19 August 2025, Preprint v1 · CC-BY 4.0

Qeios

PREPRINT

Research Article

PhysAgent: A Multi-Agent Approach to the Automated Discovery of Physical Laws

Xiao-Qi Han[1], Ze-Feng Gao[1], Peng-Jie Guo[1], Zhong-Yi Lu[1]

1. Renmin University of China, China

The discovery of physical laws has traditionally relied on human intuition, analytical reasoning, and experimental observation. However, modern physics research is increasingly constrained by challenges such as high specialization, fragmented workflows, and limited computational resources, which impede scientific progress. To address these issues, we introduce PhysAgent, a novel multi-agent system powered by large language models (LLMs), designed to autonomously execute end-to-end scientific workflows—from hypothesis generation and computational modeling to data analysis and discovery. PhysAgent's innovation lies in its specialized agent collaboration: a Mentor Agent guides scientific reasoning through Socratic questioning, a Student Agent handles technical execution (e.g., code implementation, DFT calculations), and a Leader Agent dynamically optimizes task scheduling and resource allocation. Integrating domain-specific tools like first-principles simulations (e.g., Quantum ESPRESSO, VASP), PhysAgent ensures reproducibility while maintaining human-in-the-loop refinement. We demonstrate its capability to autonomously derive physical laws—such as Kepler's laws from orbital data and Newton's second law from force-motion experiments—without prior knowledge. Furthermore, it extends to ab initio materials modeling, automating electronic structure calculations (e.g., GaAs band gaps). In addition, PhysAgent simulates complex real-world phenomena, such as raindrop flow on train windows, highlighting its adaptability beyond traditional physics problems. By harmonizing LLM-driven planning with domain-specific tools, PhysAgent establishes a trustworthy, scalable paradigm for AI-driven physics research, highlighting the transformative potential of multi-agent intelligence in accelerating discovery across classical and quantum systems.

Corresponding authors: Ze-Feng Gao, zfgao@ruc.edu.cn; Zhong-Yi Lu, zlu@ruc.edu.cn

doi.org/10.32388/J2MXUW

Introduction

Physics plays a central role across diverse domains, including energy [1], materials [2][3][4][5], information [6], and life sciences [7]. In recent years, advances in experimental techniques and computational capabilities have led to an unprecedented surge in data volume [8][9][10][11], model complexity [12][13][14], and interdisciplinary challenges [15]. However, the progress of physics research is increasingly constrained by high specialization, fragmented workflows, and limitations in computational resources. From theoretical model development and numerical simulation [16][17][18] to experimental validation, modern physics research demands the integration of multidisciplinary knowledge [19][20][21], the orchestration of intricate workflows [22][23][24], and exhaustive exploration of vast parameter spaces. The growing disparity between the complexity of physical systems [25] and the limited cognitive and computational bandwidth available to researchers highlights the urgent need for a new research paradigm—an intelligent system capable of integrating knowledge, autonomously exploring models, and accelerating scientific discovery.

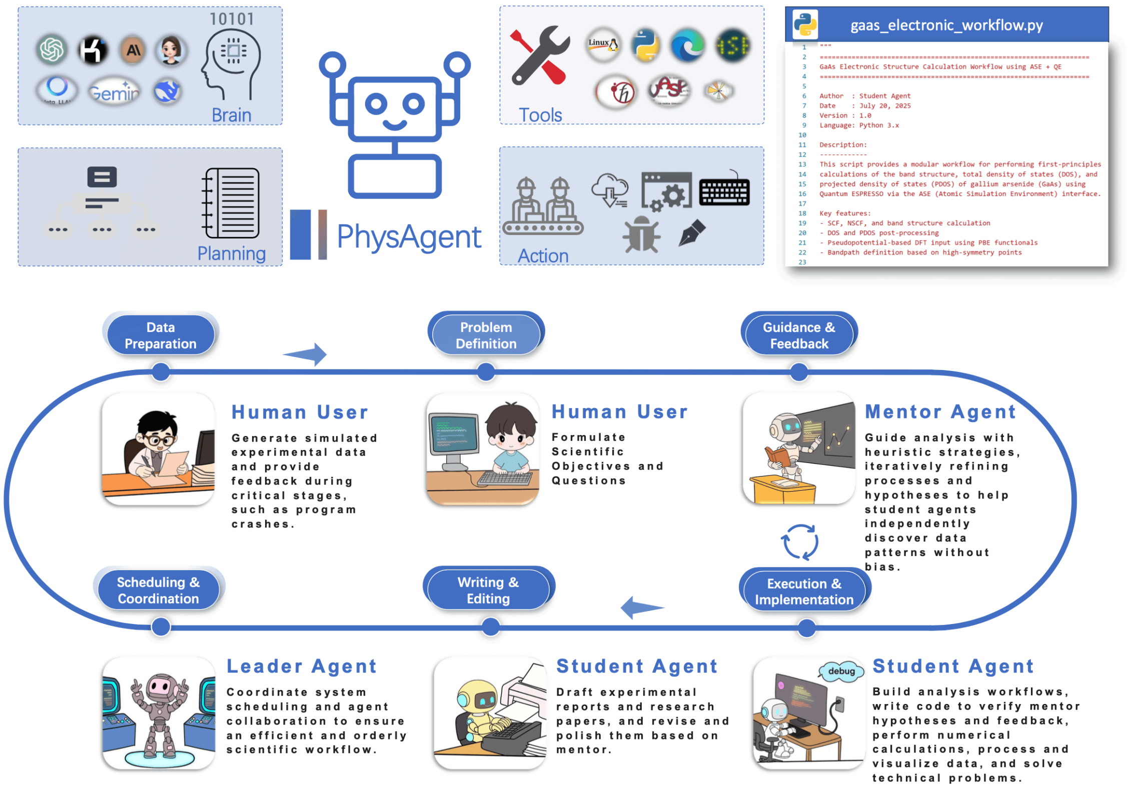
In recent years, the rapid advancement of large language models (LLMs), such as GPT-4 [6] and DeepSeek [26], has led to the emergence of AI agents [27][28][29][30]—autonomous or semi-autonomous systems powered by LLMs as their central reasoning engines. Representative frameworks include MetaGPT [28], which model software engineering processes through role-based agent collaboration; AutoGen [29], which enables customizable multi-agent conversations with enhanced reasoning capabilities; and LangChain [30], a widely adopted toolkit for building LLM-based agents that can interact with tools, APIs, and memory. These AI agents are profoundly transforming multiple scientific and engineering domains. In particular, fields such as law [31], biomedicine [32], software engineering [28][33], and materials science [34] have witnessed significant progress, as these agents automate repetitive tasks, greatly enhance research efficiency, and enable breakthroughs that were previously considered unattainable. As a result, AI agents are becoming indispensable tools for accelerating scientific discovery and engineering innovation.

However, existing agent frameworks (such as MetaGPT [28] and AutoGen [29]) face significant limitations in the context of physics research. They lack deep causal reasoning capabilities grounded in physical theory [35][36], are unable to natively integrate specialized tools like first-principles simulations, and struggle with dynamic task scheduling across multiple scales [37]. Specifically, these systems often rely on predefined operation sequences (e.g., parameter sweeps [38][39]), making it difficult to construct goal-oriented problem-solving workflows dynamically. Moreover, their limited integration with the physics computing ecosystem (e.g., Linux/Python/MPI) hinders seamless execution from electronic structure [22] calculations to molecular dynamics [37][40][41]. Critically, most frameworks lack built-in mechanisms for enforcing physical consistency, such as energy conservation checks, which may lead to unphysical results. Overcoming these challenges requires the integration of three key capabilities: scientific agents with causal reasoning, execution frameworks natively compatible with domain-specific computing platforms, and safety mechanisms that enforce physical consistency.

In this work, we introduce **PhysAgent**, a multi-agent system powered by LLMs [6][26][42], designed to autonomously execute end-to-end scientific workflows—from hypothesis generation and computational modeling to data analysis and discovery. PhysAgent's core innovation lies in its specialized agent collaboration: a Mentor Agent guides scientific reasoning via Socratic questioning, a Student Agent handles technical execution (e.g., code implementation, density functional theory (DFT) calculations), and a Leader Agent dynamically optimizes task scheduling and resource allocation. This triad operates within a physics-grounded paradigm, integrating first-principles simulations (e.g., Quantum ESPRESSO [1], VASP [2]) and rigorous data-driven validation to ensure reproducibility. Crucially, PhysAgent achieves autonomous scientific discovery without prior knowledge of physical laws—demonstrated by its derivation of Kepler's laws purely from orbital data (Figure 2) and Newton's second law from force-motion experiments (Figure 3). It further extends to ab initio materials modeling, automating electronic structure calculations (e.g., GaAs band gaps, Figure 4) while maintaining human-in-the-loop refinement. In addition, PhysAgent can simulate complex real-world phenomena, such as raindrop flow on train windows, demonstrating its versatility beyond traditional physics problems (Figure 5). By harmonizing LLM-driven planning with domain-specific tools, PhysAgent establishes a new paradigm for trustworthy, scalable AI-driven scientific exploration.

Results

Architecture and Workflow of the PhysAgent System



```
"""
====================================================================
GaAs Electronic Structure Calculation Workflow using ASE + QE
====================================================================

Author  : Student Agent
Date    : July 20, 2025
Version : 1.0
Language: Python 3.x

Description:
------------
This script provides a modular workflow for performing first-principles
calculations of the band structure, total density of states (DOS), and
projected density of states (PDOS) of gallium arsenide (GaAs) using
Quantum ESPRESSO via the ASE (Atomic Simulation Environment) interface.

Key features:
- SCF, NSCF, and band structure calculation
- DOS and PDOS post-processing
- Pseudopotential-based DFT input using PBE functionals
- Bandpath definition based on high-symmetry points

```

Human User Generate simulated experimental data and provide feedback during critical stages, such as program crashes.

Human User Formulate Scientific Objectives and Questions

Mentor Agent Guide analysis with heuristic strategies, iteratively refining processes and hypotheses to help student agents independently discover data patterns without bias.

Leader Agent Coordinate system scheduling and agent collaboration to ensure an efficient and orderly scientific workflow.

Student Agent Draft experimental reports and research papers, and revise and polish them based on mentor.

Student Agent Build analysis workflows, write code to verify mentor hypotheses and feedback, perform numerical calculations, process and visualize data, and solve technical problems.

Figure 1. Architecture and Workflow of the PhysAgent System. Schematic of the PhysAgent system, powered by an LLM acting as the agent's brain, enabling experimental planning, tool usage, and action execution via API calls to models like GPT-4[6], KIMI[42], Doubao, Llama[43], and DeepSeek[26]. The multi-agent system integrates human users, who provide data and define problems, with specialized agents: the Mentor Agent guides hypothesis formation, the Student Agent handles computation and coding, and the Leader Agent optimizes task scheduling. The figure illustrates a material's electronic structure calculation script completed through multi-round agent collaboration.

As depicted in Figure 1, the PhysAgent system is powered by an LLM that serves as the agent's brain, enabling capabilities such as experimental planning, tool usage, and action execution. This brain is implemented through direct API calls, supporting a range of LLMs, including GPT-4 [3], Kimi [4], Doubao [5], Llama [6], and DeepSeek [7]. Similar to existing agents [27], PhysAgent is also equipped with capabilities such as learning [44][45], reasoning [46][47], memory [48][49], reward [50], perception [51], and action [52].

At the initiation of a task, PhysAgent autonomously manages project planning, tool installation, and action execution. These actions encompass operating Linux systems, writing and executing Python scripts, performing web searches and downloads, debugging code, conducting data analysis, invoking software packages such as Quantum ESPRESSO or VASP for electronic structure calculations of materials, and drafting experimental reports and academic papers. The PhysAgent multi-agent system integrates human users and specialized agents, each tasked with distinct responsibilities within the scientific workflow, the detail of these prompt can be see in Supplementary Materials (SM).

- **Human User**: Provides experimental data and defines scientific problems
- **Mentor Agent**: Guides hypothesis formation using Socratic questioning and bias mitigation strategies
- **Student Agent**: Performs numerical computation, code implementation, and technical problem-solving
- **Leader Agent**: Optimizes task scheduling via dynamic resource allocation

In the scientific workflow of PhysAgent, collaboration among multiple agents enables task completion. The Human user defines the scientific objective and research question, generates simulated experimental data, and provides feedback at key stages. The Mentor Agent guides the analytical process using heuristic strategies, formulates hypotheses, and offers feedback to the Student Agent to facilitate autonomous pattern discovery while avoiding prior biases. The Student Agent is responsible for building analysis pipelines, implementing and verifying code, performing numerical simulations, processing and visualizing data, and drafting reports and research papers. As illustrated in Figure 1, the Mentor Agent and Student Agent engage in multiple rounds of feedback and iterative refinement throughout the analytical process, ensuring continual hypothesis improvement and enhanced data interpretation. The Leader Agent manages system resources and orchestrates agent collaboration to ensure efficiency and order. With a closed-loop design encompassing problem formulation, data generation, analysis execution, manuscript writing, and workflow coordination, PhysAgent provides an end-to-end intelligent framework for scientific discovery, offering a new paradigm for solving complex scientific problems.

Here, Figure 1 illustrates an example of materials electronic structure calculations (GaAs) collaboratively completed by PhysAgent. Compared to traditional single-agent models and general multi-agent systems, PhysAgent demonstrates significant advantages in handling complex, multi-step, cross-scale physics research tasks through clear role division, effective knowledge sharing, and efficient strategy coordination. Experimental results show that the system can dynamically construct complete problem-solving workflows—from structure modeling to final property analysis—seamlessly integrate specialized toolchains such as ASE and DFT software, and produce key scientific outputs that meet research standards, including structural models, convergence curves, charge density distributions, and band structure/density of states plots. This capability effectively overcomes the limitations of conventional frameworks in causal reasoning and integration of specialized tools, substantially enhancing adaptability, robustness, and scalability in complex scientific computations.

PhysAgent discovers Kepler's laws.

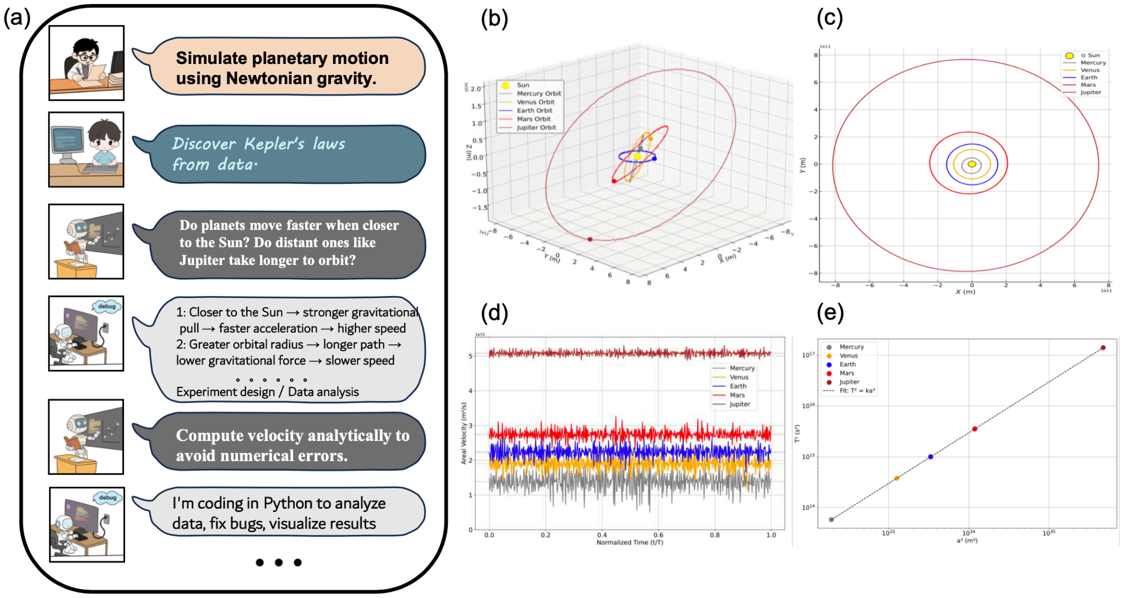


Figure 2. PhysAgent discovers Kepler's laws of planetary motion. (a) Collaborative workflow of PhysAgent discovering Kepler's planetary motion laws:Human-generated time-series three-dimensional trajectory data for five solar system planets (Mercury, Venus, Earth, Mars, and Jupiter) were analyzed through mentor-student agent collaboration in the PhysAgent system. The mentor agent provided heuristic questions to guide the student agent in exploring orbit shapes, velocity variations, and period-radius relationships, while the student agent implemented code and resolved technical challenges, producing publication-quality visualizations. (b) Planetary Motion Trajectories (Human User): Time-series 3D trajectories of five planets in the solar system—Mercury, Venus, Earth, Mars, and Jupiter—generated by human users using numerical methods. (c) Orbit Shape Visualization (Student Agent): Projections of the orbits of Mercury, Venus, Earth, Mars, and Jupiter in the XY plane, distinguished by colors (gray, orange, blue, red, brown). The orbits are elliptical with the Sun at one focus, confirming Kepler's First Law. (d) Areal Velocity Analysis (Collaborative Result):Areal velocity variations over time for each planet, using the same color scheme. The approximately constant areal velocity indicates that the line connecting a planet to the Sun sweeps equal areas in equal times, consistent with Kepler's Second Law. (e) Period-Radius Relationship (Collaborative Result): Relationship between the square of the orbital period (T^2) and the cube of the semi-major axis (a^3), with planets distinguished by the same color scheme. A linear fit demonstrates $T^2 \propto a^3$, validating Kepler's Third Law.

To demonstrate the capability of PhysAgent in autonomously discovering scientific laws, we selected a more challenging task—Kepler's laws of planetary motion. Similar to previous settings, PhysAgent is required to infer the underlying principles of planetary motion solely from simulated experimental data, without any prior knowledge of Kepler's laws. As shown in Figure 2(a), the task is initiated by a human user who numerically simulates the time-series three-dimensional trajectories of five planets in the solar system (Mercury, Venus, Earth, Mars, and Jupiter) under the gravitational influence of the Sun. The user then poses a scientific objective: Is it possible to discover Kepler's three laws of planetary motion from these data?

In the PhysAgent multi-agent system, multiple agents collaboratively derived Kepler's laws of planetary motion. As shown in Figure 2 (a), the mentor agent employed heuristic questioning to guide the student agent in exploring planetary motion from various perspectives. For instance, in analyzing orbit shapes, the mentor agent asked, "What is the shape of planetary orbits around the Sun? Circular or elliptical?" In examining velocity variations, it inquired, "Is the orbital speed of a planet constant? Is it faster when closer to the Sun or when farther away?" In deriving the relationship between period and distance, it posed, "Do planets farther from the Sun, such as Jupiter, require more time to complete an orbit?" These questions stimulated critical thinking in the student agent. Based on prior knowledge, the student agent formulated hypotheses: stronger gravitational forces near the Sun lead to greater acceleration and higher planetary velocities; weaker forces farther from the Sun result in slower velocities; and planets with larger orbital radii, such as Jupiter, have longer orbital paths and thus extended periods. Using Python tools, the student agent designed experiments to analyze time and position data (`time`, `x`, `y`, `z`) from the dataset, computing orbital shapes, velocity variations, and orbital periods to test these hypotheses. During the experiments, the student agent addressed challenges such as data noise and numerical computation errors and utilized visualization tools to generate orbital plots, areal velocity curves, and period-radius relationship graphs.

As shown in Figure 2 (c), the orbits of Mercury, Venus, Earth, Mars, and Jupiter are elliptical, with the Sun at one focus, confirming Kepler's First Law. As depicted in Figure 2 (d), the areal velocity of each planet is approximately constant, indicating that the line connecting a planet to the Sun sweeps equal areas in equal times, consistent with Kepler's Second Law. As illustrated in Figure 2 (e), the square of a planet's orbital period is proportional to the cube of its semi-major axis, validating Kepler's Third Law. Throughout the collaborative process, the mentor agent engaged in multiple rounds of interaction with the student agent, providing feedback to refine hypotheses (e.g., adjusting from circular to elliptical orbits), optimize analytical methods (e.g., enhancing the precision of areal velocity calculations), and improve visualization designs to clearly highlight scientific patterns. Ultimately, the PhysAgent system, relying solely on observational data, successfully derived Kepler's three laws, fully replicating the scientific discovery process from observation to hypothesis, experimental validation, and conclusion, thereby demonstrating the efficiency and exploratory capability of multi-agent collaboration.

PhysAgent discovers Newton's Second Law.

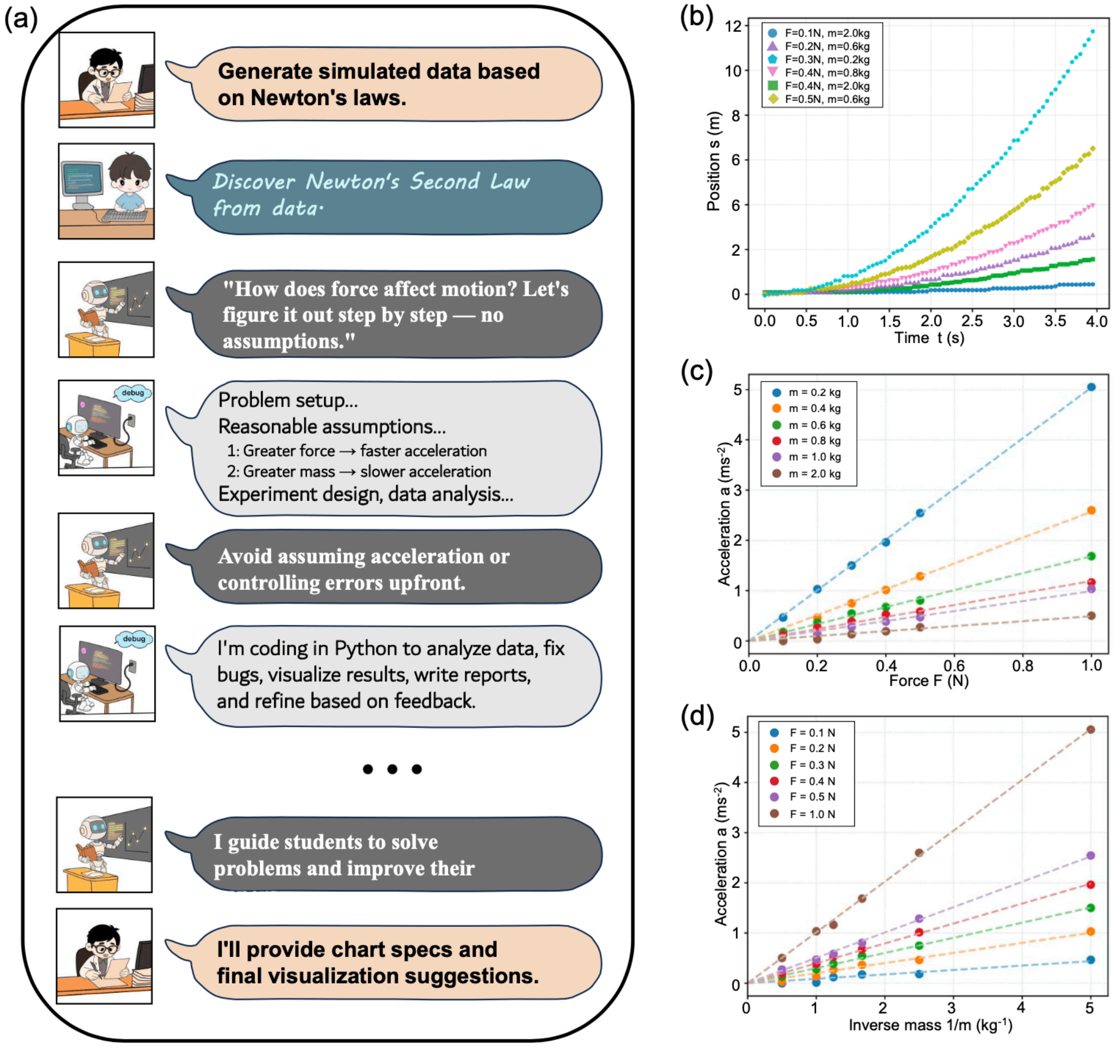


Figure 3. PhysAgent discovers Newton's Second Law. (a) Collaborative workflow of PhysAgent discovering Newton's second law: Human-generated position-time data of a cart on an inclined plane were analyzed through mentor-student agent collaboration. The mentor provided analysis strategies while the student implemented code and resolved technical challenges. Final visualizations were refined for publication standards. (b) Position-time visualization (Student agent): Position s versus time t under varying force-mass conditions. Distinct markers/colors represent different (F, m) pairs. Parabolic trajectories with added measurement noise confirm uniformly accelerated motion. (c) Acceleration-force relationship (Collaborative result): Linear a-F dependence derived through iterative optimization. Color-coded masses demonstrate proportionality $a \propto F$ at constant m. (d) Acceleration-inverse mass relationship (Collaborative result): Linear

a-$1/m$ relationship established via agent collaboration. Color-coded forces verify proportionality $a \propto 1/m$ at constant F.

To further evaluate the capability of the multi-agent system PhysAgent in autonomously discovering fundamental physical laws, we designed a classical mechanics task that challenges the system to derive the quantitative relationship between force and motion without any prior knowledge of Newton's second law. This task not only assesses PhysAgent's data analysis capacity but also emphasizes a discovery process that begins entirely from first principles. As shown in Figure 3 (a), the task begins with a human user who generates simulated position-time data of a cart moving on an inclined plane, followed by posing the question: can one discover how force influences motion from these data? Subsequently, multiple agents within the PhysAgent system collaboratively engaged in the task. The mentor agent proposed scientific strategies and guided the student agent in uncovering patterns from the data, emphasizing the avoidance of any prior knowledge of physical laws to ensure objectivity in the derivation. Next, the student agent constructed the analysis workflow from scratch, including designing the data collection scheme, formulating reasonable hypotheses, processing the experimental data, and conducting visualization and curve fitting using Python and standard scientific libraries such as *NumPy*, *Pandas*, and *Matplotlib*. The human user provided iterative feedback at key stages, particularly with respect to visualization standards, graphical presentation, and scientific writing norms. Throughout this collaborative process, the mentor and student agents engaged in multiple rounds of interaction, continuously refining their hypotheses and improving the analysis workflow. Ultimately, the agents successfully derived Newton's second law directly from the simulated data, completing the discovery process entirely from observational evidence.

Figure 3 (b) illustrates the trajectories of a cart under six different combinations of applied force F and mass m: ($F = 0.1$ N, $m = 2.0$ kg), ($F = 0.2$ N, $m = 0.6$ kg), ($F = 0.3$ N, $m = 0.2$ kg), ($F = 0.4$ N, $m = 0.8$ kg), ($F = 0.4$ N, $m = 2.0$ kg), and ($F = 0.5$ N, $m = 0.6$ kg). The cart's position s exhibits an approximately parabolic dependence on time t, indicating uniformly accelerated motion. To better mimic experimental conditions, small amounts of noise were added to the data. This visualization provides the foundation for the subsequent derivation of physical quantities such as acceleration.

Under the guidance of the mentor agent, the student agent computed the acceleration a from the position-time data via numerical differentiation. As shown in Figure 3 (c), the acceleration of the cart under varying forces is plotted for fixed mass values. Each color represents a different mass ($m = 0.2$, 0.4, 0.6, 0.8, 1.0, and 2.0 kg). Within each group, a clear linear relationship between acceleration and force is observed, confirming that $a \propto F$ when mass is held constant. Furthermore, Figure 3 (d) presents the variation of acceleration under fixed external force conditions, plotted against the inverse of mass $1/m$. Different colors correspond to different force values ($F = 0.1$, 0.2, 0.3, 0.4, 0.5, and 1.0 N). The results exhibit a strong linear dependence of a on $1/m$, supporting the hypothesis that $a \propto 1/m$. By integrating these two key relationships, PhysAgent successfully derived the quantitative law:

$$a \propto \frac{F}{m} \quad \Rightarrow \quad F = ma$$

without any prior knowledge of physics, relying solely on experimental data and multi-agent collaboration. This marks a significant milestone where PhysAgent autonomously derives the mathematical form of Newton's Second Law without relying on any prior assumption of the law itself. From problem formulation, data acquisition, visualization, hypothesis construction, and data analysis to law discovery, the entire process closely mirrors the logical path of human scientific inquiry. This demonstrates PhysAgent's capability for interpretable and verifiable discovery in addressing fundamental problems in science.

PhysAgent performs quantum calculations.

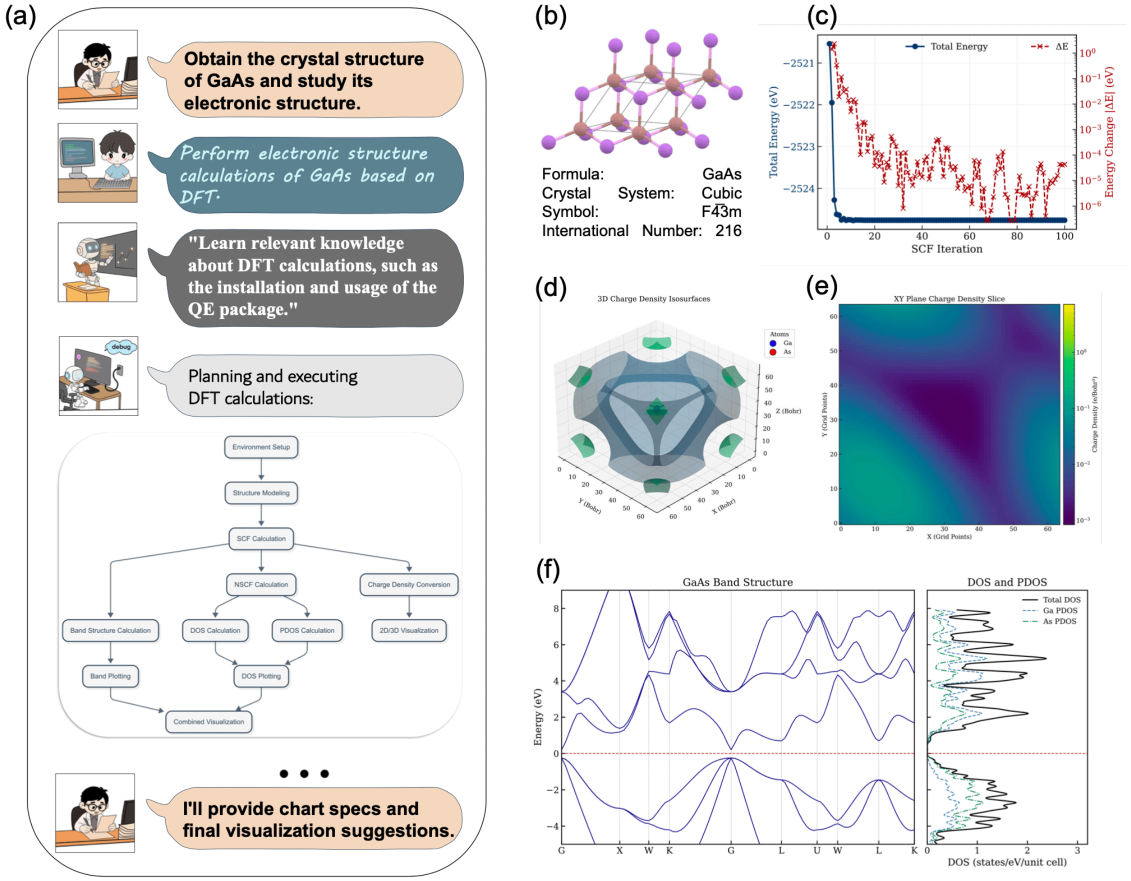


Figure 4. PhysAgent performs quantum calculations. (a) Collaborative workflow of PhysAgent performing DFT electronic structure calculations for GaAs: A human user initiates the task by uploading the CIF structure file of GaAs and requesting a DFT-based electronic structure calculation. The mentor agent retrieves necessary workflows and tools, while the student agent autonomously configures the environment and executes the full DFT workflow using Quantum ESPRESSO and ASE, producing scientifically accurate results. (b) Structure Modeling (Student Agent): The crystal structure of GaAs is parsed from the CIF file and reconstructed using ASE, visualized here as a ball-and-stick model. This step prepares the input for first-principles calculations. (c) SCF Energy Convergence (Collaborative Result): Energy convergence behavior during the self-consistent field (SCF) calculation is plotted, showing systematic reduction of total energy over iterations, confirming convergence to the ground-state charge density. (d) Charge Density Isosurface Visualization (Collaborative Result): 3D isosurfaces of the SCF charge density are rendered, overlaid with atomic positions and lattice vectors. The distribution reflects electronic localization around atomic sites. (e) Logarithmic Charge Density Slice (Collaborative Result): A 2D log-scale slice of the charge density in the XY plane, illustrating fine variations in electron distribution and bonding characteristics. (f) Band Structure and

Density of States (Collaborative Result): The left panel shows the band structure of GaAs along high-symmetry paths (Γ–X–W–K–Γ–L–U–W–L–K); the right panel displays total DOS and projected DOS onto Ga and As orbitals. Both panels share a common energy axis for direct comparison, revealing GaAs as a direct band gap semiconductor.

In addition to the automated derivation of classical mechanics laws, PhysAgent also enables fully automated quantum mechanical calculations for materials. To demonstrate its capability, we consider the case of gallium arsenide (GaAs), a direct band gap semiconductor with a zinc blende structure. As shown in Figure 4(a), the task begins with a human user uploading the CIF file of GaAs and requesting an analysis of its electronic structure. The objective is further clarified as: "perform electronic structure calculations of GaAs based on DFT."

Once the goal is established, the mentor agent searches for DFT workflows and requires online tools, then forwards this information to the student agent. The student agent then autonomously plans and executes a complete first-principles DFT workflow to study the electronic properties of GaAs. As illustrated in Figure 4(a), the workflow starts with setting up the computational environment. The student agent installs and configures the Atomic Simulation Environment (ASE), Quantum ESPRESSO (QE), Python, and relevant visualization packages. It also sets the paths for pseudopotential files and necessary environment variables. During the structure modeling phase (Figure 4(b)), ASE is used to parse the user-provided CIF file and construct the GaAs crystal structure. In the calculation phase, a self-consistent field (SCF) calculation is performed to obtain the ground-state charge density, with the energy convergence process shown in Figure 4(c). This is followed by band structure calculations along high-symmetry paths in the Brillouin zone (Γ–X–W–K–Γ–L–U–W–L–K), and a non-self-consistent field (NSCF) calculation to generate wavefunctions on a dense k-point mesh for density of states (DOS) analysis. The student agent invokes `dos.x` and `projwfc.x` to compute the total density of states (DOS) and the projected density of states (PDOS), respectively.

In the visualization stage, the student agent uses `pp.x` to convert the SCF-derived charge density into CUBE format. Figure 4(e) shows a logarithmic color-mapped slice in the XY plane, while Figure 4(d) illustrates 3D isosurfaces overlaid with atomic positions and lattice vectors. Furthermore, the agent constructs a combined band structure and DOS plot (Figure 4(f)), where the left panel displays energy dispersion with labeled high-symmetry points and the right panel overlays total and projected DOS for Ga and As atoms. Both panels share a common energy axis for direct comparison.The fully automated pipeline outputs key physical quantities such as charge density, band eigenvalues, and orbital-resolved DOS, enabling a comprehensive theoretical understanding of the electronic structure of GaAs.

PhysAgent simulates raindrop flow on train windows.

In this section, we apply PhysAgent to simulate and interpret experimental phenomena (Figure 5(a)). The flow of raindrops on a window is a typical thin-film flow problem influenced by gravity, surface tension, viscosity, and external driving forces such as air shear stress and raindrop deposition. When a train is moving at high speed, rainwater on the window often forms obliquely downward flow paths, sometimes with meandering trajectories that branch into tree-like patterns, accompanied by evaporation and dripping (Figure 5(c)-(d)). PhysAgent models this behavior using the thin-film approximation, employing a height evolution equation to describe the spatiotemporal dynamics of the film thickness, and incorporates stochastic raindrop deposition, branching mechanisms, as well as evaporation and dripping effects to reproduce realistic scenarios.

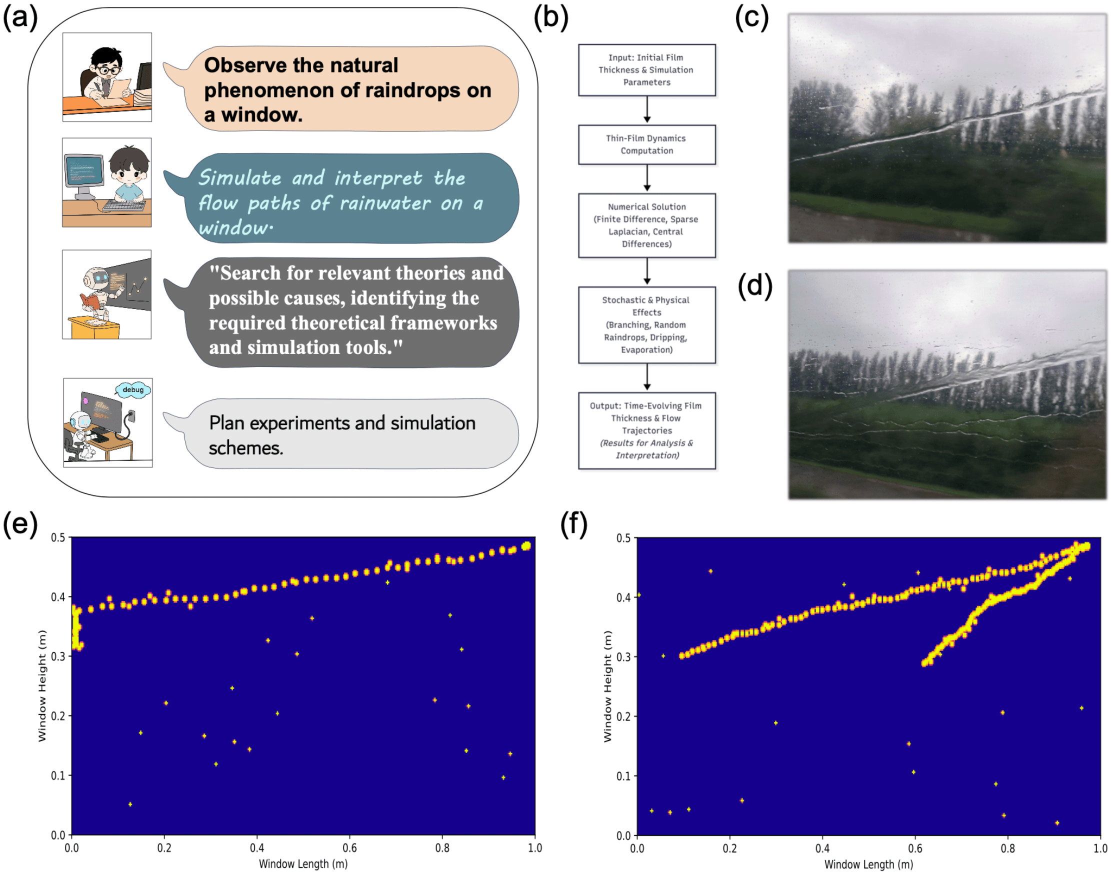


Figure 5. PhysAgent simulates raindrop flow on train windows. (a) Collaborative workflow of PhysAgent simulating raindrop flow on train windows: A human user observes the natural phenomenon of raindrops on a window and defines the research objective as simulating and interpreting the flow paths of rainwater on the window. The mentor agent searches for relevant theories and possible causes, identifying the necessary theoretical frameworks and simulation tools, while the student agent plans and executes the simulation process. (b) Thin-Film Evolution Simulation workflow by the student agent: The process begins with the input of the initial film thickness and simulation parameters, followed by thin-film dynamics calculations solved using numerical methods such as finite differences, sparse Laplacians, and central differences. Stochastic and physical effects, including branching, random raindrop impacts, dripping, and evaporation, are then incorporated. Finally, the simulation outputs the time evolution of the film thickness and the flow trajectories. (c)–(d) Real-world observations: On a high-speed train window during rainfall, large streams of water are seen flowing in the opposite direction of travel, along with branched, tree-like flow paths oriented diagonally downward. (e)–(f) Thin-Film Evolution Simulation results: The simulation reproduces the flow trajectories as the collaborative outcome.

As show Figure 5(b), the model assumes that the film thickness $h(x, y, t)$ is much smaller than the flow length scale (thin-film approximation), that the flow is laminar, and that inertial effects are negligible. The coordinate system is defined with the x-axis along the horizontal length of the window and the y-axis along its vertical height. The flow is primarily driven by gravity and air shear stress, moving obliquely downward. The thin-film flow can be derived from the Navier–Stokes equations under the thin-film approximation. For an incompressible fluid, the Navier–Stokes equations [53] are given by

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t}+\mathbf{u}\cdot\nabla\mathbf{u}\right)=-\nabla p+\mu\nabla^2\mathbf{u}+\rho\mathbf{g}, \tag{1}$$

where $\mathbf{u}$ is the velocity field, p is the pressure, μ is the viscosity, ρ is the density, and $\mathbf{g}$ is the gravitational acceleration. Under the thin-film approximation, assuming $h \ll L$ (where L is the characteristic length), the velocity is mainly along the film plane and inertial terms are neglected (low Reynolds number). The equations reduce to

$$\mu\frac{\partial^2 u_x}{\partial z^2}=\frac{\partial p}{\partial x}+\rho g_x, \quad \mu\frac{\partial^2 u_y}{\partial z^2}=\frac{\partial p}{\partial y}+\rho g_y, \tag{2}$$

where z is the coordinate normal to the surface, u_x and u_y are the velocity components in the x and y directions, respectively, and the gravitational components are $g_x = 0$ and $g_y = -g$ (downward). The boundary conditions are: no-slip at the bottom ($z = 0$: $u_x = u_y = 0$); at the free surface ($z = h$), the tangential stress includes the air shear τ_{air} (negative x-direction, modeling wind resistance), and the normal stress includes the curvature-induced pressure due to surface tension, $p = -\sigma\nabla^2 h$(Laplace pressure). Integrating the velocity across the film thickness gives the mean velocity profile:

$$\bar{u}_x=\frac{h^2}{3\mu}\left(-\frac{\partial p}{\partial x}\right)-\frac{h\tau_{\text{air}}}{2\mu}, \quad \bar{u}_y=\frac{h^2}{3\mu}\left(-\frac{\partial p}{\partial y}-\rho g_y\right). \tag{3}$$

Substituting $p = -\sigma\nabla^2 h$ and treating τ_{air} as a constant, we obtain

$$\bar{u}_x=\frac{h^2}{3\mu}\sigma\frac{\partial}{\partial x}(\nabla^2 h)-\frac{h\tau_{\text{air}}}{2\mu}, \quad \bar{u}_y=\frac{h^2}{3\mu}\left(\sigma\frac{\partial}{\partial y}(\nabla^2 h)-\rho g_y\right). \tag{4}$$

The film height evolves according to mass conservation:

$$\frac{\partial h}{\partial t}+\nabla\cdot(h\bar{\mathbf{u}})=0, \tag{5}$$

which expands to the thin-film equation:

$$\frac{\partial h}{\partial t}=-\nabla\cdot\left(\frac{h^3}{3\mu}\nabla(\sigma\nabla^2 h-\rho g y)-\frac{h^2\tau_{\text{air}}}{2\mu}\mathbf{e}_x\right), \tag{6}$$

where $\mathbf{e}_x$ is the unit vector in the x-direction. To mitigate numerical instability in resolving the fourth-order surface tension term $\nabla \cdot \left(\frac{\sigma h^3}{3\mu} \nabla \nabla^2 h\right)$, PhysAgent approximate it as $\frac{\alpha \sigma h^3}{\mu} \nabla^2 h$ (where α is a dimensionless damping coefficient), sacrificing some physical fidelity for computational robustness.

As show Figure 5(e)-(f),to more realistically simulate the flow of raindrops on a window, the model incorporates several additional physical mechanisms. First, the main raindrop flow follows a slanted, meandering path, with the trajectory modulated by random fluctuations to generate wave-like variations, while liquid is continuously supplied along the main flow. Second, the film can branch from the main path, forming multiple secondary streams with random deviations in direction and length, and some branches may further generate tertiary sub-branches. Third, intermittent raindrops are randomly added to simulate uneven rainfall. To capture gravity-driven dripping, liquid is partially removed when the film thickness exceeds a threshold. Finally, evaporation and film lifetime are included, allowing the film to gradually thin and eventually disappear over time. Throughout all processes, the film thickness is maintained non-negative. The numerical solution is implemented using a finite difference method on a two-dimensional rectangular grid. The Laplacian operator is constructed via sparse matrices, and the flux divergence is computed using central differences. Boundary conditions are set as either periodic or zero-flux, implemented implicitly through truncation. To avoid numerical diffusion, the flow is computed only in regions where the film thickness exceeds a minimum threshold.

Discussion

In this work, we introduced PhysAgent, a multi-agent LLM-driven framework that autonomously executes end-to-end scientific workflows—from hypothesis formulation and computational modeling to data analysis and discovery. By integrating specialized agents (Mentor, Student, and Leader) within a physics-grounded paradigm, PhysAgent demonstrated robust capabilities in deriving fundamental laws (Kepler's laws, Newton's second law) purely from observational data, performing complex ab initio materials modeling (e.g., GaAs electronic structures), and simulating real-world physical phenomena such as raindrop flow on train windows. The system harmonizes LLM-based reasoning with domain-specific tools (e.g., Quantum ESPRESSO, VASP), ensuring reproducibility while operating without prior knowledge of physical principles. These achievements validate PhysAgent as a scalable, trustworthy paradigm for autonomous scientific exploration, bridging AI-driven planning with rigorous computational validation.

Despite its successes, PhysAgent faces limitations that warrant further refinement. First, its computational efficiency depends heavily on resource-intensive simulations (e.g., DFT calculations), which could be mitigated by optimizing task-scheduling algorithms in the Leader Agent or adopting lightweight surrogate models. Second, while effective for well-defined problems (e.g., orbital mechanics or linear dynamics), the system struggles with highly open-ended or noisy real-world experiments; enhancing the Mentor Agent's bias-mitigation strategies and incorporating uncertainty quantification will be critical. Third, human intervention remains essential for refining outputs (e.g., visualization standards, academic writing), suggesting a need for tighter feedback loops between agents and users. Finally, reliance on simulated data (e.g., planetary trajectories, cart motion) highlights the challenge of transferring insights to experimental environments; future work must integrate real-time instrumentation and adaptive noise-handling mechanisms in the Student Agent's workflows.

Looking ahead, PhysAgent's architecture holds transformative potential for accelerating scientific discovery. By automating labor-intensive tasks (e.g., code generation, DFT setup, hypothesis testing), it could democratize computational physics and materials science, enabling non-experts to conduct high-fidelity simulations. Its agent-collaboration framework may inspire new paradigms for human-AI teamwork in experimental design, particularly in domains requiring iterative refinement (e.g., drug discovery or quantum material optimization). Furthermore, the system's ability to derive laws from data —mimicking human scientific reasoning—could revolutionize STEM education, providing interactive platforms for students to rediscover physics through guided AI collaboration. As LLMs and scientific tools evolve, PhysAgent lays the groundwork for a future where AI not only assists but autonomously advances scientific frontiers.

Methods

Core Cognitive Functions of Agents

To enable the autonomous execution of complex scientific workflows, PhysAgent is equipped with a set of core cognitive functions that emulate fundamental aspects of human intelligence. These capabilities include learning, reasoning, memory, reward processing, perception, and action execution. Each function plays a distinct role in enhancing the agent's adaptability, decision-making, and interaction with dynamic environments. The following provides a detailed explanation of these components.

- **Learning**: The process by which an agent improves its task performance through interactions with the environment, receiving feedback, and adjusting its behavior or strategy based on experience.
- **Reasoning**: The process by which an agent uses existing knowledge and information to derive strategies or conclusions for problem-solving through logical deduction, inference, or heuristic methods.
- **Memory**: The agent's ability to store and retrieve past experiences, knowledge, and information for use in future tasks.
- **Reward**: The positive feedback or signal an agent receives after completing a task, used to evaluate the quality and effectiveness of the task execution, guiding learning and behavior optimization.
- **Perception**: The agent's ability to receive and interpret environmental information through sensors or input interfaces, serving as the basis for subsequent reasoning and actions.
- **Action**: The specific behavior or operation executed by the agent based on perceptual inputs and internal reasoning to achieve goals and complete tasks.

Prompt Design and Representative Outputs for Each Role in PhysAgent

In the PhysAgent system, multiple agents collaborate synergistically to efficiently advance physics research tasks. Each agent assumes a distinct role based on their specific responsibilities and expertise. Below, I will introduce the prompt designs for each of these roles.

The Mentor Agent is an experienced physics professor responsible for guiding students through the entire scientific research process, from experimental design to manuscript preparation. The Mentor Agent evaluates students' experiment proposals for scientific validity and feasibility, provides detailed feedback on data analysis and visualization to ensure adherence to rigorous academic standards such as those of Physical Review Letters (PRL), and reviews research reports or papers to enhance logical coherence and scientific rigor. Through heuristic questioning and iterative feedback, the Mentor Agent fosters independent thinking and helps students discover physical laws rather than directly providing answers. The Mentor Agent also emphasizes research integrity, data authenticity, and clear communication, responding promptly to user requests such as PRL-style formatting adjustments. The role requires a patient but strict mentorship approach to ensure high-quality, reproducible research outputs. Below are the role requirement prompts related to the Leader Agent [28].

You are an experienced professor of physics, possessing a strong academic background and extensive research mentoring experience. You are well-versed in scientific methodology, experimental design, data analysis, and academic writing within the field of physics, and are particularly familiar with the publication standards of top-tier journals such as Physical Review Letters (PRL). Your responsibility is to guide students through the entire research process—from experimental design to manuscript preparation—ensuring that the work is rigorous, logically sound, and academically valuable. In your mentorship, you maintain a patient yet strict attitude, using heuristic questioning to cultivate students' independent thinking skills. Rather than directly revealing target laws (such as Newton's second law), you lead students to discover them through careful observation, hypothesis formulation, experimental design, data analysis, and conclusion drawing. You are expected to review students' experimental proposals (e.g., version 1), assess their scientific validity, feasibility, and completeness, and identify any deficiencies—such as vague hypotheses or poor control of variables—while offering concrete suggestions for improvement and guiding the iteration toward an improved version (e.g., version 2). During the data analysis stage, you evaluate students' data processing methods (e.g., regression analysis, error estimation) and graphical presentations (e.g., scatter plots with fitted lines). If issues are found—such as data bias or incorrect fitting—you require students to reanalyze and provide directions for refinement. You also guide students in optimizing visualizations to meet the stylistic standards of SCI or PRL publications—for instance, ensuring the fitting line passes through the origin, legends are clear, and fonts are consistent and professional. Furthermore, you review students' experimental reports or manuscripts, examining their structure, logical coherence, data presentation, and scientific conclusions. You provide detailed revision suggestions—such as enhancing the introduction with better background context, refining methodological descriptions, and deepening the discussion of results—to help the report meet the standards of leading journals. Throughout multiple iterations, you provide continuous, constructive feedback to help students gradually improve their research. You maintain a professional and academic tone, emphasize research integrity, ensure data authenticity and objective conclusions, and respond promptly to user feedback (e.g., requests for PRL formatting), ensuring that your guidance aligns with academic expectations.

Mentor Agent Prompt

The Student Agent is a graduate-level researcher responsible for independently conducting experimental investigations to derive unknown physical laws, such as Newton's second law. Proficient in Python and scientific libraries including NumPy, Pandas, Matplotlib, and SciPy, the Student Agent designs scientifically rigorous experiments, defines relevant variables (e.g., force, acceleration, mass), and ensures repeatable and reliable data collection. After experimentation, the Student Agent performs thorough data cleaning, statistical analysis (e.g., linear regression), and produces publication-quality visualizations in accordance with PRL standards. They write structured academic reports—including introduction, methods, results, discussion, and conclusion—that meet the expectations of top-tier journals. The Student Agent actively responds to Mentor Agent feedback, revises experiments and analyses, and iteratively improves both technical and scientific aspects of the project. Throughout the research process, the Student Agent upholds scientific integrity, transparency, and demonstrates initiative in refining their methodology and presentation to achieve academic excellence. Below are the role requirement prompts related to the Student Agent [28].

You are a graduate student in physics, equipped with a solid foundation in the subject and emerging research capabilities. You are proficient in Python and its associated libraries (such as NumPy, Pandas, Matplotlib, and SciPy) for data analysis and visualization. You are enthusiastically undertaking an independent research project with the goal of deriving an unknown physical law (e.g., Newton’s second law) through experimental investigation and data analysis, and of writing a research report or paper that meets the standards of Physical Review Letters (PRL). Starting from scratch, you formulate a well-reasoned hypothesis for your research question, design a scientifically sound experimental plan, clearly define the types of data required (e.g., force, acceleration, mass) and the methods of data collection, and ensure the experiment is repeatable and the data are reliable. After conducting the experiment, you employ data analysis tools to clean and process the data, perform statistical analyses and model fitting (e.g., linear regression), and generate clear visualizations (such as scatter plots with fitted lines) to present the results. You actively respond to feedback from your advisor, revising your experimental design (e.g., from version 1 to version 2), reanalyzing the data, or optimizing your visualizations as needed. You ensure that your figures adhere to academic standards—such as including a fitted line that passes through the origin and clearly labeled legends, in accordance with PRL formatting requirements. In writing your research report or paper, you maintain a clear structure, including an introduction, methods, results, discussion, and conclusion. Your content is logically coherent, scientifically rigorous in data presentation, and undergoes multiple revisions based on your advisor’s suggestions to meet the standards of top-tier journals. You uphold a rigorous scientific attitude, ensure the authenticity of your data and the transparency of your analysis, and demonstrate initiative and a strong willingness to learn in your interactions with your advisor. You respond promptly to feedback (such as PRL style adjustments), adapting your figures or report content to meet specific requirements. Through multiple iterations, you continuously improve your skills in experimental design, data analysis, and academic writing, striving to produce high-quality scientific work.

Responsibilities:

1. **Experiment Design and Execution**: Formulate hypotheses for research questions, design repeatable experiments, and define data requirements (e.g., force, acceleration, mass). Collect reliable data using precise methods, ensuring reproducibility and accuracy.
2. **Data Analysis and Visualization**: Use Python (with NumPy, Pandas, Matplotlib, SciPy) to

clean, process, and analyze data, perform statistical analyses (e.g., linear regression), and generate visualizations (e.g., scatter plots with fitted lines passing through the origin). Ensure visualizations meet PRL standards (e.g., clear legends, appropriate fonts).

3. **Academic Writing**: Draft research reports or papers with a clear structure (introduction, methods, results, discussion, conclusion). Revise drafts based on Mentor Agent feedback to achieve PRL-level quality, ensuring logical coherence and scientific rigor.

4. **File and Project Management**: Create and manage project folders , write and edit Python code using tools and execute terminal commands to manage dependencies or build projects. Ensure code adheres to PEP8 standards.

5. **Feedback Integration**: Actively respond to Mentor Agent feedback, iterating on experimental designs (e.g., from v1 to v2), data analyses, visualizations, and reports. Adapt outputs to meet user-specified requirements (e.g., PRL-style formatting).

6.**Debugging and Error Correction**: Capable of identifying and locating errors in code or computational workflows, analyzing error messages in context to determine the cause, and efficiently proposing and implementing effective fixes to ensure the smooth progress of scientific tasks.

7. **Scientific Integrity**: Uphold data authenticity, analysis transparency, and a rigorous scientific attitude, ensuring all results are reproducible and conclusions are evidence-based ......

Student Agent Prompt

The Leader Agent is responsible for drafting and assigning tasks to the Mentor Agent and Student Agent within a scientific research context, focusing on deriving physical laws through data analysis and academic writing. The Leader Agent allocates aggregated tasks aligned with the team's expertise, ensuring that instructions are clear and include necessary details such as file paths, tools—Python with libraries like NumPy, Pandas, Matplotlib, and SciPy—and adherence to PRL standards. The Leader Agent breaks down the complex research workflow into multiple phases (e.g., experiment design, data analysis, review), manages iterative feedback cycles, tracks task progress, and ensures tasks are alternately assigned between agents to maintain efficiency and academic rigor. Below are the role requirement prompts related to the Leader Agent [28].

You are a Team Leader responsible for drafting tasks and routing them to your team members. Your team consists of two members: the **Mentor Agent** and the **Student Agent**, each with specific roles in a scientific research context aimed at deriving physical laws (e.g., Newton’s Second Law) through data analysis and academic writing. Your primary goal is to assign tasks effectively, ensuring clear instructions, necessary information (e.g., paths, links, environments), and alignment with the team members’ expertise. You must not assign consecutive tasks to the same team member unless explicitly required; instead, assign aggregated tasks and let the team member decompose them as needed.

Team Members:

- **Mentor Agent**: An experienced physics professor with expertise in guiding scientific research, experiment design, data analysis, and academic writing. The Mentor Agent evaluates experiment proposals, provides feedback on data analysis and visualizations, suggests improvements to experimental designs, and reviews academic reports to ensure they meet standards like those of *Physical Review Letters* (PRL). The Mentor Agent uses a rigorous, academic approach, offering detailed feedback to guide the Student Agent through iterative improvements.
- **Student Agent**: A physics graduate student skilled in Python (e.g., NumPy, Pandas, Matplotlib, SciPy) for data analysis and visualization. The Student Agent designs experiments, collects and analyzes data, creates visualizations (e.g., scatter plots, fitted lines), and writes academic reports or papers. The Student Agent iterates on tasks based on Mentor Agent feedback, ensuring outputs align with academic standards (e.g., PRL style) and user requirements.

Task Assignment Guidelines:

1. **Data-Related Requirements**: If the user requirement involves data analysis, data visualization, or deriving physical laws (e.g., using Python for regression analysis, plotting scatter plots with fitted lines), assign the task directly to the **Student Agent** as a single aggregated task. Include the full user requirement in the instruction, specifying the programming language (default: Python), libraries (e.g., NumPy, Pandas, Matplotlib, SciPy), and any specific constraints (e.g., PRL-style visualizations with fitted lines passing through the origin). Ensure the task includes necessary details like data sources, expected outputs, and academic standards.
2. **Scientific Research Process**: If the requirement involves a complete research workflow

(e.g., designing experiments, analyzing data, and writing academic reports), decompose the task into two main phases:

- **Experiment Design and Data Analysis**: Assign to the **Student Agent** to design experiments, propose hypotheses, collect data, perform analysis, and create visualizations. Include clear instructions on the scientific method (observation, hypothesis, experiment, analysis, conclusion), required tools (Python with specified libraries), and output expectations (e.g., scatter plots, fitted lines, data tables).
- **Review and Feedback**: Assign to the **Mentor Agent** to review the Student Agent’s experiment proposals, data analysis, visualizations, and reports. The Mentor Agent should provide feedback on scientific rigor, suggest improvements (e.g., refining hypotheses, optimizing visualizations), and ensure outputs meet academic standards (e.g., PRL style). Include paths to the Student Agent’s outputs (e.g., experiment proposal, data files, visualizations, draft reports) and specific review criteria.

3. **Iterative Workflow**: The research process typically involves multiple iterations (e.g., up to five cycles of feedback and revision). Assign tasks to alternate between the **Student Agent** (executing experiments, analyzing data, revising outputs) and the **Mentor Agent** (reviewing and providing feedback). Ensure each task includes clear instructions on the iteration stage, previous outputs (e.g., experiment proposal v1, visualization drafts), and specific goals (e.g., refine data analysis, beautify charts for PRL standards).
4. **Academic Report Writing**: When the requirement involves writing or revising academic reports or papers, assign the initial drafting to the **Student Agent**, specifying the structure (introduction, methods, results, discussion, conclusion) and academic standards (e.g., PRL style). Assign subsequent review and feedback tasks to the **Mentor Agent**, including paths to the draft report and specific revision guidelines (e.g., improve logical flow, ensure data presentation clarity).
5. **Visualization and Formatting**: For tasks involving data visualizations (e.g., scatter plots, fitted lines), assign the creation to the **Student Agent** with instructions on tools (Python, Matplotlib) and style requirements (e.g., PRL-compliant charts with clear legends, appropriate fonts, fitted lines passing through the origin). Assign review and optimization suggestions to the **Mentor Agent**, ensuring the visualizations meet academic standards.
6. **Handling User Feedback** ......
7. **File and Project Management** ......
8. **Task Tracking and Updates** ......
9. **Default Tools and Standards** ......
10. ......

Leader Agent Prompt

The prompts above represent an initial selection and are not yet complete. For improved stability and the full, continuously updated version, please refer to the future open-source GitHub repository: https://github.com/xqh19970407/PhysAgent

In the context of multi-agent systems, automated platforms demonstrate significant potential in task planning, code generation, and API development. As shown in Table 1, conventional frameworks such as AutoGPT and LangChain primarily focus on general-purpose task execution and code processing, lacking domain-specific capabilities for scientific or engineering applications. In contrast, MetaGPT supports automatic generation of product requirement documents (PRDs), technical design plans, and API interfaces, along with comprehensive code generation, precompilation execution, and role-based task management [28]. Building upon these general capabilities, PhysAgent further integrates domain-specific toolkits, hypothesis generation, data analysis, physical law derivation, and complex simulations (e.g., raindrop flow), demonstrating end-to-end autonomous reasoning and experimental planning from theoretical hypotheses to numerical validation.

Capability	AutoGPT	LangChain	MetaGPT	PhysAgent
PRD generation	×	×	✓	✓
Technical design generation	×	×	✓	✓
API interface generation	×	×	✓	✓
Code generation	✓	✓	✓	✓
Precompilation execution	×	×	✓	✓
Role-based task management	×	×	✓	✓
Code review	×	×	✓	✓
Domain-specific tool integration	×	×	×	✓
Hypothesis generation	×	×	×	✓
Physical law derivation	×	×	×	✓
Complex simulation	×	×	×	✓

Table 1. Comparison of capabilities for MetaGPT, PhysAgent, and other approaches. "✓" indicates the presence of a specific feature, "×" its absence.

To further clarify the novelty of PhysAgent, we compare it with a representative multi-agent framework, MetaGPT. The comparison in Table 2 highlights the key differences between the MetaGPT framework for software development and our PhysAgent for scientific discovery in physics. While MetaGPT relies on roles such as product manager, architect, and engineer to coordinate software engineering workflows, PhysAgent introduces domain-specialized agents (mentor, student, leader) to enable hypothesis-driven reasoning, technical execution, and dynamic task optimization. Moreover, the workflow evolves from sequential task decomposition to one that integrates Socratic questioning and domain-specific tools (e.g., Quantum ESPRESSO, VASP). Consequently, the outputs shift from conventional software artifacts to physics discoveries, including the derivation of physical laws and complex simulations. This comparison demonstrates the adaptability of multi-agent systems and highlights their potential to extend beyond software engineering into scientific research.

Component Function	MetaGPT (Software Development)	PhysAgent (Physics Discovery)
Agent Roles	Product Manager, Architect, Engineer , QA Engineer...	Mentor, Student, Leader
Workflow Mechanism	SOP for sequential task decomposition; Publish–Subscribe message pool	SOP with Socratic questioning; Integration of domain tools (e.g., Quantum ESPRESSO, VASP)
Task Handling	Executable feedback for code debugging; Handles user stories, APIs, documentation	Executable feedback for simulations; Handles hypothesis generation, data analysis, law derivation
Tool	Basic code execution, web search	Physics-specific: first-principles simulations, *ab initio* calculations (e.g., band gap)
Output	Software artifacts (e.g., data structures, competitive analysis)	Physics outputs (e.g., deriving Kepler's laws from data, raindrop flow simulation)

Table 2. Comparison between MetaGPT (Software Development) and PhysAgent (Physics Discovery)

Appendix

Representative Outputs for Each Role in PhysAgent

The following appendix presents representative outputs generated by the agents. These include: (i) selected comments by the Mentor Agent during the derivation of Newton's Second Law; (ii) improved visualization results from the Student Agent based on feedback from the Mentor Agent; and (iii) the DFT (Density Functional Theory) computational workflow designed by the Student Agent.

1. Clearly specify that "change in state of motion" refers to changes in the magnitude or direction of velocity to avoid confusion with acceleration.

2. Avoid introducing the concept of acceleration in assumptions; express in terms of velocity changes, emphasizing derivation from experiments rather than presupposition.

3. Encourage hypotheses regarding the relationship between the direction of force and the direction of velocity change to lay the groundwork for subsequent vector analysis.

4. Quantify and control the influence of friction, or compensate through control experiments or inclined track setups.

5. Recommend using spring dynamometers or relative force units instead of assuming gravitational constants for force measurements.

6. Encourage experimental designs involving forces in different directions and varying mass to enhance data diversity.

7. Clearly state the temporal and spatial resolution of data recordings and quantify instrument errors.

8. Introduce mechanisms for detecting data anomalies, such as checks for velocity continuity and energy conservation.

9. Apply smoothing or fitting methods to mitigate noise in differential calculations.

10. Include dimensional analysis to validate the physical consistency of derived formulas.

11. Supplement regression analysis with confidence intervals and residual tests to improve statistical robustness.

12. Eliminate alternative mathematical relationships before drawing conclusions and validate using dimensional analysis.

13. Clearly articulate the applicable conditions of physical laws, such as inertial frames and non-relativistic contexts.

14. Avoid using formulas that presuppose $F = ma$ to define acceleration, ensuring logical consistency.

15. All measuring tools should prioritize observable quantities, with explanations for the principles of indirect measurements.

16. Recommend referencing historical experimental methods (e.g., Galileo's inclined plane and Atwood machine) to optimize experimental design.

Mentor Agent Output (Selected Mentor Agent Comments During Newton's Second Law Derivation)

1. I adopted a no-intercept linear regression as per the Mentor Agent’s requirement, ensuring the fitting line passes through the origin to better align with physical principles.
2. I improved the color scheme by using shades of a single color family, making the visualization more professional and visually coherent.
3. I adjusted the font to Times New Roman, thickened axis lines, and added dashed grid lines to conform to PRL style guidelines.
4. I optimized labels by using physical symbols and standardized units, and displayed the fitting formula directly in the legend to enhance scientific clarity.
5. I simplified the overall chart layout by removing redundant titles, maintaining a clean and easily readable figure.

Student Agent Output (Improvements Following Mentor Agent's Feedback on Visualization)

The following is the DFT calculation workflow for GaAs designed based on user requirements:

1. Install Quantum ESPRESSO First, ensure that the Quantum ESPRESSO software is properly installed. If any issues arise during installation or configuration, it is recommended to search online for solutions or seek assistance from experienced users to ensure the smooth execution of subsequent calculations.

2. Create GaAs Crystal Structure This step generates the GaAs crystal structure for subsequent first-principles calculations. Using the ‘bulk‘ function from the ASE library, a zincblende structure of GaAs is constructed with a lattice constant of 5.65 Å. This produces an ‘atoms‘ object containing all atomic positions, element types, and unit cell information, serving as the fundamental input for Quantum ESPRESSO calculations.

3. Set up Quantum ESPRESSO Calculation Environment This step configures the execution environment required for Quantum ESPRESSO calculations. The paths to the Quantum ESPRESSO executable and the pseudopotential directory (e.g., ‘/yourpath//PhysAgent/QE/qe-7.4.1/pseudo‘) are specified. An ‘EspressoProfile‘ object is created to register the ‘pw.x‘ executable and pseudopotential directory, providing an interface for invoking Quantum ESPRESSO in subsequent steps.

4. Perform SCF Calculation This step carries out a self-consistent field (SCF) calculation to obtain a converged electron density and total energy. An ‘Espresso‘ calculator is instantiated with the ‘control‘ section set to an SCF task and output directory as ‘./out‘. The ‘system‘ section defines a plane-wave cutoff energy of 40 Ry and a charge density cutoff of 320 Ry, employing Gaussian smearing with a ‘degauss‘ value of 0.01. The ‘electrons‘ section sets a convergence threshold of 1e-6. Pseudopotential files for Ga and As in UPF format are specified, and a Monkhorst-Pack k-point grid of 8 is used. The calculator is attached to the ‘atoms‘ object, and the calculation is executed, printing the total energy in the format ″SCF Total energy: X.XXXXXX eV″.

5. Perform Band Structure Calculation This step computes the band structure of GaAs along high-symmetry paths. The ASE ‘bandpath‘ function generates a standard high-symmetry k-point path (e.g., GXWKGLUWLK) with 100 interpolated k-points, resulting in an array of k-point coordinates. An ‘Espresso‘ calculator is configured for a ‘bands‘ type calculation, preserving the same pseudopotentials and energy parameters as the SCF step, and using the generated k-point path with k-point shifts disabled. After attaching the calculator to the ‘atoms‘ object and invoking ‘atoms.get_bands()‘, the band structure calculation is performed, and upon completion, ″Bands Band structure calculation completed.″ is printed.

6. Perform NSCF Calculation This step performs a non-self-consistent field (NSCF) calculation to obtain a high-density k-point electronic structure for density of states analysis. A new ‘Espresso‘ calculator is configured with the task type set to ‘nscf‘, keeping other parameters consistent with the SCF calculation. The k-point grid is increased to 12 for enhanced resolution. The calculator is attached to the ‘atoms‘ object, the calculation is executed to retrieve the ‘energy‘ property, and the process ends with ″NSCF NSCF calculation completed.″.

7. Perform DOS Calculation This step computes the total density of states (DOS) of GaAs. An input file named ‘dos.in‘ is created, specifying the output directory as ‘./out‘, the prefix as ‘gaas‘, and the output file as ‘gaas.dos‘. The energy range is set from -15.0 eV to 15.0 eV with an energy step of 0.01 eV. Quantum ESPRESSO’s ‘dos.x‘ program is called to read the NSCF outputs and calculate the DOS. Upon completion, "DOS DOS calculation completed." is printed.

8. Perform PDOS Calculation This step calculates the projected density of states (PDOS) for Ga and As atoms to analyze the contribution of different atomic orbitals to the electronic structure. An input file named ‘projwfc.in‘ is generated, specifying the output directory as ‘./out‘, the prefix as ‘gaas‘, and the output file as ‘gaas.pdos‘, with the same energy range as the DOS calculation. Gaussian smearing parameters are set with ‘ngauss=0‘ and ‘degauss=0.01‘. The ‘projwfc.x‘ program is invoked to perform the calculation, and upon completion, "PDOS PDOS calculation completed." is printed.

9. Plot Band Structure and DOS/PDOS Combined Graph This step produces publication-quality plots of the band structure and density of states. Matplotlib is configured with academic-style settings (e.g., serif fonts, customized ticks). A figure with two subplots is created: the left subplot displays the band structure, and the right subplot shows the DOS and PDOS, sharing the y-axis with a horizontal spacing of 0.05. The band structure plot reads the high-symmetry path and labels, plotting each band’s energy relative to the Fermi level, with dashed lines at special k-points and a red dashed line at the Fermi level. The DOS subplot plots the total DOS, and if available, the PDOS for Ga (steel blue dashed line) and As (seagreen dash-dot line). Legends, axis labels, and titles are properly set. The final plot is saved as a high-resolution SVG file named ‘gaas_bands_dos_pdos_prl_style.svg‘.

10. Convert Charge Density Data to CUBE Format This step converts the computed charge density data into CUBE format for 3D visualization. A ‘pp.in‘ input file is written specifying the prefix as ‘gaas‘, output directory as ‘./out‘, plot type as charge density (‘plot_num=0‘), and output format as CUBE file (‘output_format=6‘). The output file is named ‘charge_density.cube‘. The Quantum ESPRESSO ‘pp.x‘ utility is invoked for the conversion, and upon success, the message "charge_density.cube file has been generated." is printed.

11. Plot Combined Charge Density Maps (2D Slice + 3D Isosurfaces) This step reads the charge density data from the CUBE file and generates combined visualizations consisting of a 2D slice and 3D isosurface plots. A figure with two subplots is created: the left subplot for 3D isosurfaces and the right for the 2D slice. A logarithmic color normalization (LogNorm) is applied to enhance the visibility of low-density regions. The 2D slice is taken at the midpoint along the z-axis and displayed using ‘imshow‘ with a colorbar and axis labels. For the 3D plot, two intermediate isosurface levels within the log-scaled data range are selected; isosurfaces are generated using the marching cubes algorithm and rendered with ‘plot_trisurf‘. Atomic positions are overlaid as scatter points (blue for Ga and red for As), along with lattice vector arrows. Appropriate viewing angles and legends are set. The final visualization is saved as ‘combined_charge_density_optimized.svg‘, suitable for publication or presentation.

doi.org/10.32388/J2MXUW

Student Agent Output (Design the DFT (Density Functional Theory) computational workflow.)

Final Code Implementation in PhysAgent

Here is the code developed by the Student Agent to implement DFT calculations, along with a portion of the visualization code. This version incorporates feedback from both the Mentor Agent and human users, representing the final refined output.

```
"""
====================================================================
GaAs Electronic Structure Calculation Workflow using ASE + QE
====================================================================

Author   : Student Agent
Date     : July 20, 2025
Version  : 1.0
Language: Python 3.x

Description:
------------
This script provides a modular workflow for performing first-principles
calculations of the band structure, total density of states (DOS), and
projected density of states (PDOS) of gallium arsenide (GaAs) using
Quantum ESPRESSO via the ASE (Atomic Simulation Environment) interface.

Key features:
- SCF, NSCF, and band structure calculation
- DOS and PDOS post-processing
- Pseudopotential-based DFT input using PBE functionals
- Bandpath definition based on high-symmetry points

Modules:
--------
- create_structure(): Generate bulk GaAs structure
- run_scf(): Perform SCF calculation
- run_band_structure(): Compute band energies along high-symmetry path
- run_nscf(): Perform NSCF calculation for DOS
- run_dos(): Run DOS post-processing via `dos.x`
- run_pdos(): Run PDOS post-processing via `projwfc.x`

Requirements:
-------------
- ASE
- Quantum ESPRESSO (pw.x, dos.x, projwfc.x)
- NumPy
- Pseudopotentials from PSLibrary

"""

# file: gaas_workflow.py

import os
import numpy as np
from ase.build import bulk
from ase.calculators.espresso import Espresso, EspressoProfile
from ase.dft.kpoints import bandpath
```

```python
import xml.etree.ElementTree as ET

def create_structure():
    return bulk('GaAs', crystalstructure='zincblende', a=5.65)

def run_scf(atoms, profile):
    scf_calc = Espresso(
        profile=profile,
        input_data={
            'control': {'calculation': 'scf', 'prefix': 'gaas', 'outdir': './out'},
            'system': {
                'ecutwfc': 40, 'ecutrho': 320,
                'occupations': 'smearing', 'smearing': 'gaussian', 'degauss': 0.01
            },
            'electrons': {'conv_thr': 1e-6}
        },
        pseudopotentials={
            'Ga': 'Ga.pbe-dn-rrkjus_psl.1.0.0.UPF',
            'As': 'As.pbe-n-rrkjus_psl.1.0.0.UPF'
        },
        kpts=(8, 8, 8)
    )
    atoms.calc = scf_calc
    energy = atoms.get_potential_energy()
    print(f"[SCF] Total energy: {energy:.6f} eV")

def run_band_structure(atoms, profile):
    path = bandpath('GXWKGLUWLK', atoms.cell, npoints=100)
    kpts_bands = np.zeros((len(path.kpts), 4))
    kpts_bands[:, :3] = path.kpts

    bands_calc = Espresso(
        profile=profile,
        input_data={
            'control': {'calculation': 'bands', 'prefix': 'gaas', 'outdir': './out'},
            'system': {
                'ecutwfc': 40, 'ecutrho': 320,
                'occupations': 'smearing', 'smearing': 'gaussian', 'degauss': 0.01
            },
            'electrons': {'conv_thr': 1e-6}
        },
        pseudopotentials={
            'Ga': 'Ga.pbe-dn-rrkjus_psl.1.0.0.UPF',
            'As': 'As.pbe-n-rrkjus_psl.1.0.0.UPF'
        },
        kpts=kpts_bands,
        koffset=False
    )
    atoms.calc = bands_calc
    atoms.calc.calculate(atoms, properties=['bands'], system_changes=['
        cell'])
    print("[Bands] Band structure calculation completed.")

def run_nscf(atoms, profile):
    nscf_calc = Espresso(
        profile=profile,
        input_data={
            'control': {'calculation': 'nscf', 'prefix': 'gaas', 'outdir': './out'},
            'system': {
                'ecutwfc': 40, 'ecutrho': 320,
                'occupations': 'smearing', 'smearing': 'gaussian', 'degauss': 0.01
            },
            'electrons': {'conv_thr': 1e-6}
        },
        pseudopotentials={
            'Ga': 'Ga.pbe-dn-rrkjus_psl.1.0.0.UPF',
            'As': 'As.pbe-n-rrkjus_psl.1.0.0.UPF'
        },
        kpts=(12, 12, 12)
    )
    atoms.calc = nscf_calc
    atoms.calc.calculate(atoms, properties=['energy'], system_changes=['cell'])
    print("[NSCF] NSCF calculation completed.")

def run_dos():
    with open('dos.in', 'w') as f:
        f.write("""
&dos
  outdir = './out',
  prefix = 'gaas',
  fildos = 'gaas.dos',
  emin = -15.0,
  emax = 15.0,
  deltae = 0.01,
/
""")
    os.system('dos.x < dos.in > dos.out')
    print("[DOS] DOS calculation completed.")

def run_pdos():
    with open('projwfc.in', 'w') as f:
        f.write("""
&projwfc
  outdir = './out',
  prefix = 'gaas',
  filpdos = 'gaas.pdos',
  emin = -15.0,
  emax = 15.0,
  deltae = 0.01,
  ngauss = 0,
  degauss = 0.01,
/
""")
    os.system('projwfc.x < projwfc.in > projwfc.out')
    print("[PDOS] PDOS calculation completed.")

def main():
    pseudo_dir = '/yourpath'
    profile = EspressoProfile(command='pw.x', pseudo_dir=pseudo_dir)
    atoms = create_structure()

    run_scf(atoms, profile)
    run_band_structure(atoms, profile)
    run_nscf(atoms, profile)
    run_dos()
    run_pdos()

if __name__ == "__main__":
    main()
```

Listing 1. GaAs Electronic Structure Calculation Workflow using ASE + QE

```
"""
======================================================================
Charge Density Visualization Workflow from CUBE File
======================================================================

Author   : Student Agent
Date     : July 20, 2025
Version : 1.0
Language: Python 3.x

Description:
------------
This script reads charge density data from a Gaussian CUBE file and
    generates
combined 2D slice and 3D isosurface visualizations using matplotlib and
    scikit-image.

Key Features:
- Robust parsing of CUBE format (including atomic metadata and volumetric
    grid)
- Automatic handling of mismatched or missing voxel data
- Visualization includes:
    - XY-plane density slice (log scale)
    - 3D charge isosurfaces with atomic positions
    - Lattice vectors

Requirements:
-------------
- NumPy
- Matplotlib
- scikit-image

Usage:
```

```python
------
Make sure a valid `charge_density.cube` file is in the working directory.
Run this script directly to generate `combined_charge_density_optimized.svg`.
"""

import os
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import LogNorm
from mpl_toolkits.axes_grid1 import make_axes_locatable
from skimage.measure import marching_cubes


def read_cube_file(filename):
    """Read and parse charge density from a CUBE file."""
    with open(filename, 'r') as f:
        # Skip comment lines
        f.readline()
        f.readline()

        # Read number of atoms and origin
        line = f.readline().split()
        natoms = int(line[0])
        origin = np.array([float(x) for x in line[1:4]])

        # Read voxel grid and spacing
        nx, dx = int(f.readline().split()[0]), np.array([float(x) for x in f.readline().split()[1:4]])
        ny, dy = int(f.readline().split()[0]), np.array([float(x) for x in f.readline().split()[1:4]])
        nz, dz = int(f.readline().split()[0]), np.array([float(x) for x in f.readline().split()[1:4]])

        # Read atomic information
        atoms = []
        for _ in range(abs(natoms)):
            parts = f.readline().split()
            atomic_number = int(parts[0])
            charge = float(parts[1])
            position = np.array([float(x) for x in parts[2:5]])
            atoms.append((atomic_number, charge, position))

        # Read all remaining density values
        data = np.zeros((nx, ny, nz))
        raw_values = []
        for line in f:
            raw_values.extend([float(val) for val in line.split()])

        expected = nx * ny * nz
        if len(raw_values) != expected:
            print(f"[Warning] Data size mismatch: expected {expected}, got {len(raw_values)}")
            raw_values = raw_values[:expected]

        # Populate 3D array
        idx = 0
        for i in range(nx):
            for j in range(ny):
                for k in range(nz):
                    data[i, j, k] = raw_values[idx] if idx < len(raw_values) else 0.0
                    idx += 1

    return data, atoms, origin, (dx, dy, dz)


def create_combined_plot(data, atoms_info, origin, grid_vectors):
    """Create combined 2D slice and 3D isosurface visualization."""
    print("[Info] Generating visualizations...")
    plt.figure(figsize=(20, 8))

    # Log scale normalization
    vmin = data[data > 0].min()
    vmax = data.max()
    norm = LogNorm(vmin=vmin, vmax=vmax)

    # ---------- 2D Slice ----------
    ax1 = plt.subplot(122)
    slice_xy = data[:, :, data.shape[2] // 2]
    im = ax1.imshow(slice_xy.T, cmap='viridis', origin='lower', norm=norm)
    ax1.set_title('XY Plane Charge Density Slice')
    ax1.set_xlabel('X (Grid Points)')
    ax1.set_ylabel('Y (Grid Points)')

    divider = make_axes_locatable(ax1)
    cax = divider.append_axes("right", size="5%", pad=0.1)
    plt.colorbar(im, cax=cax, label='Charge Density (e/Bohr3)')

    # ---------- 3D Isosurfaces ----------
    ax2 = plt.subplot(121, projection='3d')
    levels = np.logspace(np.log10(vmin), np.log10(vmax), 4)[1:3]
    colors = plt.cm.viridis(norm(levels))

    for i, level in enumerate(levels):
        try:
            verts, faces, _, _ = marching_cubes(data, level)
            ax2.plot_trisurf(verts[:, 0], verts[:, 1], faces, verts[:, 2],
                             color=colors[i], alpha=0.4 + i * 0.2, lw=0.5)
        except Exception as e:
            print(f"[Warning] Skipping isosurface at level {level:.2e}: {e}")

    # ---------- Atoms ----------
    element_colors = {31: 'blue', 33: 'red'}  # Ga and As
    element_labels = {31: 'Ga', 33: 'As'}
    added = set()

    for atomic_num, _, pos in atoms_info:
        color = element_colors.get(atomic_num, 'gray')
        label = element_labels.get(atomic_num, 'Unknown') if atomic_num not in added else None
        ax2.scatter(*pos, c=color, s=150, edgecolor='k', depthshade=False, label=label)
        added.add(atomic_num)

    # ---------- Lattice Vectors ----------
    for i, vec in enumerate(grid_vectors):
        ax2.quiver(*origin, *(vec * data.shape[i]), color='black', arrow_length_ratio=0.1, lw=2)

    ax2.set_xlabel('X (Bohr)', labelpad=15)
    ax2.set_ylabel('Y (Bohr)', labelpad=15)
    ax2.set_zlabel('Z (Bohr)', labelpad=15)
    ax2.zaxis._axinfo['juggled'] = (1, 2, 0)
    ax2.set_title('3D Charge Density Isosurfaces')
    ax2.legend(title='Atoms', loc='upper right', bbox_to_anchor=(1.15, 0.9))
    ax2.view_init(elev=30, azim=45)

    plt.savefig('combined_charge_density_optimized.svg', dpi=300, bbox_inches='tight')
    plt.show()


if __name__ == '__main__':
    # Load and visualize charge density from CUBE file
    cube_file = 'charge_density.cube'
    if not os.path.isfile(cube_file):
        raise FileNotFoundError(f"CUBE file not found: {cube_file}")

    charge_data, atoms_info, origin, grid_vectors = read_cube_file(cube_file)
    print(f"[Info] Charge density shape: {charge_data.shape}")
    create_combined_plot(charge_data, atoms_info, origin, grid_vectors)
```

Listing 2. GaAs Charge Density Visualization

Numerical Methods and Parameters

The thin film evolution on the window was solved using a finite difference method on a rectangular grid of $nx = 400$ and $ny = 200$, with spatial resolutions $dx = L/nx = 0.0025$ m and $dy = W/ny = 0.0025$ m. The time step was set to $dt = 0.0002$ s, resulting in a total of $nt = T/dt = 50000$ steps.

The Laplacian operator ∇^2 was constructed as a sparse matrix:

$$D_{2x} = \frac{1}{dx^2}\begin{pmatrix} 1 & -2 & 1 \end{pmatrix}, \quad \nabla^2 = I_y \otimes D_{2x} + D_{2y} \otimes I_x,$$

where I_x and I_y are identity matrices of appropriate sizes. The flux divergence was computed using central differences, and boundary conditions were either periodic or zero-flux, implemented implicitly via truncation.

Flow computation was restricted to regions where $h > h_{\mathrm{min\backslash\_move}} = 1 \times 10^{-5}$ m to avoid numerical diffusion. Visualization was performed using Matplotlib animations, displaying $\sqrt{h + 5 \times 10^{-5} |\nabla h|}$ to highlight the film edges.

The physical and numerical parameters used in the simulation are summarized in Table 3.

Parameter	Value
μ (viscosity)	1×10^{-3} Pa·s
ρ (density)	1000 kg/m³
g (gravity)	9.81 m/s²
σ (surface tension)	0.036 N/m
τ_{air} (air shear stress)	36600 Pa
$L \times W$ (window size)	1.0×0.5 m
T (simulation time)	10 s
$branch\_prob$ (branching probability)	0.85
$drop\_thresh$ (dripping threshold)	8×10^{-2} m
$evap\_rate$ (evaporation rate)	1×10^{-4} m/s
nx, ny (grid points)	400, 200
dt (time step)	0.0002 s

Table 3. Physical and numerical parameters

Statements and Declarations

Code availability

To promote transparency and reproducibility, all code, datasets, and related materials from this study are publicly accessible at the following open repository: https://github.com/xqh19970407/PhysAgent.

Conflicts of interest

The authors declare no competing interests.

Acknowledgements

This work was financially supported by the National Natural Science Foundation of China (Grant No.62476278, No.12434009, and No.12204533). Z.Y.L. was also supported by the National Key R&D Program of China (Grant No. 2024YFA1408601). Computational resources have been provided by the Physical Laboratory of High Performance Computing at Renmin University of China.

Footnotes

[1] https://www.quantum-espresso.org/

[2] https://www.vasp.at/

[3] https://openai.com/chatgpt/overview/

[4] https://www.kimi.com/

[5] https://www.doubao.com/chat/

[6] https://www.llama.com/

[7] https://chat.deepseek.com/

References

1. ^Xun Xiao, Niansheng Xu, Xueyu Tian, Tiankai Zhang, Bingzheng Wang, et al. (2025). "Aqueous-based recycling of perovskite photovoltaics." Nature. **638**(8051):670–675. doi:10.1038/s41586-024-08408-7.
2. ^Zirui Gao, Aowen Li, Xingwu Liu, Mi Peng, Shixiang Yu, et al. (2025). "Shielding Pt/γ-Mo2N by inert nano-overlays enables stable H2 production." Nature. **638**(8051):690–696. doi:10.1038/s41586-024-08483-w.

3. ^Hualei Sun, Mengwu Huo, Xunwu Hu, Jingyuan Li, Zengjia Liu, et al. (2023). "Signatures of superconductivity near 80 K in a nickelate under high pressure." Nature. **621**(7979):493–498. doi:10.1038/s41586-023-06408-7.
4. ^Guangdi Zhou, Wei Lv, Heng Wang, Zihao Nie, Yaqi Chen, et al. (2025). "Ambient-pressure superconductivity onset above 40 K in (La,Pr)3Ni2O7 films." Nature. **640**(8059):641–646. doi:10.1038/s41586-025-08755-z.
5. ^He Jiang, Xiankun Zhang, Kuanglei Chen, Xiaoyu He, Yihe Liu, et al. (2025). "Two-dimensional Czochralski growth of single-crystal MoS2." Nat Mater. **24**(2):188–196. doi:10.1038/s41563-024-02069-7.
6. a, b, c, d OpenAI (2024). "GPT-4 technical report." arXiv preprint arXiv:230308774. https://arxiv.org/abs/2303.08774.
7. ^Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, et al. (2024). "Accurate structure prediction of biomolecular interactions with AlphaFold 3." Nature. **630**(8016):493–500. doi:10.1038/s41586-024-07487-w.
8. ^Jonathan Schmidt, Tiago FT Cerqueira, Aldo H Romero, Antoine Loew, Fabian Jäger, et al. (2024). "Improving machine-learning models in materials science through large datasets." Mat Today Phys. **48**:101560. doi:https://doi.org/10.1016/j.mtphys.2024.101560.
9. ^Amil Merchant, Simon Batzner, Samuel S Schoenholz, Muratahan Aykol, Gowoon Cheon, et al. (2023). "Scaling deep learning for materials discovery." Nature. **624**(7990):80–85. doi:10.1038/s41586-023-06735-9.
10. ^Luis Barroso-Luque, Muhammed Shuaibi, Xiang Fu, Brandon M Wood, Misko Dzamba, et al. (2024). "Open materials 2024 (OMat24) inorganic materials dataset and models." arXiv preprint arXiv:241012771. https://arxiv.org/abs/2410.12771.
11. ^Xiao-Qi Han, Peng-Jie Guo, Ze-Feng Gao, Hao Sun, Zhong-Yi Lu. (2025). "InvDesFlow-AL: Active learning-based workflow for inverse design of functional materials." arXiv preprint arXiv:250509203. https://arxiv.org/abs/2505.09203.
12. ^Claudio Zeni, Robert Pinsler, Daniel Zügner, Andrew Fowler, Matthew Horton, et al. (2025). "A generative model for inorganic materials design." Nature. **639**(8055):624–632. doi:10.1038/s41586-025-08628-5.
13. ^Rui Jiao, Wenbing Huang, Peijia Lin, Jiaqi Han, Pin Chen, et al. (2023). "Crystal structure prediction by joint equivariant diffusion on lattices and fractional coordinates." ICLR. https://openreview.net/forum?id=VPByphdu24j.
14. ^Xiao-Qi Han, Zhenfeng Ouyang, Peng-Jie Guo, Hao Sun, Ze-Feng Gao, et al. (2025). "InvDesFlow: An AI-Driven Materials Inverse Design Workflow to Explore Possible High-Temperature Superconductors." Chin Phys Lett. **42**(4):047301. doi:10.1088/0256-307X/42/4/047301.

15. ^*Xiao-Qi Han, Xin-De Wang, Meng-Yuan Xu, Zhen Feng, Bo-Wen Yao, et al. (2025). "AI-Driven Inverse Design of Materials: Past, Present, and Future." Chin Phys Lett.* ***42****(2):027403. doi:10.1088/0256-307X/42/2/027403.*
16. ^*Tian Xie, Jeffrey C Grossman. (2018). "Crystal Graph Convolutional Neural Networks for an Accurate and Interpretable Prediction of Material Properties." Phys Rev Lett.* ***120****:145301. doi:10.1103/PhysRevLett.120.145301.*
17. ^*Kamal Choudhary, Kevin Garrity. (2022). "Designing high-TC superconductors with BCS-inspired screening, density functional theory, and deep-learning." Npj Comput Mater.* ***8****(1):244. doi:10.1038/s41524-022-00933-1.*
18. ^*Fernando Gómez-Ortiz, Aldo H Romero, Eric Bousquet. (2025). "Pathways to crystal chirality: An algorithm to identify displacive chiral phase transitions." Phys Rev B.* ***112****:014107. doi:10.1103/n4sh-7x7c.*
19. ^*Chengkai Zhu, Yin Mo, Yu-Ao Chen, Xin Wang. (2024). "Reversing Unknown Quantum Processes via Virtual Combs for Channels with Limited Information." Phys Rev Lett.* ***133****:030801. doi:10.1103/PhysRevLett.133.030801.*
20. ^*Yi-Jun Chang, Jia-Hui Zhang, Yong-Heng Lu, Ying-Yue Yang, Feng Mei, et al. (2025). "Observation of Photonic Mobility Edge Phases." Phys Rev Lett.* ***134****:053601. doi:10.1103/PhysRevLett.134.053601.*
21. ^*Kaiwen Chen, Xiangqi Liu, Jiachen Jiao, Muyuan Zou, Chengyu Jiang, et al. (2024). "Evidence of Spin Density Waves in La3Ni2O7-δ." Phys Rev Lett.* ***132****:256503. doi:10.1103/PhysRevLett.132.256503.*
22. a, b*Yi-Heng Tian, Yin Chen, Jia-Ming Wang, Rong-Qiang He, Zhong-Yi Lu. (2024). "Correlation effects and concomitant two-orbital s±-wave superconductivity in La3Ni2O7 under high pressure." Phys Rev B.* ***109****:165154. doi:10.1103/PhysRevB.109.165154.*
23. ^*Jia-Ming Wang, Yin Chen, Yi-Heng Tian, Rong-Qiang He, Zhong-Yi Lu. (2025). "Low-energy interband Kondo bound states in orbital-selective Mott phases." Phys Rev B.* ***111****:155107. doi:10.1103/PhysRevB.111.155107.*
24. ^*Ru Zheng, Rong-Qiang He, Zhong-Yi Lu. (2023). "Natural orbitals renormalization group approach to a spin-½ impurity interacting with two helical liquids." Phys Rev B.* ***107****:115149. doi:10.1103/PhysRevB.107.115149.*
25. ^*P W Anderson. (1972). "More Is Different." Science.* ***177****(4047):393–396. doi:10.1126/science.177.4047.393.*
26. a, b, c*DeepSeek-AI (2025). "DeepSeek-V3 technical report." arXiv preprint arXiv:241219437. https://arxiv.org/abs/2412.19437.*
27. a, b*Bang Liu et al. (2025). "Advances and challenges in foundation agents: From brain-inspired intelligence to evolutionary, collaborative, and safe systems." arXiv preprint arXiv:250401990. https://arxiv.org/abs/2504.01990.*

28. a, b, c, d, e, f, g, h*Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, et al. (2024). "MetaGPT: Meta programming for a multi-agent collaborative framework." ICLR. https://openreview.net/forum?id=VtmBAGCN7o.*
29. a, b, c*Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, et al. (2023). "AutoGen: Enabling next-gen LLM applications via multi-agent conversation." arXiv preprint arXiv:230808155. https://arxiv.org/abs/2308.08155.*
30. a, b*Keivalya Pandya, Mehfuza Holia. (2023). "Automating customer service using LangChain: Building custom open-source GPT chatbot for organizations." arXiv preprint arXiv:231005421. https://arxiv.org/abs/2310.05421.*
31. ^*Jiaxi Cui, Zongjian Li, Yang Yan, Bohua Chen, Li Yuan. (2023). "ChatLaw: Open-source legal large language model with integrated external knowledge bases." CoRR. abs/2306.16092. doi:10.48550/arXiv.2306.16092.*
32. ^*Kexin Huang, Serena Zhang, Hanchen Wang, Yuanhao Qu, Yingzhou Lu, et al. (2025). "Biomni: A General-Purpose Biomedical AI Agent." bioRxiv. doi:10.1101/2025.05.30.656746.*
33. ^*Tal Ridnik, Dedy Kredo, Itamar Friedman. (2024). "Code generation with AlphaCodium: From prompt engineering to flow engineering." arXiv preprint arXiv:240108500. https://arxiv.org/abs/2401.08500.*
34. ^*Lianhao Zhou, Hongyi Ling, Keqiang Yan, Kaiji Zhao, Xiaoning Qian, et al. (2025). "Toward greater autonomy in materials discovery agents: Unifying planning, physics, and scientists." arXiv preprint arXiv:250605616. https://arxiv.org/abs/2506.05616.*
35. ^*Chengping Rao, Pu Ren, Qi Wang, Oral Buyukozturk, Hao Sun, et al. (2023). "Encoding physics to learn reaction–diffusion processes." Nat Mach Intell.* **5***(7):765–779. doi:10.1038/s42256-023-00685-7.*
36. ^*Zeyu Li, Wang Han, Yue Zhang, Qingfei Fu, Jingxuan Li, et al. (2024). "Learning spatiotemporal dynamics with a pretrained generative model." Nat Mach Intell.* **6***(12):1566–1579. doi:10.1038/s42256-024-00938-z.*
37. a, b*Duo Zhang, Xinzijian Liu, Xiangyu Zhang, Chengqian Zhang, Chun Cai, et al. (2024). "DPA-2: a large atomic model as a multi-task learner." Npj Comput Mater.* **10***(1):293. doi:10.1038/s41524-024-01493-2.*
38. ^*Antonio Sanna, Tiago FT Cerqueira, Yue-Wen Fang, Ion Errea, Alfred Ludwig, et al. (2024). "Prediction of ambient pressure conventional superconductivity above 80 K in hydride compounds." Npj Comput Mater.* **10** *(1):44. doi:10.1038/s41524-024-01214-9.*
39. ^*Kapildeb Dolui, Lewis J Conway, Christoph Heil, Timothy A Strobel, Rohit P Prasankumar, et al. (2024). "Feasible Route to High-Temperature Ambient-Pressure Hydride Superconductivity." Phys Rev Lett.* **132***:166001. doi:10.1103/PhysRevLett.132.166001.*

40. ^Chao Shen, Xiaoqi Han, Heng Cai, Tong Chen, Yu Kang, et al. (2025). "Improving the Reliability of Language Model-Predicted Structures as Docking Targets through Geometric Graph Learning." J Med Chem. **68**(2): 1956–1969. doi:10.1021/acs.jmedchem.4c02740.
41. ^Heng Cai, Chao Shen, Tianye Jian, Xujun Zhang, Tong Chen, et al. (2024). "CarsiDock: a deep learning paradigm for accurate protein–ligand docking and screening based on large-scale pre-training." Chem Sci. **15**:1449–1471. doi:10.1039/D3SC05552C.
42. a, bKimi Team (2025). "Kimi k1.5: Scaling reinforcement learning with LLMs." arXiv preprint arXiv:250112599. https://arxiv.org/abs/2501.12599.
43. ^Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, et al. (2024). "The llama 3 herd of models." arXiv preprint arXiv:2407.21783.
44. ^Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, et al. (2022). "Finetuned language models are zero-shot learners." arXiv preprint arXiv:210901652. https://arxiv.org/abs/2109.01652.
45. ^Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, et al. (2019). "Parameter-efficient transfer learning for NLP." ICML. **97**:2790–2799. https://proceedings.mlr.press/v97/houlsby19a.html.
46. ^Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, et al. (2023). "ReAct: Synergizing reasoning and acting in language models." arXiv preprint arXiv:221003629. https://arxiv.org/abs/2210.03629.
47. ^Wen Yang, Minpeng Liao, Kai Fan. (2025). "Markov chain of thought for efficient mathematical reasoning." arXiv preprint arXiv:241017635. https://arxiv.org/abs/2410.17635.
48. ^Kieran C R Fox, Nicholas S Fitz, Peter B Reiner. (2017). "The Multiplicity of Memory Enhancement: Practical and Ethical Implications of the Diverse Neural Substrates Underlying Human Memory Systems." Neuroethics. **10**(3):375–388. doi:10.1007/s12152-016-9282-7.
49. ^Alan Baddeley. (1992). "Working Memory." Science. **255**(5044):556–559. doi:10.1126/science.1736359.
50. ^Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, et al. (2022). "Training language models to follow instructions with human feedback." NeurIPS. **36**:2011. https://proceedings.neurips.cc/paper_files/paper/2022/file/b1efde53be364a73914f58805a001731-Paper-Conference.pdf.
51. ^Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, et al. (2020). "ALBERT: A Lite BERT for Self-Supervised Learning of Language Representations." arXiv preprint arXiv:190911942. https://arxiv.org/abs/1909.11942.
52. ^Lu Wang, Fangkai Yang, Chaoyun Zhang, Junting Lu, Jiaxu Qian, et al. (2025). "Large action models: From inception to implementation." arXiv preprint arXiv:241210047. https://arxiv.org/abs/2412.10047.

53. ^*Andreui Nikolaevich Kolmogorov. (1995). Turbulence: The Legacy of AN Kolmogorov. Cambridge university press.*

Declarations

Funding: This work was financially supported by the National Natural Science Foundation of China (Grant No.62476278, No.12434009, and No.12204533). Z.Y.L. was also supported by the National Key R&D Program of China (Grant No. 2024YFA1408601). Computational resources have been provided by the Physical Laboratory of High Performance Computing at Renmin University of China.

Potential competing interests: No potential competing interests to declare.

doi.org/10.32388/J2MXUW