot recover the correct model. Interestingly, we also find that GPT 5 [39] performs much better than Gemini 2.5 pro [42] (for details, see Appendix E).

The superior performance of LLM agents based on state-of-the-art reasoning models relies on significant run-time investment. For the systems studied here, a single exploration ranges from a few minutes to over 1.5 h, with the bottleneck being the LLM's response time, not the numerical simulations of the system. We also observe that the agent generally requires more time to solve more complex systems than simpler ones (see Fig. 10). Nevertheless, based on our experience as working theoretical physicists, we estimate that the overall run-time for a single task is already lower than what even an expert human would typically need for such an initially completely open exploration of an unknown system.

B. Dynamics of waves and fields

A fundamental cornerstone of the physical description of nature is the notion of dynamics of many degrees of freedom arranged in an extended space, with locality and translational invariance as guiding principles. These lead naturally to the evolution equations of fields and waves, which also form the basis of the most fundamental theories we have. We ask the LLM agent to explore this domain by providing access to experimental runs where the agent can set up an arbitrary initial configuration of a field ϕ and observe the subsequent dynamics $\phi(x,t)$ that is governed by a classical evolution equation unknown to the agent. The agent is provided only with information about the system's topology: "This physical system consists of a complex-valued field evolving on 100 discrete coupled nodes. The nodes are arranged in 1D with periodic boundary conditions, i.e., the first and last node are neighbors." As before, the agent is asked to implement a function that reproduces the experimental observations. Examples that we tested the agent on include the nonlinear Schrödinger equation, describing the dynamics of matter waves of interacting atoms or light waves subject to nonlinearities, and the time-dependent Ginzburg-Landau equation, describing the evolution of an order parameter field, e.g., in a superconductor, relaxing to its thermodynamic equilibrium configuration. In each case, there can be variations like the presence of an arbitrary external potential or of longer-range coupling terms on the underlying lattice. The SCIEXPLORER can discover the true model for any of these variations when given multiple attempts (see Fig. 5). However, it struggles with an artificial model describing relaxation into a sinusoidal potential for the field.

In our studies, we observe that the agent often lays out a detailed exploration plan in advance, mentioning the most important field theories as plausible options, sometimes including expected signatures, such as Bragg scattering for linear waves in periodic potentials or solitonlike features for Kerr-type and other nonlinearities. It tends to start with a Gaussian wave packet as its choice of initial condition in the first experiment. If the results of this experiment indicate wavelike dynamics, the agent often adopts a

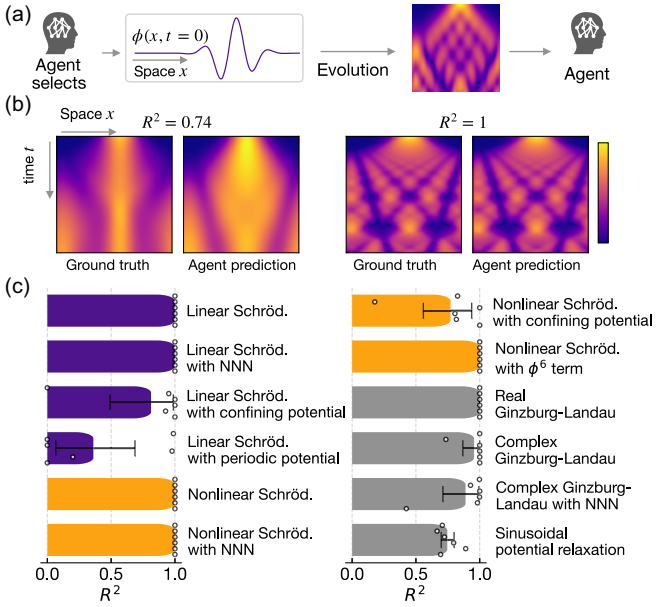

FIG. 5. Waves and fields. (a) In its exploration of waves and fields, the agent can select the initial field configuration for each experimental run. (b) Evolution of the absolute square $|\phi|^2$ of a Gaussian wave packet for the true model and the agent's discovered model (announced by the agent at the end of the exploration). The true model on the left is a linear Schrödinger equation with confining potential (we do not show the most accurate model discovered by the agent in its multiple runs). On the right, the true model is a complex Ginzburg-Landau equation with next-nearest-neighbor (NNN) hopping on a tight-binding lattice. The R^2 value is calculated between the evolution equations for the true and the predicted model, for multiple reasonable initial conditions (see Appendix C for details). (c) Statistics for various scenarios. We run six independent explorations, and the agent can recover the true model ($R^2 \approx 1$) in all but one scenario in at least one attempt.

principled approach where it first tries to extract the dispersion relation. One of its strategies is to launch several plane waves of varying wave number and low amplitude to avoid nonlinearities, and to analyze the evolution of the phase of the field ϕ. Once that is settled, it tries to determine any nonlinearities by checking the amplitude dependence of the evolution. In linear systems, the agent often fits a correct 100×100 Hamiltonian to the data but cannot save it, since it is stored only in the external memory not accessible by the `save_result` tool (counted as $R^2 = 0$). If the initial dynamics are not wavelike but contractive, with dissipation leading to some steady state, it will adapt its strategy accordingly. In both cases, the agent often visualizes the field $\phi(x, t)$ using space-time density plots, e.g., with the declared aim to "spot qualitative structures (breathers, oscillations, solitons)." To compare its proposed model with observational data, the agent typically implements a split-step integrator or, in linear systems, diagonalizes the Hamiltonian to simulate the proposed field equation. Using these strategies, we find it to quickly make systematic progress (see Supplemental Material, Table S4, for the models discovered by the agent and Supplemental Material, S3D, for an example exploration [40]).

C. Quantum many-body physics

In domains ranging from condensed matter physics to particle physics, the fundamental language is that of quantum many-body systems. The essence of this domain, with complexities like the exponential growth of the Hilbert space with particle number and the presence of entanglement, is captured well by models of interacting spins or qubits. We therefore let the agent explore and analyze spin Hamiltonians, a task with wide-ranging applications from solid-state physics to quantum technologies [43,44]. In terms of experiments, we test two variations. (i) In one setting, the agent can initialize the system in a product state and then observe the temporal evolution of the expectation values of single-spin operators. (ii) In another setting, the agent can construct arbitrary observables (Hermitian operators) and obtain their expectation values in the ground state of the unknown Hamiltonian (see Fig. 6). In both settings, the agent is provided with information that the system has a 1D or 2D configuration and that it consists of spins.

We find that the agent has an active working knowledge of quantum many-body physics and spin models, in particular, including standard models like transverse Ising, XXZ, and Heisenberg, but also, e.g., Dzyaloshinskii-Moriya (spin-orbit induced) chiral couplings. It can deploy that knowledge reliably in this challenge.

For the dynamical setting, where the time traces of spin expectation values can look very complex, the agent can nevertheless quickly rule out some hypotheses. For example, it may run short-time dynamics for multiple initial states, which allows it to first check for conserved quantities such as the global magnetization, compute dominant frequency components using Fourier transforms, and discriminate local and many-body interactions. It then often obtains fits to hypothesized families of Hamiltonians (for an example, see Supplemental Material, S3E [40]). We also introduced an even more challenging scenario, where the agent can only observe and control a subset of spins. As expected, this appears to be one of the most complex tasks, and the agent only sometimes makes good progress.

In the ground-state setting, the agent typically first probes the symmetry and structure of the Hamiltonian by selecting a limited set of intelligently chosen operators, obtaining single-spin expectation values as well as nearest-neighbor correlators but also more subtle quantities suited, e.g., for exploring potential chirality, like $\langle \hat{S}_x^j \hat{S}_y^{j+1} - \hat{S}_y^j \hat{S}_x^{j+1} \rangle$. It quickly deduces aspects like open boundary conditions, translational invariance, and the likely absence of some families of spin-coupling terms in the Hamiltonian. It then selects candidate spin Hamiltonians,

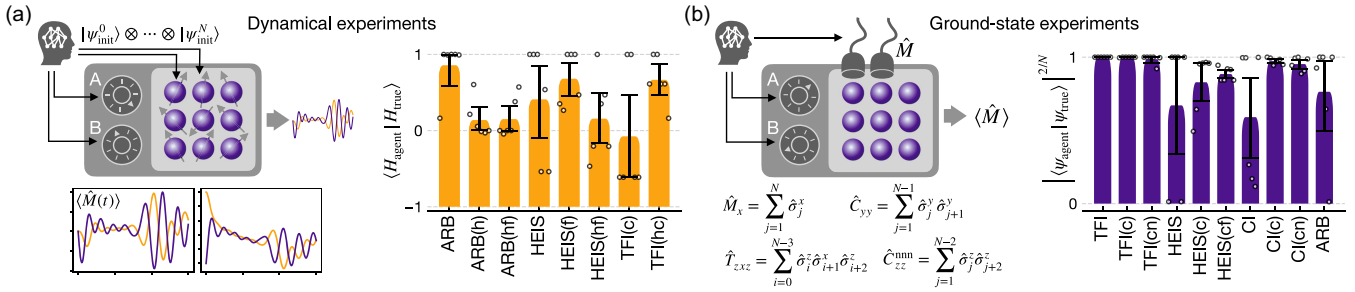

FIG. 6. Quantum many-body physics. (a) Discovering the Hamiltonian of a system of multiple spin-1/2 particles, based on observing the dynamics for initial conditions selected by the agent. Each experimental run produces the evolution of the single-spin expectation values $\langle \hat{M}(t) \rangle$ (with $\hat{M} = \hat{\sigma}_j^x$, $\hat{\sigma}_j^y$, and $\hat{\sigma}_j^z$), for an initial (product) state selected by the agent. In some experiments, we ask the agent to discover a whole class of Hamiltonians by allowing it to control experimental parameters, whose meaning it is not aware of ("A" and "B"). The bottom left shows an example—the complex dynamics of two spins of a Heisenberg model. Right: performance for various scenarios. Normalized scalar product between the true Hamiltonian and the Hamiltonian proposed by the agent (1 means perfect match; for details, see Appendix D). We consider the following systems: ARB, arbitrarily chosen Hamiltonian acting on 3 spins; HEIS, 1D Heisenberg model with 10 spins; TFI, 1D transverse field Ising model with 10 spins. The letters in brackets denote whether the agent can vary the value of a field parameter (f), of a coupling parameter (c), or observe only two spins of a larger chain (h). (b) Hamiltonian discovery by measuring the ground-state expectation values of spin operators $\hat{M}$ defined by the agent. Some examples are shown on the bottom left. Right: fidelity scaled by the number of spins N between the ground state of the Hamiltonian predicted by the agent after the exploration and the true ground state. We consider the following systems: TFI, 1D transverse field Ising model with 10 spins; HEIS, 2D Heisenberg model with 9 spins; CI, 1D cluster Ising model with three-body interactions and 10 spins; ARB, arbitrarily chosen, translationally invariant Hamiltonian acting on 10 spins. Letters denote the same as in (a). Additionally, the agent can sometimes vary the number of spins in the chain, as denoted by (n).

conjectures values of their parameters, and runs simulations to confirm or reject these hypotheses. The agent is also cautious enough to keep an open mind (as per its instructions) and lists possibilities that have not yet been refuted, like more subtle, longer-range couplings.

In an extension to both the dynamical and ground-state setting, we let the agent control one or several parameters whose meaning it is unaware of (mimicking closely a potential experimental scenario of exploration), letting it discover a whole family of Hamiltonians. We find that the agent can also successfully solve this extended task, for example, by creating visualizations of expectation values against the strength of the tunable parameters (for an example, see Supplemental Material, S3F [40]).

Overall, as demonstrated in Fig. 6, the agent also performs impressively well in discovering models in quantum many-body physics, again without any fine-tuned instructions or a prescribed exploration loop.

D. Experiments with noise

A common challenge in recovering physical models from experimental data is the presence of measurement errors. For classical systems, measurement errors are often modeled as Gaussian noise, which we add to the mechanical and field systems. In quantum mechanics, experimental observables cannot be measured exactly. Instead, they are estimated from a finite number of single-shot measurements. Therefore, we investigate the agent's performance under a finite number of measurement shots per expectation value. We run additional experiments on one prototypical system from each physical scenario presented above, using two different noise levels. We find that while SCIEXPLORER's predictions become less consistent as noise increases, it can recover highly accurate models even under significant measurement noise for all the considered systems in at least one of six attempts per system (see Fig. 7).

E. Common failure modes

As demonstrated above, the general knowledge and reasoning capabilities of state-of-the-art LLMs are impressive. However, they are by no means perfect and still occasionally hallucinate facts, overlook important clues, or perform superficial analyses.

To potentially improve SCIEXPLORER in the future, a detailed understanding of its failure modes is desirable. Therefore, we manually analyze all failed attempts reported in this article to identify recurring mistakes made by the agent. We categorize these errors into four groups, noting that a single failed run may be assigned to multiple categories (see Fig. 8). In particular, in the quantum systems, and occasionally in the mechanical systems, the agent does not perform sufficiently diverse experiments to uncover all qualitative signatures of the system. For example, in the quantum ground-state setting, it may fail to consider enough expectation values, leading to an incorrect model that nevertheless fits all observed measurement results. Another common problem is that the agent discovers the

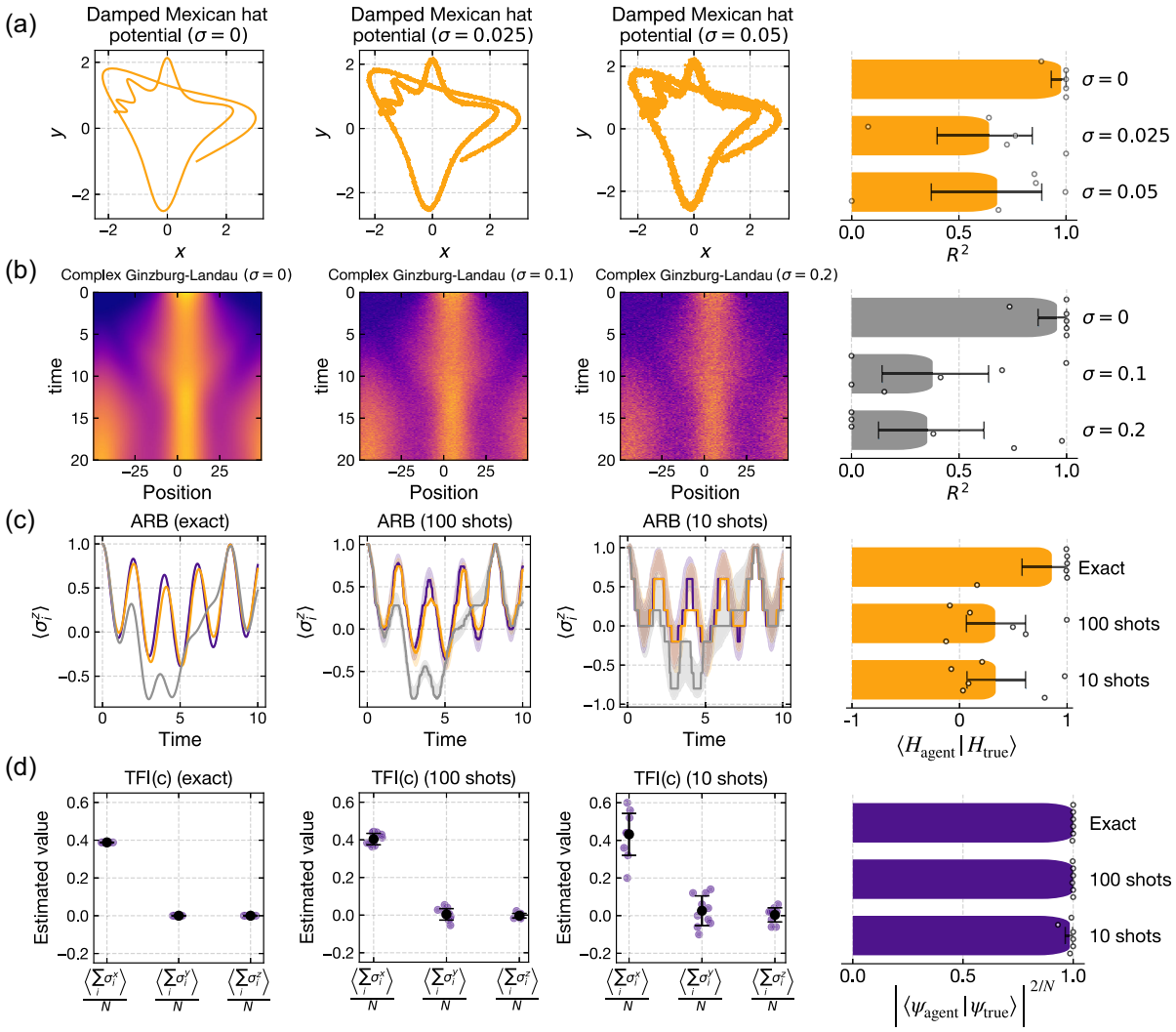

FIG. 7. Experiments with noise. While the predictions of SCIEXPLORER become less consistent for higher measurement noise levels, it successfully recovers a highly accurate model in at least one of six attempts per noise level and scenario. (a) Left: example trajectories of a particle in the damped Mexican hat potential under varying Gaussian noise levels. Right: performance of SCIEXPLORER for different noise strengths. (b) Left: evolution of the absolute square $|\phi|^2$ of a Gaussian wave packet in the complex Ginzburg-Landau scenario under varying Gaussian noise levels. Right: performance of SCIEXPLORER for different noise strengths. (c) Left: example trajectories of single-spin σ^z expectation values under the arbitrary Hamiltonian in the quantum dynamics setting. Expectation values are estimated from different numbers of single-shot measurements. Shaded areas indicate standard deviations over multiple runs. Right: performance of SCIEXPLORER for various numbers of measurement shots. (d) Left: ground-state expectation values of the total spin operators in x, y, and z direction in the transverse field Ising model with variable coupling for different numbers of measurement shots. Dots show ten examples of estimated expectation values and bars denote standard deviation. Right: performance of SCIEXPLORER for various numbers of measurement shots.

correct qualitative model structure but fails to infer the correct numerical parameters. In the mechanical and wave settings, this often results from unsuccessful parameter fits. In the quantum setting, most errors arise from simple sign errors or from missing factors of 2 in some Hamiltonian terms. In such cases, the agent fails to even consider the possibility of an overall scaling factor or sign change. When analyzing mechanical systems, the agent also frequently overlooks qualitative clues that are, in principle, contained in its analysis results, most often in visualizations. Examples include missing fast oscillations modulating an overall trend or failing to detect the breaking of translational invariance by an external potential in field systems (for an example, see Supplemental Material, S3C [40]). Most runs fail not because the agent missed an obvious clue (at least not obvious to the first author of

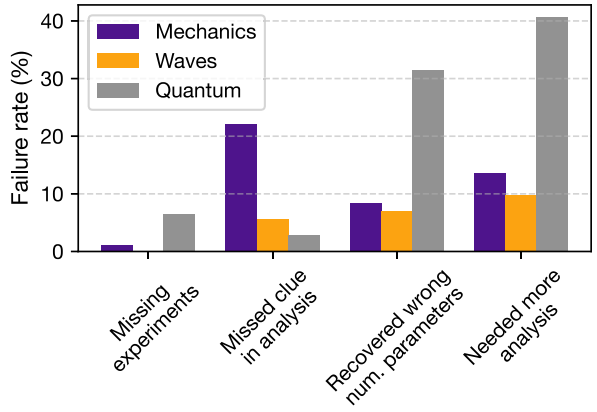

FIG. 8. Common failure modes. Failure rate per physical domain (as percentage of total runs) categorized by different reasons. Each exploration potentially contributes to multiple bars. Failure reasons were assigned through human judgment after manually analyzing all failed attempts (i.e., R^2 or overlap < 0.99). Missing experiments: the experiments run by the agent did not reveal all the qualitative information needed to extract the model. Missed clue in analysis: there was a clue in the agent's analysis results that the agent missed (typically in a plot). Recovered wrong num. parameters: agent found the correct model structure, but failed to find the correct numerical parameters (e.g., failed fits, sign errors). Needed more analysis: the agent did not miss a clear clue but instead did not implement enough analysis routines to uncover the true model.

this paper), but because its analysis was not sufficiently exhaustive to identify the correct model, leading the agent to commit prematurely to an incorrect model.

Many of these failure modes could potentially be mitigated through more sophisticated prompting (e.g., to avoid sign errors in quantum many-body systems). However, we deliberately avoided such interventions in the spirit of not introducing task- or system-specific fine-tuning. Other failure modes may only be alleviated by future LLMs with, for example, improved visual capabilities.

SCIEXPLORER relies on an LLM as its backbone and thereby inherits the LLM's intrinsic priors about useful strategies and common physical systems. Because current LLMs are mostly trained on human-generated text, SCIEXPLORER employs heuristics similar to a human physicist. While this helps to navigate the exponentially large search space of possible solutions, it also means that there exists a bias toward "likely" models. Therefore, just like for a human, recovering entirely new or uncommon models is difficult for SCIEXPLORER. However, physically relevant models of an experimental system will typically not contain completely unknown ingredients, but rather combinations of individual, in principle known, parts (e.g., different terms in a wave equation). To test SCIEXPLORER's abilities on "uncommon" systems, we included examples such as the "arbitrary potentials," and "arbitrary Hamilonian," which are uncommon combinations of known ingredients, and the "sinusoidal potential relaxation," which is a completely unphysical system. We find that the agent's success rate is lower on these systems than on more standard ones. Still, it can recover the true underlying model in the nonstandard mechanical and quantum systems in some attempts, while it finds only approximate solutions in the wave system containing an artificial "$\sin(0.1|\phi_n|^2)\phi_n$" term in the wave equation. To improve SCIEXPLORER's performance in these uncommon cases, multiple strategies could be employed in the future. In many real applications, some information about the underlying model may be available, which can easily be incorporated into SCIEXPLORER as plain text. Even just the information that one expects uncommon terms in the solution could help to avoid focusing on known models. Additionally, one could encourage SCIEXPLORER to leverage data-driven model discovery techniques such as SINDY [41] that rely less on heuristics and are therefore also less influenced by the LLM's priors. Here, SCIEXPLORER could potentially still improve the technique substantially by tailoring it to the specific system, e.g., by enforcing previously discovered symmetries.

F. Comparison to symbolic regression

In this article, we describe a general AI scientist and benchmark its capabilities in a verifiable setting, namely, discovering the model of an unknown physical system. The systems considered here are all governed by analytic expressions. Symbolic regression is a research field concerned with automatically extracting such expressions from data [45]. A human attempting to solve the same task used to benchmark SCIEXPLORER (model discovery) would typically use symbolic regression techniques as part of its discovery process. These techniques can be categorized into three classes. First, standard regression assumes a known analytical expression in which only numerical constants must be adjusted. Second, sparse regression techniques fit a linear combination of ansatz functions to the data (e.g., SINDY [41,46] for ordinary differential equations and PDEFIND [47] for partial differential equations). By including a loss term that encourages sparsity in the resulting prefactors, irrelevant ansatz functions can often be eliminated, leaving only the relevant terms. Finally, some techniques do not require a predefined ansatz and instead navigate the exponentially large space of analytical expressions (e.g., AIFEYNMAN [48,49] for functions).

When solving model discovery tasks, SCIEXPLORER improves upon these techniques in several qualitative ways. SCIEXPLORER can control experiments, thereby integrating an active learning component. Moreover, SCIEXPLORER has physical intuition like a human researcher, which it uses to navigate the exponentially large space of possible models.

SCIEXPLORER operates with minimal prior information and does not require a predefined ansatz. However, when additional information about the model is known, it can easily be incorporated as plain text. Finally, SCIEXPLORER not only addresses symbolic regression tasks but can also tackle more general challenges, such as program discovery. When appropriate, SCIEXPLORER uses symbolic regression techniques itself. In fact, after inferring a suitable ansatz from the qualitative signatures of an unknown system, SCIEXPLORER often employs sparse regression as part of its discovery process, implementing it from scratch.

We provide a quantitative comparison between SCIEXPLORER and the symbolic regression techniques SINDY, AIFEYNMAN, and PDEFIND in Appendix H. We find that even when a user manually carries out the additional steps required by these techniques (selecting experiments, hyperparameters, and ansatz functions), SCIEXPLORER discovers more accurate and interpretable models.

IV. CONCLUSION

We have demonstrated that SCIEXPLORER can successfully address open-ended, research-style tasks by interacting with physical systems through repeated cycles of experimentation and analysis. The LLM at the core of SCIEXPLORER leverages broad knowledge across diverse areas of physics, such as classical mechanics, wave dynamics, and quantum systems, to autonomously characterize systems in a single-shot manner, i.e., without example solutions and with only minimal task- or system-specific instructions. Equipped with generic, code-based analysis and visualization tools, the agent can implement an active learning process to solve tasks such as discovering equations of motion from observed dynamics and recovering Hamiltonians, or even Hamiltonian families, from expectation values.

In physics, many modern experiments are controlled through code-based interfaces, which makes SCIEXPLORER directly applicable in experimental settings such as complex fluids, cold atomic gases, strongly correlated electronic and spin systems, or quantum simulators. While model discovery remains highly relevant in these systems, SCIEXPLORER could, for example, also map out phase diagrams or optimize control tasks in unknown or partially known systems.

The iterative process of experiment selection, analysis, and hypothesis generation is not unique to physics but lies at the heart of the natural sciences more broadly. Because LLMs possess substantial knowledge across scientific domains such as chemistry [50,51] and biology [52,53], and because SCIEXPLORER requires no task-specific fine-tuning, applying this framework to other scientific fields is a natural next step, for example, to study predator-prey dynamics in biology or chemical nonequilibrium dynamics.

ACKNOWLEDGMENTS

This research is part of the Munich Quantum Valley, which is supported by the Bavarian state government with funds from the Hightech Agenda Bayern Plus.

DATA AVAILABILITY

The complete SCIEXPLORER framework, including documentation on adding new experiments and analysis tools, is available as a PYTHON package on GitHub [54]. Additionally, we provide all code used to run the explorations presented in this manuscript, logs of all explorations, and tools to print or summarize those explorations in a second GitHub repository [55].

APPENDIX A: DETAILS ON THE SCIEXPLORER

After supplying the generic and rather brief system prompt (identical throughout all experiments), the LLM agent receives a short description of the problem class (e.g., a dynamical system or a spin system) and the task (e.g., discovering equations of motion or the Hamiltonian). In addition, as for any application of agentic tool use, the LLM has access to the documentation of the tools it can access. To facilitate the efficient integration of new analysis tools and experiments into the SCIEXPLORER framework, we automatically generate application programming interface (API)-compatible documentation by parsing the docstrings of tool methods. We list all such inputs to the LLM (prompts, problem class description, task description, tool documentation) in Appendix F and Supplemental Material [40].

In the following steps of the automated conversation, the agent receives only a hint about the number of remaining conversation steps and tool calls, and a reminder to heed the system prompt.

In all explorations, the agent has access to two generic core tools: `execute_code` for executing generic PYTHON code and `plot_from_code` for creating MATPLOTLIB plots.

The results of each tool call are saved to the external memory under a result label, which the agent is automatically asked to specify to enable descriptive names. These results are then accessible as local variables in the `execute_code` and `plot_from_code` tools. The tool results are communicated to the agent as plain text for arrays with fewer than 10 entries. Otherwise, only the shape and type are returned to prevent excessively large context windows. If a tool call produces an image, it is returned to the agent for visual inspection.

The agent can choose to finish the exploration at any time by calling the `save_result` tool. It is then asked to announce its result in the form of PYTHON code that defines the solution hypothesized by the agent (see below).

After a given run of the agent, we save the entire conversation for further inspection.

APPENDIX B: MECHANICAL SYSTEMS

We use a differential equation solver to solve the equations of motion for the experimental runs of the system that is unknown to the agent. To run an experiment, the agent can specify initial conditions, a start and end time, and the time resolution to receive generalized coordinates and velocities of the system evaluated on a time grid using the `observe_experiment` tool. We allow the agent to observe up to 5 experiments within a single tool call to reduce the steps needed to explore the system.

When the agent completes its explorations, it submits the hypothesized model in the form of a PYTHON function `predict_experiment` that should reproduce the `observe_experiment` function of the unknown system, which typically includes code to solve an ordinary differential equation (ODE) and its right-hand side (rhs). However, we never inform the agent that the system is governed by an ODE, and it could, in principle, submit PYTHON code implementing any model.

To evaluate the performance of the agent, we require a single continuous value describing how "close" the agent's prediction is to the truth. We estimate the coefficient of determination,

$$R^2(\vec{x},\vec{y}) = 1 - \frac{\sum_i (y_i - x_i)^2}{\sum_i (\bar{x} - x_i)^2},$$

between the true ($\vec{x}$, with mean $\bar{x}$) velocities and accelerations of the system (given by the rhs of its ODE) and their prediction by the agent's `predict_experiment` function ($\vec{y}$, estimated from two consecutive data points by choosing a small time resolution). We sample 1000 random position-velocity-time combinations and then compute R^2 for each coordinate and velocity separately and average them to obtain a single metric. The R^2 value can range from $-\infty$ (bad fit) over 0 [as good as predicting $\vec{y} = \text{mean}(\vec{x})$] to 1 (perfect fit). Therefore, we lower bound this value by zero to enable reasonable averaging of multiple runs. For partially observable systems, this method is not applicable, since correctly predicting the evolution of the hidden particles is a crucial part of the task and we cannot just sample random positions in phase space. Instead, we simulate 100 trajectories with both the true dynamics and the agent's prediction with random initial conditions of the observed particles. Finally, we compute the mean R^2 of the trajectory of the observed particle between the true and predicted dynamics, i.e., we ask the agent to reproduce the observed dynamics, but do not require it to correctly reproduce the hidden dynamics.

APPENDIX C: FIELDS AND WAVES

We set up a split-step integrator for the evolution of a complex field ϕ on a 1D lattice, providing the experimental results for a hidden evolution equation. The agent specifies start and end time, time resolution, and initial conditions using JAX code [56] (`observe_experiment` tool). As above, the agent saves its model in the form of a `predict_experiment` function which should reproduce the experimental results.

To evaluate the performance of the agent, we proceed as follows. We first run the wave evolution for different reasonable initial conditions (multiple Gaussian wave packets with and without momentum and with different amplitudes) to obtain a large set of physically relevant field or wave functions $\phi(x,t)$. We then numerically calculate $d\phi(x,t)/dt$ for all initial conditions and all times t under both the true model and the agent's prediction and calculate the R^2 value between them.

APPENDIX D: QUANTUM MANY-BODY PHYSICS SYSTEMS

We consider a system of N spin-1/2 degrees of freedom (qubits). We predefine the Pauli matrices $\hat{S}_x^j$, $\hat{S}_y^j$, and $\hat{S}_z^j$. These matrices can be used to define Hamiltonians or operators to observe. An example of such code looks like the following, for the transverse quantum Ising model:

```
H=0 * Sz [0]
for j in range(10):
    H+=Sx[j] @ Sx[j+1]
for j in range(10):
    H-=0.5*Sz[j]
```

Depending on the scenario, the agent can either retrieve ground-state expectation values of previously specified operators (using the `set_operator` and `observe_experiment` tools) or observe the dynamics of a (hidden) Hamiltonian by specifying an initial product state of the spins (`set_blochvectors` and `observe_experiment` tools). In the latter case, we also consider a more challenging scenario in which the agent is only allowed to observe and initialize a selected subset of spins, while all other spins are set to a default state in each experimental run.

For both ground-state and time-dependent experiments, we also consider scenarios where the Hamiltonian depends on one or more tunable parameters (as in $\hat{H} = \sum_j \hat{\sigma}_j^x \hat{\sigma}_{j+1}^x - A \sum_j \hat{\sigma}_j^z$), which the agent can set before running an experiment. This allows model discovery of a whole parametrized family of Hamiltonians. In a variant of this scenario, the agent may also choose the number N of spins in the experimental system, e.g., to explore finite-size effects in translationally invariant models.

To save its result, the agent is asked to provide the Hamiltonian of the system in the format described above. In the ground-state experiments, we evaluate the agent's performance by calculating the fidelity per spin between

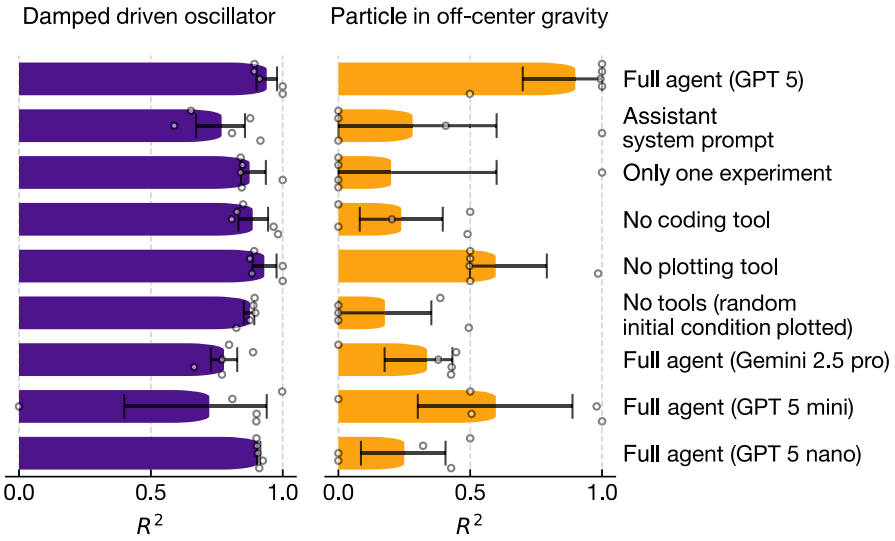

FIG. 9. Ablation studies. Mean performance of the agent in the damped driven oscillator and the particle in off-center gravity task with different modifications. Error bars are 95% bootstrap confidence intervals. Full agent (GPT 5): original SCIEXPLORER with GPT 5 [39] as an LLM backbone. Assistant system prompt: changed system prompt to "You are a helpful assistant." Only one experiment: agent can observe only one trajectory. No coding tool: agent cannot execute arbitrary PYTHON code, but only plot experimental results. No plotting tool: the agent cannot generate visualizations. Instead, down-sampled experimental trajectories are stored in the agent's context. No tools: the agent cannot use any tools. In the beginning, it is instead provided with time evolution and phase space plots of ten trajectories with random initial conditions. Full agent (Gemini 2.5 pro): original SCIEXPLORER with Gemini 2.5 pro [42] as an LLM backbone. Full agent (GPT 5 mini): original SCIEXPLORER with GPT 5 mini as an LLM backbone. Full agent (GPT 5 nano): original SCIEXPLORER with GPT 5 nano as an LLM backbone.

the ground state predicted by the agent and the true ground state, i.e., $|\langle\psi_{\text{agent}}|\psi_{\text{true}}\rangle|^{2/N}$, where N is the number of spins in the system. This quantity is defined so as to become system-size independent in the thermodynamic limit of large N. For the dynamics tasks, we first shift the Hamiltonians by $H_i' = H_i - \text{tr}(H_i)/2^N$ and then compute the scalar product of these shifted Hamiltonians as $\text{tr}[(H'_{\text{true}})^\dagger H'_{\text{agent}}]/\max(||H'_{\text{true}}||, ||H'_{\text{agent}}||)^2$, where $||\cdot||$ denotes the Frobenius norm $||A|| = \sqrt{\text{tr}[A^\dagger A]}$. Taking the maximum of both norms in the denominator ensures that multiplying the predicted Hamiltonian by a scalar, thereby effectively rescaling time, results in a lower quality score. For tasks with tunable parameters, we average the quality score over 100 random values of these parameters.

APPENDIX E: FURTHER ANALYSIS OF THE SCIEXPLORER

To assess which components of the SCIEXPLORER are essential, we performed ablation experiments on two representative systems: the damped driven oscillator and the particle in off-center gravity (Fig. 9). The relative importance of individual components differs between these systems. A structured system prompt appears to improve performance in both cases, whereas access to multiple experiments and visualization tools is particularly critical in the gravity system. Without the coding tool, the agent sometimes recovers the correct qualitative model but consistently fails to identify accurate numerical parameters. Without any access to tools and instead provided with plots of trajectories under random initial conditions (matching the number of experiments selected by the full agent), the LLM is unable to identify accurate models.

Finally, we observe substantial variation across LLMs: GPT 5 [39] dramatically outperforms Gemini 2.5 pro [42], which fails to recover the true models even when granted access to the full SCIEXPLORER framework. Similarly, GPT 5 nano fails to find the true models in all attempts. GPT 5 mini performs slightly better, recovering the true model in one of five attempts per system.

APPENDIX F: PROMPTS

Throughout all tasks and physics scenarios, we use the same generic and relatively brief system prompt, essentially instructing the LLM to act like a cautious scientist:

Start of system prompt.

(i) Act as a computational physicist dedicated to thoroughly resolving the user's query through careful planning, hypothesis generation, and iterative verification.

(ii) In your first message, create a comprehensive plan to solve the users query. Include an extensive list of candidate hypotheses.

(iii) Initially, conduct at least 5 different experiments spanning the entire range of reasonable initial con-

ditions. Make sure to cover also extreme cases. Then, create informative plots of your experimental results.

(iv) Withhold any final answer until you are sure that no further improvements of your hypothesis are possible.

(v) Before submitting your final answer, simulate your proposed model using the same initial conditions as in your experiments, and compare the results. Only submit your final answer if the simulation results closely match your experimental data.

(vi) If you have run a tool but still need to extract the results (e.g. via visualization), just briefly explain what tool you will call next to extract the results.

(vii) "System prompt" Otherwise, at each step, you must answer the following questions:

(1) What can you learn from the new tool results (if any)?

(2) Which old hypotheses still fit your data?

(3) Which new hypotheses might be worthwhile considering?

End of system prompt.

Even though we ask the agent to answer the questions above at each step, it does not reliably do so. However, summaries of the internal thinking process of the agent indicate that it considers the questions internally. We use the following intermediate user message after each response from the agent:

Start of intermediate message.

Keep solving the problem while following your system prompt.

End of intermediate message.

APPENDIX G: RESOURCE REQUIREMENTS

In Fig. 10, we show the run-time for all explorations of the mechanical systems which typically ranges from a few minutes to over 1.5 h.

As shown in Fig. 11, running SCIEXPLORER with a state-of-the-art LLM (in this case, GPT 5) costs around two USD per exploration on average. Although higher than typical specialized machine learning algorithms (e.g., AIFEYNMAN or SINDY), the cost is small compared to the hourly salary of a Ph.D. student in theoretical physics, who would be needed to fully replicate SCIEXPLORER's capabilities.

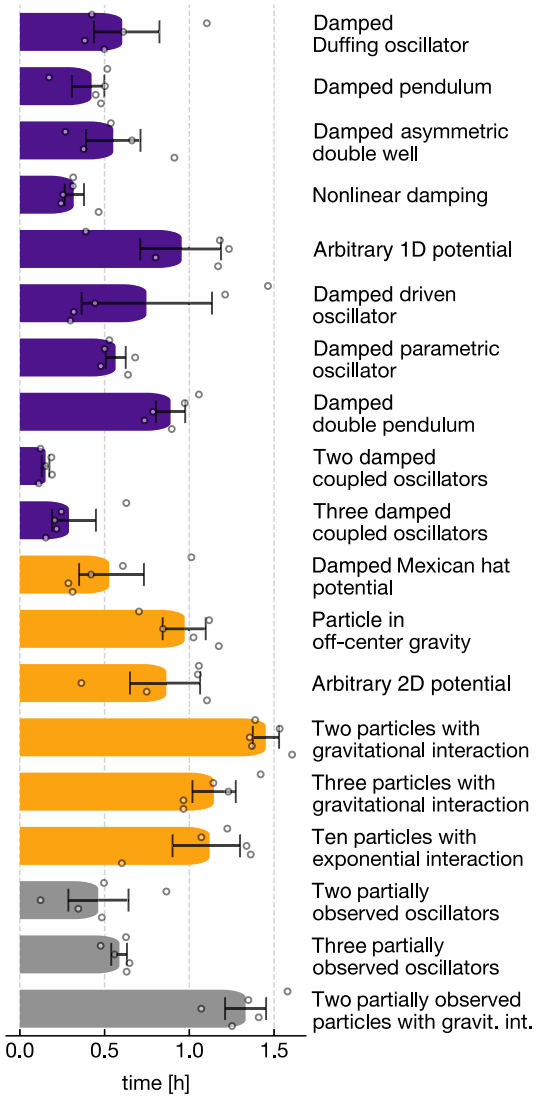

FIG. 10. Run-time analysis. Run-time of the classical mechanics explorations shown in Fig. 3 of the main text in hours. Run times range from a few minutes to over 1.5 h.

APPENDIX H: COMPARISON TO STANDARD SYMBOLIC REGRESSION TECHNIQUES

In the following, we compare SCIEXPLORER to the standard symbolic regression techniques SINDY and

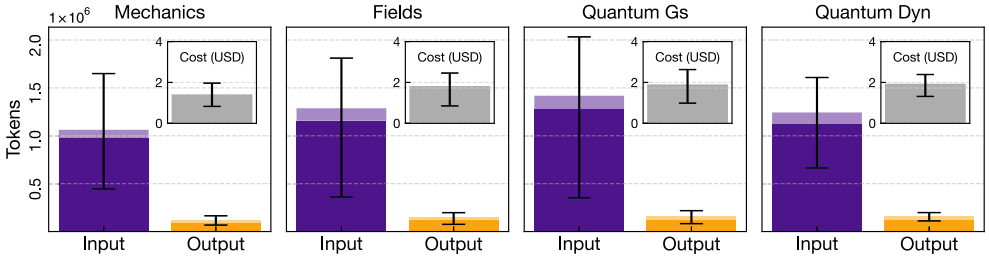

FIG. 11. Token usage and cost. Average tokens used in a single exploration for the different domains. Violet bars are input tokens (dark, cached from previous requests; bright,new input), yellow bars are output tokens (dark, thinking tokens; bright, text output tokens). The inset shows the average API costs per exploration for GPT 5. Error bars denote 25 and 75 percentiles across all experiments in the domain.

AIFEYNMAN2 for the mechanical systems and PDEFIND for field systems.

1. SINDY and AIFEYNMAN2 for mechanical systems

SINDY [41] is a technique specifically developed to discover ordinary differential equations from data. It requires the user to propose a set of ansatz functions that may appear in the right-hand side of the system's ODE. It then uses sparse linear regression on the system's acceleration to find the numerical prefactors of these terms. The loss function combines the mean squared error between the true and predicted acceleration with an L1 regularization term on the prefactors of the terms in the predicted rhs. This L1 loss is weighted by a hyperparameter α, with higher values promoting sparser solutions. We use the implementation PySINDy [46] with a sequentially thresholded Ridge regression optimizer, iteratively setting values below 0.01 to zero. The threshold is chosen such that no constants present in the true models are eliminated (the smallest one being 0.043). The performance of SINDY strongly depends on the sparsity penalty α. Therefore, we select the optimal $\alpha \in [0.01, 0.05, 0.1]$ for each experimental system. As the ansatz, we consider two different function libraries. First, we consider a purely polynomial library containing polynomials up to degree three (polyn). Second, we add sin and cos functions with frequencies scanned from 0.5 to 6 in steps of 0.5 to the polynomial library. To even further reduce the difficulty for SINDY, we also include the exact drive frequency for the driven oscillators $\omega = 1.551$ (polyn + trig). These two libraries are chosen since they provide a good trade-off between the number of ansatz functions and the number of systems that could be fitted by them. We fit ten random trajectories of the system with 20001 time points each. We compare the fit quality and run-time of SINDY with SCIEXPLORER in Fig. 12. Although SINDY is much faster than SCIEXPLORER, it can recover only a small subset of solutions with high accuracy. Even some systems whose rhs could be exactly fitted with the provided function libraries (marked by stars) are not recovered.

AIFEYNMAN2 is a symbolic regression technique that fits a neural network to the provided data and uses physics-inspired methods, such as testing for translational invariance and symmetries, to simplify the regression task. AIFEYNMAN2 can recover only the analytical expression of a function rather than an ODE. Therefore, we test how well it can fit the rhs of the mechanical systems' ODEs. We provide the value of the exact rhs at 200 010 points to AIFEYNMAN2, which would normally have to be estimated from data, thereby avoiding additional errors that would arise from estimation. A known problem of AIFEYNMAN2 is the divergence of its neural network during training. To alleviate this problem, we ran AIFEYNMAN2 twice for each system: once with the direct experimental data and once with the data normalized to zero mean and unit standard deviation, selecting the better of the two results. We find that AIFEYNMAN2 can accurately recover only the simplest systems considered (see Fig. 12).

Both SINDY and AIFEYNMAN2 have hyperparameters and require choices such as the selection of ansatz functions. Our experiments show that even with considerable human effort to choose suitable parameters and algorithms, these techniques do not match SCIEXPLORER in fit quality. However, we do not claim that SINDY is incapable of finding the correct model, provided the correct hyperparameters and a reduced set of suitable ansatz functions. In fact, SCIEXPLORER often employs SINDY-style techniques after inferring an appropriate ansatz from the qualitative signatures of the system's dynamics.

2. PDEFIND for field systems

PDEFIND [47] generalizes SINDY to partial differential equations. As above, we use the PySINDy implementation with a sequentially thresholded Ridge regression optimizer. PDEFIND's performance strongly depends on its hyperparameters. Therefore, we scan both the L1 loss $\alpha \in [0.01, 0.05, 0.1]$ and the threshold $\in [0.00001, 0.0001, 0.001, 0.01, 0.1]$. We use two function libraries. An exact library tailored to each model, containing only the functions necessary to fit the true underlying model, along with

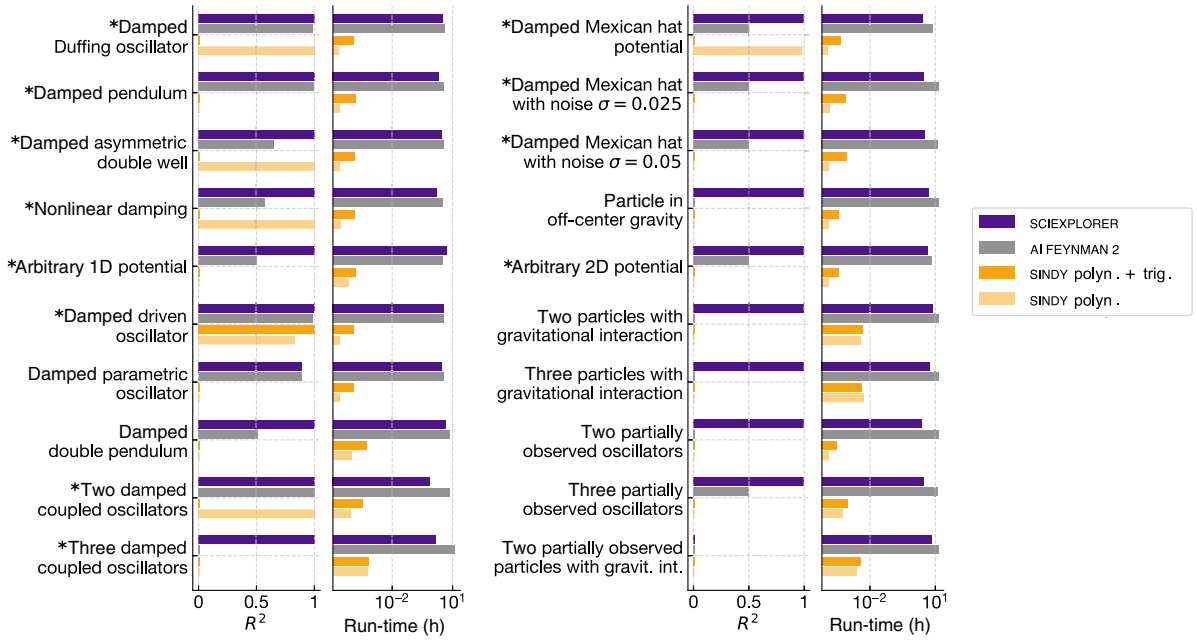

FIG. 12. Comparison to SINDY and AIFEYNMAN2. Performance and run-time of SCIEXPLORER (best of 5), AIFEYNMAN2, and SINDY with two different function libraries (see main text). Stars mark systems that could, in principle, be exactly recovered with the function libraries provided to SINDY. SINDY requires substantially less run-time but cannot recover many of the models SCIEXPLORER finds. AIFEYNMAN2 requires a similar run-time but also falls significantly short of SCIEXPLORER in fit quality. Missing bars indicate an $R^2 < 0$ or, in the case of AIFEYNMAN2, that no result was produced due to the divergence of the neural network trained during its discovery attempt. The run-time of SCIEXPLORER is the sum of the five independent explorations we ran per system. It could be further reduced by parallelizing these runs. Similarly, AIFEYNMAN2 could be sped up by parallelizing over the individual dimensions of the ode and by running on a GPU.

the correct order of spatial derivatives (exact), and a fixed library capable of fitting all models (all). We fit trajectories of four representative initial conditions, using numerical parameters slightly adapted from those used to assess fit quality. Each run of PDEFIND takes only tens of seconds. However, it frequently fails to recover a well-fitting model and often produces models with more terms than required, reducing their interpretability (see Fig. 13). We stress that we considered a scenario that already requires substantial additional human effort compared to running SCIEXPLORER, i.e., identifying a small set of accurate ansatz functions.

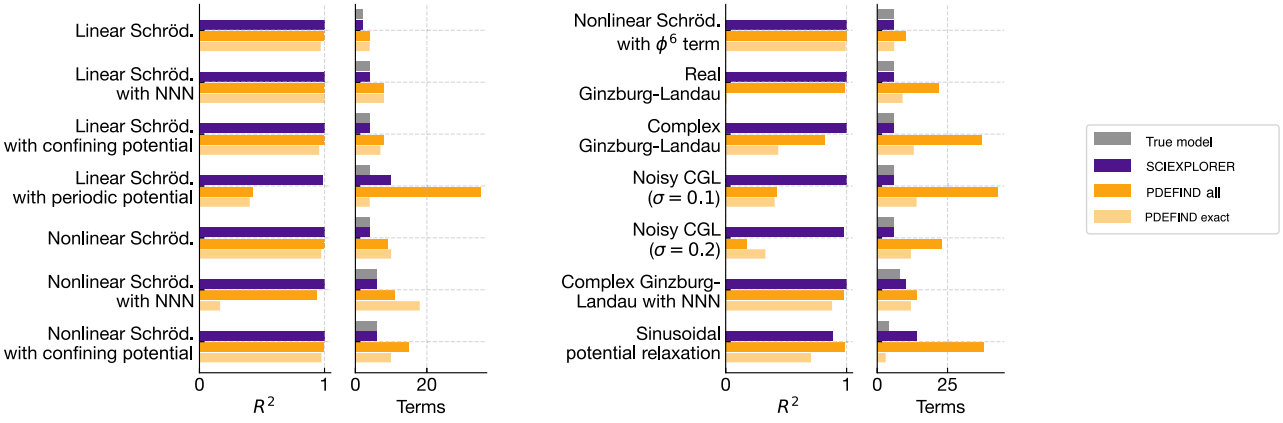

FIG. 13. Comparison to PDEFIND. Performance and number of terms in the recovered partial differential equation for SCIEXPLORER (best of 5) and PDEFIND for a general ansatz library (all) and a library specialized for each system (exact). SCIEXPLORER consistently finds more accurate models with fewer terms than PDEFIND.

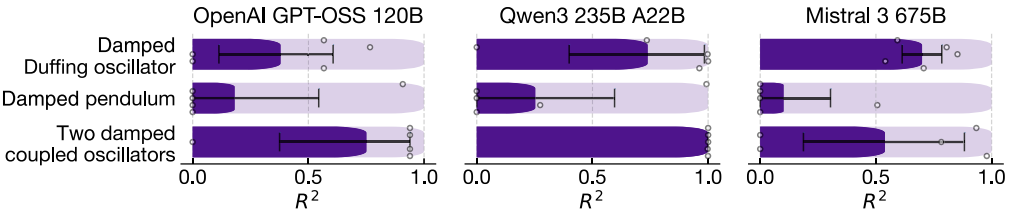

FIG. 14. Experiments with open-source models. Performance of three open-source models on selected mechanical systems. All models struggle even with the easier tasks considered in the main text. Shaded bars indicate the performance of GPT 5, which finds perfect models in all five attempts per system.

APPENDIX I: EXPERIMENTS WITH OPEN-SOURCE MODELS

The experiments in the main text use GPT 5 as the state-of-the-art backbone of SCIEXPLORER. In the long term, it would be desirable to base SCIEXPLORER on open-source models. To this end, we integrate the OpenAI Chat API, which is compatible with many open-source models, into SCIEXPLORER. We present initial experiments with the open-source models OpenAI GPT-OSS 120B (supports reasoning, no vision), Qwen3 235B A22B (supports reasoning, no vision), and Mistral 3 675B (no reasoning, supports vision). Unfortunately, these models already struggle with even the simplest tasks presented in the main text (see Fig. 14). A common problem of GPT-OSS is not following the user's instruction, e.g., calling the `save_result` in the first step without any information. Mistral 3 can visualize trajectories but often hallucinates image content. While it can write code to fit experimental trajectories, it typically proposes overly simple models and uses inaccurate initial guesses for the fits. Qwen3 performs better than the other open-source models and occasionally employs advanced techniques, such as sparse linear regression of the acceleration, to recover the correct model. Nevertheless, it still often hallucinates, for instance, submitting models with incorrect numerical constants even when its fits returned the true values. These results suggest that developing a competitive multimodal LLM with robust reasoning capabilities should be a priority for the open-science community.

[1] Alexey A. Melnikov, Hendrik Poulsen Nautrup, Mario Krenn, Vedran Dunjko, Markus Tiersch, Anton Zeilinger, and Hans J. Briegel, *Active learning machine learns to create new quantum experiments*, Proc. Natl. Acad. Sci. U.S.A. **115**, 1221 (2018).

[2] Nathan J. Szymanski, Bernardus Rendy, Yuxing Fei, Rishi E. Kumar, Tanjin He, David Milsted, Matthew J. McDermott, Max Gallant, Ekin Dogus Cubuk, Amil Merchant, Kim *et al.*, *An autonomous laboratory for the accelerated synthesis of novel materials*, Nature (London) **624**, 86 (2023).

[3] Xiaobo Li, Yu Che, Linjiang Chen, Tao Liu, Kewei Wang, Lunjie Liu, Haofan Yang, Edward O. Pyzer-Knapp, and Andrew I. Cooper, *Sequential closed-loop Bayesian optimization as a guide for organic molecular metallophotocatalyst formulation discovery*, Nat. Chem. **16**, 1286 (2024).

[4] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko *et al.*, *Highly accurate protein structure prediction with AlphaFold*, Nature (London) **596**, 583 (2021).

[5] Amil Merchant, Simon Batzner, Samuel S. Schoenholz, Muratahan Aykol, Gowoon Cheon, and Ekin Dogus Cubuk, *Scaling deep learning for materials discovery*, Nature (London) **624**, 80 (2023).

[6] Annabelle Bohrdt, Christie S. Chiu, Geoffrey Ji, Muqing Xu, Daniel Greif, Markus Greiner, Eugene Demler, Fabian Grusdt, and Michael Knap, *Classifying snapshots of the doped Hubbard model with machine learning*, Nat. Phys. **15**, 921 (2019).

[7] Hugo Dalla-Torre, Liam Gonzalez, Javier Mendoza-Revilla, Nicolas Lopez Carranza, Adam Henryk Grzywaczewski, Francesco Oteri, Christian Dallago, Evan Trop, Bernardo P. de Almeida, Hassan Sirelkhatim *et al.*, *Nucleotide transformer: Building and evaluating robust foundation models for human genomics*, Nat. Methods **22**, 287 (2025).

[8] Till Richter, Mojtaba Bahrami, Yufan Xia, David S. Fischer, and Fabian J. Theis, *Delineating the effective use of self-supervised learning in single-cell genomics*, Nat. Mach. Intell. **7**, 68 (2025).

[9] Miles Cranmer, Alvaro Sanchez-Gonzalez, Peter Battaglia, Rui Xu, Kyle Cranmer, David Spergel, and Shirley Ho, *Discovering symbolic models from deep learning with inductive biases*, in *Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS '20* (Curran Associates Inc, Red Hook, NY, 2020), ISBN 9781713829546, https://proceedings.neurips.cc/paper_files/paper/2020/file/c9f2f917078bd2db12f23c3b413d9cba-Paper.pdf.

[10] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa, *Large language models are zero-shot reasoners*, Advances in Neural Information Processing Systems (2022), Vol. **35**, pp. 22199–22213, https://proceedings.neurips.cc/paper_files/paper/2022/hash/8bb0d291acd4acf06ef112099c16f326-Abstract-Conference.html.

[11] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millicah, Malcolm Reynolds *et al.*, *Flamingo: A*

visual language model for few-shot learning, in *Proceedings of the 36th International Conference on Neural Information Processing Systems, NIPS '22* (Curran Associates Inc, Red Hook, NY, 2022), ISBN 9781713871088.

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

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

[14] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu, *A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions*, ACM Trans. Inf. Syst. **43**, 42 (2025).

[15] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao, *React: Synergizing reasoning and acting in language models*, in *Proceeding sof the International Conference on Learning Representations* (2023), https://openreview.net/forum?id=WE_vluYUL-X.

[16] 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 *Proceedings of the 38th International Conference on Neural Information Processing Systems, NIPS '24* (Curran Associates Inc, Red Hook, NY, 2025), ISBN 9798331314385, https://proceedings.neurips.cc/paper_files/paper/2024/file/5a7c947568c1b1328ccc5230172e1e7c-Paper-Conference.pdf.

[17] 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:2408.06292.

[18] Daniil A. Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes, *Autonomous chemical research with large language models*, Nature (London) **624**, 570 (2023).

[19] Andres M. Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D. White, and Philippe Schwaller, *Augmenting large language models with chemistry tools*, Nat. Mach. Intell. **6**, 525 (2024).

[20] Yixiang Ruan, Chenyin Lu, Ning Xu, Yuchen He, Yixin Chen, Jian Zhang, Jun Xuan, Jianzhang Pan, Qun Fang, Hanyu Gao *et al.*, *An automatic end-to-end chemical synthesis development platform powered by large language models*, Nat. Commun. **15**, 10160 (2024).

[21] Yusuf H. Roohani, Andrew H. Lee, Qian Huang, Jian Vora, Zachary Steinhart, Kexin Huang, Alexander Marson, Percy Liang, and Jure Leskovec, *Biodiscoveryagent: An AI agent for designing genetic perturbation experiments*, in *Proceedings of the Thirteenth International Conference on Learning Representations* (2025), https://openreview.net/forum?id=HAwZGLcye3.

[22] Yuanhao Qu, Kaixuan Huang, Ming Yin, Kanghong Zhan, Dyllan Liu, Di Yin, Henry C. Cousins, William A. Johnson, Xiaotong Wang, Mihir Shah *et al.*, *Crispr-GPT for agentic automation of gene-editing experiments*, Nat. Biomed. Eng. **10**, 245 (2025).

[23] Haining Pan, Nayantara Mudur, William Taranto, Maria Tikhanovskaya, Subhashini Venugopalan, Yasaman Bahri, Michael P. Brenner, and Eun-Ah Kim, *Quantum many-body physics calculations with large language models*, Commun. Phys. **8**, 49 (2025).

[24] Tuan Dung Nguyen, Yuan-Sen Ting, Ioana Ciucă, Charlie O'Neill, Ze-Chang Sun, Maja Jabłońska, Sandor Kruk, Ernest Perkowski, Jack Miller, Jason Li *et al.*, *AstroLLaMA: Towards specialized foundation models in astronomy*, arXiv:2309.06126.

[25] Xiaoxuan Wang, Ziniu Hu, Pan Lu, Yanqiao Zhu, Jieyu Zhang, Satyen Subramaniam, Arjun R. Loomba, Shichang Zhang, Yizhou Sun, and Wei Wang, *SciBench: Evaluating college-level scientific problem-solving abilities of large language models*, arXiv:2307.10635.

[26] Shi Qiu, Shaoyang Guo, Zhuo-Yang Song, Yunbo Sun, Zeyu Cai, Jiashen Wei, Tianyu Luo, Yixuan Yin, Haoxu Zhang, Yi Hu *et al.*, *PHYBench: Holistic evaluation of physical perception and reasoning in large language models*, arXiv:2504.16074.

[27] Xinyu Zhang, Yuxuan Dong, Yanrui Wu, Jiaxing Huang, Chengyou Jia, Basura Fernando, Mike Zheng Shou, Lingling Zhang, and Jun Liu, *Physreason: A comprehensive benchmark towards physics-based reasoning*, arXiv:2502.12054.

[28] Minyang Tian, Luyu Gao, Shizhuo Dylan Zhang, Xinan Chen, Cunwei Fan, Xuefei Guo, Roland Haas, Pan Ji, Kittithat Krongchon *et al.*, *SciCode: A research coding benchmark curated by scientists*, in *Advances in Neural Information Processing Systems*, edited by A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (Curran Associates, Inc., Red Hook, New York, 2024), Vol. 37, pp. 30624–30650, https://proceedings.neurips.cc/paper_files/paper/2024/file/36850592258c8c41cecdaa3dea5ff7de-Paper-Datasets_and_Benchmarks_Track.pdf.

[29] Sirui Lu, Zhijing Jin, Terry Jingchen Zhang, Pavel Kos, J Ignacio Cirac, and Bernhard Schölkopf, *Can theoretical physics research benefit from language agents?*, arXiv:2506.06214.

[30] Kristian G. Barman, Sascha Caron, Emily Sullivan, Henk W. de Regt, Roberto Ruiz de Austri, Mieke Boon, Michael Färber, Stefan Fröse, Faegheh Hasibi, Andreas Ipp *et al.*, *Large physics models: Towards a collaborative approach with large language models and foundation models*, arXiv:2501.05382.

[31] Ziqi Wang, Hongshuo Huang, Hancheng Zhao, Changwen Xu, Shang Zhu, Jan Janssen, and Venkatasubramanian Viswanathan, *DREAMS: Density functional theory based research engine for agentic materials simulation*, arXiv:2507.14267.

[32] Licong Xu, Milind Sarkar, Anto I. Lonappan, Íñigo Zubeldia, Pablo Villanueva-Domingo, Santiago Casas, Christian Fidler, Chetana Amancharla, Ujjwal Tiwari, Adrian Bayer *et al.*, *Open source planning & control system with language agents for autonomous scientific discovery*, arXiv:2507.07257.

[33] Santiago Casas, Christian Fidler, Boris Bolliet, Francisco Villaescusa-Navarro, and Julien Lesgourgues, *CLAPP: The class LLM agent for pair programming*, arXiv:2508.05728.

[34] Baohua Zhang, Xin Li, Huangchao Xu, Zhong Jin, Quansheng Wu, and Ce Li, *Topomas: Large language model driven topological materials multiagent system*, arXiv:2507.04053.

[35] Darui Lu, Jordan M. Malof, and Willie J. Padilla, *An agentic framework for autonomous metamaterial modeling and inverse design*, arXiv:2506.06935.

[36] Ankita Sharma, YuQi Fu, Vahid Ansari, Rishabh Iyer, Fiona Kuang, Kashish Mistry, Raisa Islam Aishy, Sara Ahmad, Joaquin Matres, Dirk R. Englund, and Joyce K. S. Poon, *AI agents for photonic integrated circuit design automation*, arXiv:2508.14123.

[37] Rong Sha, Binglin Wang, Jun Yang, Xiaoxiao Ma, Chengkun Wu, Liang Yan, Chao Zhou, Jixun Liu, Guochao Wang, Shuhua Yan, and Lingxiao Zhu, *LLM-based multi-agent copilot for quantum sensor*, arXiv:2508.05421.

[38] Shuxiang Cao, Zijian Zhang, Mohammed Alghadeer, Simone D. Fasciati, Michele Piscitelli, Mustafa Bakr, Peter Leek, and Alán Aspuru-Guzik, *Automating quantum computing laboratory experiments with an agent-based AI framework*, Patterns **6**, 101372 (2025).

[39] OpenAI, GPT-5 system card, 2025, https://cdn.openai.com/gpt-5-system-card.pdf.

[40] See Supplemental Material at http://link.aps.org/supplemental/10.1103/xnqc-q6nt for details on analysis tools, experiment tools, physical systems, models discovered by the agent, and summaries of example explorations.

[41] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz, *Discovering governing equations from data by sparse identification of nonlinear dynamical systems*, Proc. Natl. Acad. Sci. U.S.A. **113**, 3932 (2016).

[42] *Google AI, Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities*, arXiv:2507.06261.

[43] Mario Krenn, Jonas Landgraf, Thomas Foesel, and Florian Marquardt, *Artificial intelligence and machine learning for quantum technologies*, Phys. Rev. A **107**, 010101 (2023).

[44] Agnes Valenti, Evert van Nieuwenburg, Sebastian Huber, and Eliska Greplova, *Hamiltonian learning for quantum error correction*, Phys. Rev. Res. **1**, 033092 (2019).

[45] Mikel Landajuela, Chak Shing Lee, Jiachen Yang, Ruben Glatt, Claudio P. Santiago, Ignacio Aravena, Terrell Mundhenk, Garrett Mulcahy, and Brenden K. Petersen, *A unified framework for deep symbolic regression*, Proceedings of the 36th International Conference on Neural Information Processing Systems, NIPS '22 (2022), Vol. **35**, pp. 33985–33998, https://proceedings.neurips.cc/paper_files/paper/2022/file/dbca58f35bddc6e4003b2dd80e42f838-Paper-Conference.pdf.

[46] Brian de Silva, Kathleen Champion, Markus Quade, Jean-Christophe Loiseau, J. Kutz, and Steven Brunton, *PySINDy: A PYTHON package for the sparse identification of nonlinear dynamical systems from data*, J. Open Source Software **5**, 2104 (2020).

[47] Samuel H. Rudy, Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz, *Data-driven discovery of partial differential equations*, Sci. Adv. **3**, e1602614 (2017).

[48] Silviu-Marian Udrescu and Max Tegmark, *AI Feynman: A physics-inspired method for symbolic regression*, Sci. Adv. **6**, eaay2631 (2020).

[49] Silviu-Marian Udrescu, Andrew Tan, Jiahai Feng, Orisvaldo Neto, Tailin Wu, and Max Tegmark, *AI Feynman 2.0: Pareto-optimal symbolic regression exploiting graph modularity*, Advances in Neural Information Processing Systems (2020), Vol. **33**, pp. 4860–4871, https://proceedings.neurips.cc/paper/2020/hash/33a854e247155d590883b93bca53848a-Abstract.html.

[50] Kevin Maik Jablonka, Philippe Schwaller, Andres Ortega-Guerrero, and Berend Smit, *Leveraging large language models for predictive chemistry*, Nat. Mach. Intell. **6**, 161 (2024).

[51] Joren Van Herck, María Victoria Gil, Kevin Maik Jablonka, Alex Abrudan, Andy S. Anker, Mehrdad Asgari, Ben Blaiszik, Antonio Buffo, Leander Choudhury, Clemence Corminboeuf *et al.*, *Assessment of fine-tuned large language models for real-world chemistry and material science applications*, Chem. Sci. **16**, 670 (2025).

[52] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang, *Biobert: A pre-trained biomedical language representation model for biomedical text mining*, Bioinformatics **36**, 1234 (2019).

[53] Renqian Luo, Liai Sun, Yingce Xia, Tao Qin, Sheng Zhang, Hoifung Poon, and Tie-Yan Liu, *Biogpt: Generative pre-trained transformer for biomedical text generation and mining*, Briefings Bioinf. **23**, bbac409 (2022).

[54] Nägele Maximilian and Marquardt Florian, *SCIEXPLORER framework*, https://github.com/MaxNaeg/SciExplorer.git (2025).

[55] Nägele Maximilian and Marquardt Florian, *SCIEXPLORER results*, https://github.com/MaxNaeg/SciExplorerResults.git (2025).

[56] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang, *JAX: Composable transformations of* Python + NumPy *programs*, http://github.com/jax-ml/jax (2018).