



Cite this: DOI: 10.1039/d5dd00582e

ChatMat: a multi-agent chemist for autonomous material prediction and exploration

Shuai Lv, ^a Lei Peng, ^a Shizhe Jiao, ^b Yufan Yao, ^b Wentiao Wu ^a
and Wei Hu ^{*a}

Large language models (LLMs) have emerged as powerful tools for general-purpose tasks, but their performance in domain-specific applications, particularly in material property prediction, is still constrained by limited access to specialized knowledge. To overcome these challenges, we introduce ChatMat, an artificial intelligence (AI) chemist powered by a multi-agent system, capable of performing complex material property predictions with minimal human intervention. By leveraging LLMs such as GPT-4o or local foundation models, ChatMat autonomously interprets unstructured textual prompts, plans scientific procedures, and executes complex materials workflows from data retrieval to simulation and modeling with minimal human input. The system is orchestrated by a Manager agent, which interfaces with human researchers and coordinates four role-specific agents: Property Depositor, Computing Designer, density functional theory (DFT) Operator, and Machine Learning-driven Potential Energy Surface (ML-PES) Performer. This modular, multi-agent architecture enables the seamless integration of data-driven and physics-based techniques, establishing a robust, autonomous pipeline for material prediction and novel material exploration. We demonstrate the versatility and efficacy of ChatMat through four experimental tasks of increasing complexity, including structure generation, charge density distribution acquisition, database operation, and ML-PES construction. Furthermore, a series of quantitative evaluation metrics have been designed to benchmark its performance, illustrating ChatMat's reliability and adaptability across diverse materials domains. Our work bridges the gap between autonomous experimental research and computational science, showcasing the potential of domain-specific autonomous research to accelerate material prediction and exploration.

Received 23rd December 2025

Accepted 19th April 2026

DOI: 10.1039/d5dd00582e

rsc.li/digitaldiscovery

1 Introduction

Large language models (LLMs)^{1–3} have revolutionized natural language processing (NLP) by solving complex linguistic tasks and demonstrating cognition-like behaviors.⁴ With billions of parameters and pre-training on diverse corpora, LLMs demonstrate exceptional capabilities in semantic understanding, question answering, content generation, and code development across a wide range of tasks involving open human-machine interaction.^{3,5,6} LLMs now present unprecedented opportunities for automating reasoning in physical and chemical research, fostering a rapidly growing ecosystem of LLM-enabled tools.^{7–12} These advancements hold immense potential to accelerate scientific discovery through close collaboration between human researchers and LLMs.

Despite their remarkable success in open-domain applications, LLMs encounter challenges in specialized, domain-

specific tasks, such as material-property prediction.^{13,14} Unlike their rapid adoption in fields like medicine,^{15–18} LLMs have had a limited impact on property prediction.¹⁹ Four key challenges contribute to this gap: (i) material systems lack representations that convey their structural complexity in text-compatible forms, which restricts the expressiveness of LLMs;^{20,21} (ii) the scarcity of domain-specific datasets prevents LLMs from capturing the intricate physico-chemical relationships that define materials;^{7,22} (iii) laboratory workflows are still largely manual, and automation in materials research lags behind other scientific domains.²³ (iv) LLMs and machine learning-based models inherently struggle to incorporate true physical and chemical laws, limiting their ability to fully replicate real-world material behavior beyond surface-level correlations.^{24–26} These challenges underscore the need for an LLM-based agent capable of effectively predicting material properties while simultaneously exploring new materials and scientific hypotheses with minimal human intervention.

A promising solution lies in the emergence of autonomous agents that augment LLMs with domain-specific tools and plugins.²⁷ Early attempts to adapt large language models to scientific problems focused on prompt design enriched with

^aSchool of Artificial Intelligence and Data Science, University of Science and Technology of China, Hefei, Anhui 230026, China. E-mail: whuustc@ustc.edu.cn

^bHefei National Research Center for Physical Sciences at the Microscale, University of Science and Technology of China, Hefei, Anhui 230026, China



ChatMat MultiAgent

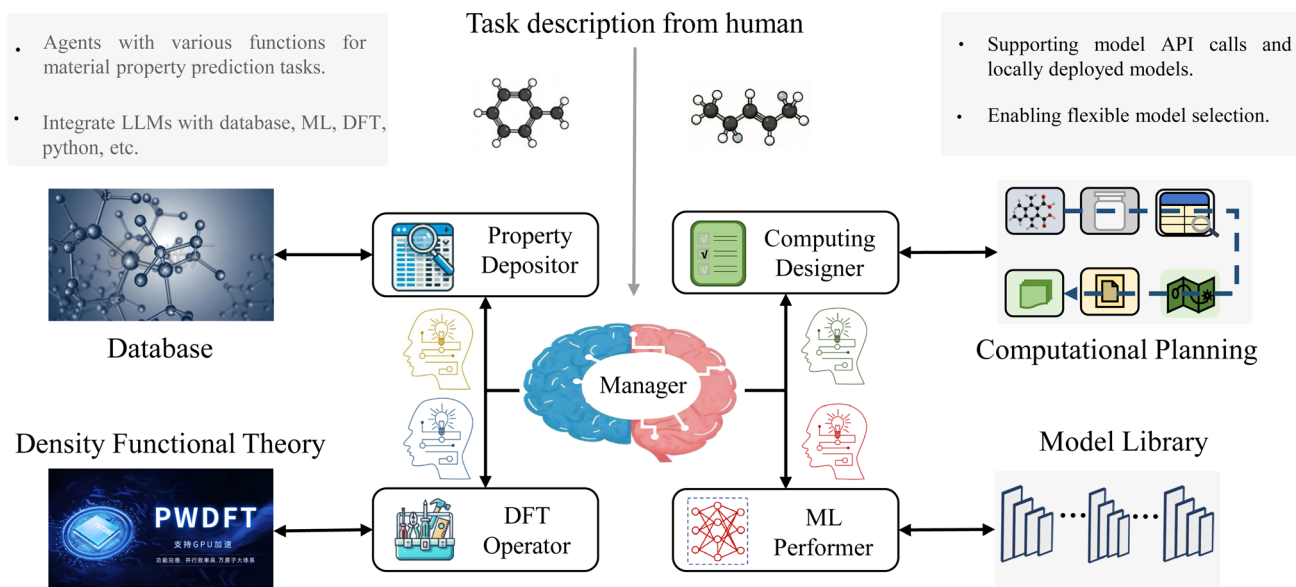


Fig. 1 Schematic representation of the ChatMat architecture. The system comprises a central Manager agent that coordinates four role-specific agents: Property Depositor, Computing Designer, DFT Operator, and ML Performer. Each agent integrates LLMs with specialized tools to execute automated material prediction and exploration tasks. ML: machine learning; DFT: density functional theory; LLM: large language model; API: application programming interface.

domain-specific knowledge.^{8,28–33} However, this approach remains constrained to static textual priors.³⁴ Furthermore, their execution of the computational or experimental procedures necessary for real-world discovery is often insufficient. For instance, early tool-integration strategies³³ exhibit an over-reliance on pre-existing databases, resulting in a lack of flexibility to dynamically route tasks and reduce computational overhead.

Moving beyond static prompting, several studies have proposed hierarchical agent architectures that integrate an LLM “brain” with specialized tools.^{35–39} Recent developments in autonomous agent-based systems have underscored the need for specialized tools that integrate LLMs with expert-driven workflows. Simultaneously, despite the advancements in LLM adoption, property prediction in computational materials science remains largely dependent on two computational paradigms, density functional theory (DFT) simulations and machine learning-driven potential energy surface (ML-PES).^{40,41} While DFT simulations, which solve the electronic Schrödinger equation, are widely applicable, they are computationally expensive.⁴² ML-PESs, while offering considerable speed advantages, often face limitations in generalization due to their dependence on training data, relying heavily on DFT baselines for accuracy.^{43,44} Therefore, a balance between these methods is essential for accurate materials property prediction.

To address these gaps, we introduce ChatMat, an autonomous multi-agent AI chemist that integrates DFT simulations and ML-PESs through a unified LLM interface. By seamlessly alternating between DFT and ML evaluations, ChatMat provides accurate and efficient property predictions without the need for

expert intervention. The system consists of specialized sub-agents: Property Depositor, Computing Designer, DFT Operator, and ML-PES Performer, which collaborate under a central Manager (Fig. 1) to plan, execute, and validate results. This division of labor ensures comprehensive, end-to-end knowledge of the property-prediction pipeline while maintaining flexibility for future expansions allowing for the integration of new tools with minimal engineering effort. In doing so, ChatMat reduces the barrier to accurate materials modeling, making it accessible to both specialists and non-experts alike, and markedly accelerates material discovery through both prediction and exploration.

2 Methods

2.1 Design of ChatMat

ChatMat leverages a decentralized multi-agent system comprising expert-designed agents specializing in physical chemistry, operating by prompting an LLM (GPT-4o in our experiments) with specific instructions concerning the task and the desired format, as outlined in ref. 45. The architecture of ChatMat consists of a central coordinating entity, the Manager agent, which interacts with human researchers and manages four role-specific autonomous agents: Property Depositor, Computing Designer, DFT Operator, and ML Performer. The Manager has a predefined system prompt, within which its role, tasks, specific functions, and interface descriptions for all entities it can access, as well as message formats for workflows and communications, are defined. This prompt-based multi-agent architecture offers high extensibility. Adding a new



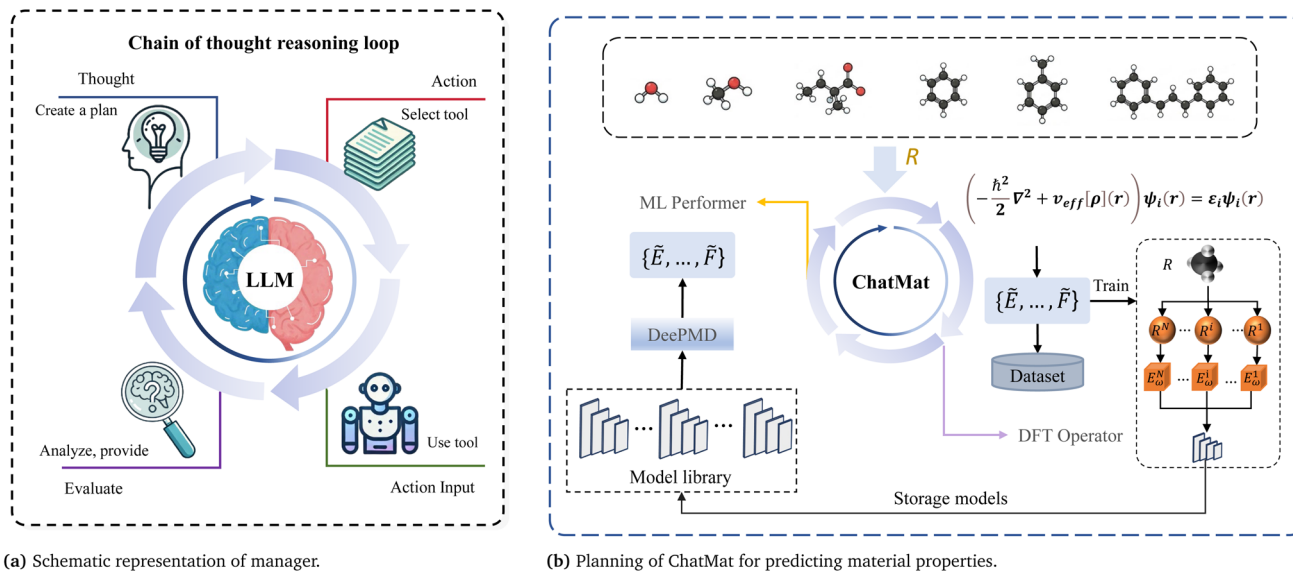


Fig. 2 Overview of the Manager's reasoning process and the computational planning workflow. (a) The "Chain of Thought" reasoning loop of the Manager based on an LLM, consisting of Thought, Action, Action Input, and Evaluate steps. (b) The decision-making logic of the Computing Designer agent, which determines whether to apply the ML or DFT method based on error thresholds. LLM: large language model; ML: machine learning; DFT: density functional theory.

agent to ChatMat involves simply defining its underlying Python function and updating the Manager's system prompt with the agent's description and usage examples, without the need for model retraining.

By leveraging the capabilities of the LLM and LangChain (<https://github.com/langchain-ai/langchain>), the Manager agent interacts with human researchers using natural language. It interprets the demand description, decomposes tasks, and instructs the role-specific agents to perform the specified operations autonomously, thereby facilitating an automated, decentralized experimental workflow based on the input description. As shown in Fig. 2(a), this multi-agent collaborative process requires the Manager's LLM to reason about the task's current state, assess its relevance to the final goal, and plan the next steps accordingly, thereby demonstrating its level of understanding. After reasoning in the "Thought" step, the Manager selects a specific subordinate agent (preceded by the keyword "Action") and generates the corresponding instruction for this agent (preceded by the keyword "Action Input"). The Manager's text generation then pauses, and the system delegates the execution to the requested agent using the provided input. The result is then returned to the Manager's LLM, preceded by the keyword "Evaluate", and the Manager proceeds to the "Thought" step again. This iterative multi-agent coordination continues until the final answer is reached. This reasoning loop operates synchronously in that the Manager halts its generation process while the worker agent performs the task and awaits the return signal. This design facilitates robust dynamic error handling. If a subordinate agent's execution fails, the error log is captured and returned as the "Evaluate" signal. The Manager then detects the failure and autonomously attempts to devise a corrective plan by either re-instructing the current agent or routing the task to an

alternative agent. A maximum retry limit (set to three in our experiments) is enforced to prevent infinite loops, ensuring that the system either self-corrects or halts to report persistent issues.

Each role-specific agent operates with its own LLM instance(s) to function autonomously within its designated domain (Fig. 1). The Property Depositor functions as an autonomous coding and retrieval agent to write Python code and mine the Material Property Database. The Computing Designer receives instructions from the Manager regarding intended experiments, plans computational tasks, and generates step-by-step procedures. The DFT Operator acts as an independent execution agent that analyzes the structural file, converts it into the required format for implementation, and generates the DFT software startup code, which it then initiates. The ML Performer searches for suitable pre-trained machine learning models in the Model Library, utilizing them as provided or autonomously enhancing them by expanding the model and incorporating additional training data.

ChatMat leverages the multi-agent architecture depicted in Fig. 1 to address this challenge. When material property queries are submitted by users in textual form, ChatMat responds by coordinating these specialized agents to provide a detailed, scientifically accurate description related to the material in question.

2.2 Generating experimental data from scientific retrieval

2.2.1 Property depositor. As shown in Fig. 3(a), unlike prior single-agent frameworks that rely on a solitary language model to sequentially invoke external tools, ChatMat utilizes a decentralized multi-agent paradigm. When users query information about specific materials, the central Manager agent does not execute searches directly. Instead, it delegates the task to the



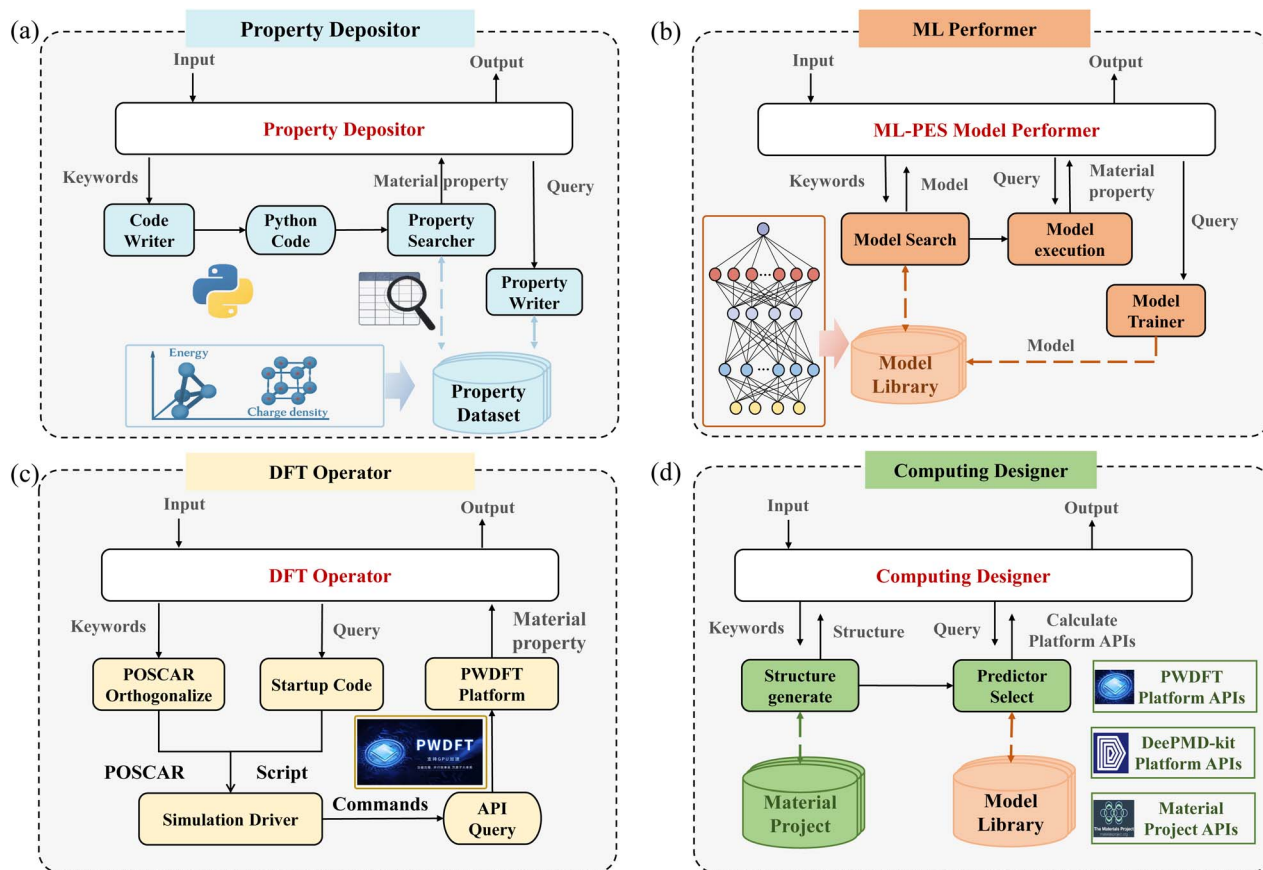


Fig. 3 Detailed architectures of the four role-specific agents. (a) Property Depositor. (b) ML Performer. (c) DFT Operator. (d) Computing Designer. Insets depict the specific underlying computational tools, APIs, and model architectures integrated into each agent's workflow. ML: machine learning; DFT: density functional theory; API: application programming interface; ML-PES: machine learning-driven potential energy surface (panel (c) is adapted and extended from our prior framework³³ to reflect the newly introduced multi-agent orchestration logic).

specialized Property Depositor agent. To retrieve data from pre-existing databases, the Property Depositor autonomously generates and executes Python code *via* a Code Interpreter. By decoupling this data-mining task from the main reasoning loop, the system mitigates the context-window exhaustion and tool overload inherent in single-agent architectures. Given the inherent time lag in LLM pre-training data, the Property Depositor is also equipped with SerpAPI to access the internet for up-to-date knowledge. This feature enables the model to access relevant data on a wide range of scientific topics and serves as a dynamic fallback option when the local databases lack sufficient information or when the Manager encounters an out-of-distribution query.

2.3 Generating experimental procedures from protocols

2.3.1 Computing designer. To ensure accurate predictions of material properties, the Computing Designer adopts a flexible approach to structure acquisition. It primarily relies on the Materials Project database⁴⁶ to retrieve reliable, standardized molecular structures from an extensive collection. To meet specific research needs, the agent also supports the integration of user-uploaded structural files and custom internal datasets, facilitating the study of novel systems that have not yet been

indexed in public repositories. Once the molecular structure is obtained from either source, the Computing Designer autonomously applies the Gram–Schmidt orthogonalization method to ensure consistency and enhance computational efficiency, effectively offloading this procedural overhead from the central Manager agent.

For material property queries where precalculated look-up table data are unavailable, computational simulations offer a valuable alternative. However, these simulations are time-consuming and resource-intensive. In previous single-agent systems, the LLM was tasked with manually selecting calculation tools, which frequently led to context dilution under complex tasks. The optimal resolution in ChatMat's decentralized architecture is to allow the Computing Designer to algorithmically judge and route the task to the appropriate specialized agent, as shown in Fig. 3(d). In this work, the Computing Designer determines whether to employ the DFT Operator or the ML Performer based on the following methodology.⁴⁷

Given a configuration \mathcal{R}_t , with t representing a continuous or discrete series of operations, we define the error indicator ϵ_t as the maximal standard deviation of the atomic force predicted by the model ensemble,



$$\varepsilon_t = \max_i \sqrt{\langle \|F_{w,i}(\mathcal{R}_t) - \langle F_{w,i}(\mathcal{R}_t) \rangle\|^2 \rangle} \quad (1)$$

where $F_{w,i}(\mathcal{R}_t) = -\nabla_i E_w(\mathcal{R}_t)$ denotes the force on the atom with index i predicted by the model E_w and ∇_i denotes the derivative with respect to the coordinate of the i -th atom. Both of the notations $\langle \dots \rangle$ in eqn (2) denote the expectation with respect to the ensemble of models and are estimated using the average of model predictions. For example, $\langle F_{w,i}(\mathcal{R}_t) \rangle$ is estimated using

$$\langle F_{w,i}(\mathcal{R}_t) \rangle = \frac{1}{N_m} \sum_{\alpha=1}^{N_m} F_{w,\alpha,i}(\mathcal{R}_t) \quad (2)$$

where N_m represents the number of models in the ensemble.

The ensemble model deviation serves as a robust indicator of the true prediction error, effectively distinguishing between the reliable interpolation regime and the unreliable extrapolation regime.⁴⁸ In our multi-agent approach, the selection of the appropriate execution agent for predicting material properties is governed by a dynamic threshold σ_{10} . This threshold is carefully chosen rather than arbitrarily determined.⁴⁷ Specifically, σ_{10} is set slightly above the ML Performer's intrinsic fitting error, defined as 1.1 times the training root-mean-square error (RMSE) of the forces. This margin accounts for the inherent noise in the training data while ensuring that the ML model is trusted only within its verified accuracy regime. This guarantees that the system is not overly confident in its predictions for configurations that are less reliable, while still leveraging the high-throughput capabilities of the ML Performer where appropriate.

Algorithm 1 Prediction Tool Selection Algorithm.

Require: Ensemble of Models $\{E_1, E_2, \dots, E_{N_m}\}$, Set of Configurations $\{\mathcal{R}_1, \mathcal{R}_2, \dots, \mathcal{R}_n\}$, Threshold σ_{10}

Ensure: Output Sets $R_{\text{ml}}, R_{\text{dft}}$

```

1: Initialize:  $R_{\text{ml}} \leftarrow \emptyset$   $\triangleright$  Set for configurations handled by
   machine learning models
2: Initialize:  $R_{\text{dft}} \leftarrow \emptyset$   $\triangleright$  Set for configurations handled by
   first-principles methods
3: for each configuration  $\mathcal{R}_i$  in  $\{\mathcal{R}_1, \dots, \mathcal{R}_n\}$  do
4:   for each model  $E_\alpha$  in  $\{E_1, \dots, E_{N_m}\}$  do
5:     Compute  $F_{w,\alpha,i}(\mathcal{R}_i) = -\nabla_i E_w(\mathcal{R}_i)$  for each atom  $i$ 
6:   end for
7:   for each atom  $i$  do
8:      $\langle F_{w,i}(\mathcal{R}_i) \rangle = \frac{1}{N_m} \sum_{\alpha=1}^{N_m} F_{w,\alpha,i}(\mathcal{R}_i)$   $\triangleright$  Calculate the
     expectation
9:      $\sigma_i = \sqrt{\langle \|F_{w,i}(\mathcal{R}_i) - \langle F_{w,i}(\mathcal{R}_i) \rangle\|^2 \rangle}$   $\triangleright$  Calculate the error
10:   end for
11:    $\varepsilon_i \leftarrow \max_i \sigma_i$ 
12:   if  $\varepsilon_i \geq \sigma_{10}$  then
13:     Add  $\mathcal{R}_i$  to  $R_{\text{dft}}$   $\triangleright$  Assign configuration to
     first-principles set if error exceeds the threshold
14:      $R_{\text{dft}} = \{\mathcal{R}_n \mid n \in I_{\text{dft}}, I_{\text{dft}} = \{n \mid \varepsilon_i \geq \sigma_{10}\}\}$ 
15:   else
16:     Add  $\mathcal{R}_i$  to  $R_{\text{ml}}$   $\triangleright$  Otherwise, assign configuration to
     machine learning set
17:      $R_{\text{ml}} = \{\mathcal{R}_n \mid n \in I_{\text{ml}}, I_{\text{ml}} = \{n \mid \varepsilon_i < \sigma_{10}\}\}$ 
18:   end if
19: end for
20: return  $R_{\text{ml}}, R_{\text{dft}}$ 

```

To facilitate this autonomous routing, we design Algorithm 1 to classify molecular configurations into two sets: R_{ml} for the ML Performer and R_{dft} for the DFT Operator. Configurations in R_{ml} are those for which the predicted atomic forces fall within an acceptable error range, $\varepsilon_t < \sigma_{10}$, ensuring that the pre-trained ML models can be confidently utilized by the ML Performer. Conversely, configurations in R_{dft} exhibit larger errors ($\varepsilon_t \geq \sigma_{10}$), indicating the need for the Computing Designer to escalate the task to the DFT Operator for more accurate first-principles calculations.

Therefore, for a new structure, the Computing Designer algorithmically determines whether it belongs to R_{ml} or R_{dft} based on the computed error indicator ε_t . This decentralized decision-making process ensures that the most appropriate and computationally efficient sub-agent is chosen for each material property prediction, retaining system robustness without encumbering the Manager agent's primary reasoning loop.

2.4 From experimental procedures to automated execution of DFT

ChatMat has a specialized agent to achieve automatic execution of DFT. To ensure quick and scalable results for *ab initio* computations, the Manager delegates quantum mechanical evaluations to the autonomous DFT Operator. This agent is equipped with the capability to submit and manage jobs on high-performance computing (HPC) clusters. Exclusively tasked with employing the plane-wave density functional theory (PWDFT) platform⁴⁹ for efficiently calculating complex material properties, the DFT Operator leverages the cluster's vast resources. PWDFT supports multi-accelerator and parallel configurations, providing superior computational efficiency and scalability compared to traditional DFT codes.^{50,51} Its high-performance optimization drastically reduces simulation time, making it well-suited for high-throughput calculations and accelerating material property prediction workflows. Consequently, as shown in Fig. 3(c), PWDFT is employed independently by the DFT Operator within an HPC environment, preventing the Manager from being stalled by heavy computational overhead.

2.4.1 DFT calculations (DFT operator). The foundation of DFT methods lies in solving the Schrödinger equation:

$$\hat{H}\Psi = E\Psi, \quad (3)$$

where \hat{H} is the Hamiltonian operator, Ψ is the wavefunction of the system, and E is the energy eigenvalues. In the context of DFT, the energy of a system is expressed as a functional of the electron density $\rho(\mathbf{r})$:

$$E[\rho] = T[\rho] + \int V_{\text{ext}}(\mathbf{r})\rho(\mathbf{r})\mathrm{d}\mathbf{r} + \frac{1}{2} \iint \frac{\rho(\mathbf{r})\rho(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} \mathrm{d}\mathbf{r}\mathrm{d}\mathbf{r}' + E_{\text{xc}}[\rho], \quad (4)$$

where $T[\rho]$ denotes the kinetic energy functional, $V_{\text{ext}}(\mathbf{r})$ is the external potential, and $E_{\text{xc}}[\rho]$ represents the exchange–correlation energy functional.

The Kohn–Sham equations, central to DFT, are given by



$$\left(-\frac{\hbar^2}{2m}\nabla^2 + V_{\text{eff}}(\mathbf{r})\right)\psi_i(\mathbf{r}) = \varepsilon_i\psi_i(\mathbf{r}), \quad (5)$$

where $\psi_i(\mathbf{r})$ are the Kohn–Sham orbitals, ε_i are the orbital energies, and $V_{\text{eff}}(\mathbf{r})$ is the effective potential,

$$V_{\text{eff}}(\mathbf{r}) = V_{\text{ext}}(\mathbf{r}) + \int \frac{\rho(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} d\mathbf{r}' + \frac{\delta E_{\text{xc}}[\rho]}{\delta \rho(\mathbf{r})}. \quad (6)$$

Here, $\delta E_{\text{xc}}[\rho]/\delta \rho(\mathbf{r})$ denotes the functional derivative of the exchange–correlation energy with respect to the electron density $\rho(\mathbf{r})$.

Additionally, *Ab initio* Molecular Dynamics (AIMD) integrates these calculations into the equations of motion:

$$m_i \frac{d^2 \mathbf{R}_i}{dt^2} = -\nabla_{\mathbf{R}_i} E[\rho], \quad (7)$$

where m_i is the mass of the i -th atom, \mathbf{R}_i is the position vector of the i -th atom, and $\nabla_{\mathbf{R}_i} E[\rho]$ is the gradient of the energy with respect to the position of the i -th atom. The DFT Operator autonomously performs the PWDFD simulation to obtain the total energy E_{total} , forces \mathbf{F}_i , and other relevant properties for a given molecular structure. Upon completion, it returns the results to the Manager *via* the “Evaluate” signal.

2.4.2 Data storage (property depositor). Upon receiving the successful DFT evaluation, the Manager transitions to the “Thought” step to plan the data curation and takes “Action” to invoke the Property Depositor. This agent parses the raw DFT outputs and securely stores the computed properties in the dataset with corresponding molecular identifiers, facilitating quick retrieval for future predictions.

2.4.3 Training ML-PES model (ML performer). The stored data are used to train the DeePMD model.^{52,53} The training process involves optimizing the network parameters θ to minimize the loss function:

$$\mathcal{L}(\theta) = \sum_{i=1}^N \left(E_{\text{total}}^{\text{pred}} - E_{\text{total}}^{\text{true}}\right)^2 + \lambda \|\theta\|^2, \quad (8)$$

where $E_{\text{total}}^{\text{pred}}$ is the predicted total energy, $E_{\text{total}}^{\text{true}}$ is the true total energy from DFT calculations, λ is a regularization parameter, and $\|\theta\|^2$ is the L2 norm of the network parameters to prevent overfitting.

2.4.4 Model utilization (ML performer). Once trained by the ML Performer, the DeePMD model can predict the properties of similar molecules directly. For subsequent queries, the Manager’s reasoning loop can bypass the computationally expensive DFT Operator entirely, delegating tasks directly to the ML Performer to drastically enhance overall system efficiency.

2.5 From experimental procedures to automated execution of ML

2.5.1 ML predictor. Following the decentralized reasoning loop depicted in Fig. 2(a), when the Computing Designer determines that a molecular configuration falls within the reliable interpolation regime ($\varepsilon_t < \sigma_{10}$), the central Manager agent issues a “Thought” to bypass computationally expensive first-principles calculations. It then executes an “Action (Select

Agent)” to route the structural data directly to the ML Performer.

As shown in Fig. 3(b), due to its ability to efficiently handle high-dimensional potential energy surfaces and accurately model complex material behaviors through deep learning techniques, the DeePMD framework is employed autonomously by the ML Performer. ML-PES models provide substantial reductions in computational costs while retaining comparable accuracy to first-principles methods. By training on extensive datasets derived from DFT calculations, these machine learning models can learn complex relationships between molecular structures and their properties.⁴³ A common approach involves representing the potential energy $E_{\text{PES}}(\mathbf{R})$ as a function learned by the ML model:

$$E_{\text{PES}}(\mathbf{R}) \approx \text{ML Model}(\mathbf{R}), \quad (9)$$

where \mathbf{R} denotes the atomic coordinates of the molecule.

Techniques such as neural networks, Gaussian processes, and kernel ridge regression have been employed to develop PES models capable of predicting properties like binding energies, reaction rates, and molecular conformations with impressive speed. For instance, the Neural Network Potential (NNP) can be expressed as

$$E_{\text{NNP}}(\mathbf{R}) = \sum_{i=1}^N \mathcal{N}(\mathbf{G}_i), \quad (10)$$

where $\mathcal{N}(\mathbf{G}_i)$ is the neural network function applied to the symmetry functions \mathbf{G}_i , which represent the local environment of the i -th atom, and N is the total number of atoms in the system.

In this work, we utilize the DeePMD, which employs deep neural networks to accurately represent the PES. It maps the symmetry functions to an energy contribution

$$E_i = \text{Deep neural network}(\mathcal{G}_i), \quad (11)$$

where \mathcal{G}_i represents a different symmetry function for the i -th atom.

Then, the DeePMD model expresses the total energy as a sum of atomic contributions as

$$E_{\text{total}} = \sum_{i=1}^N E_i, \quad (12)$$

where E_i is the energy contribution from the i -th atom and N is the total number of atoms in the system.

In particular, the schematic diagram of the relationship between the tasks from experimental procedures to automated execution of DFT and ML is shown in Fig. 2(b). Upon successful prediction, the ML Performer returns the computed properties to the Manager *via* the “Evaluate” signal, completing the multi-agent asynchronous workflow without encumbering the Manager’s cognitive capacity.

3 Results and discussion

To intuitively demonstrate the professional capabilities of ChatMat, we compared its responses with those of GPT-4o and



Table 1 Sample questions drawn from the task set and the corresponding first-sentence answers produced by GPT-4o and ChatMat

Query	GPT-4o response	ChatMat response
Could you list the details of the atomic forces section for H ₂ ?	The atomic forces section for H ₂ typically refers to the forces acting on each atom in the molecule (incorrect)	The details of the atomic forces section for H ₂ are as follows: force for centroid (x) [a.u.]: -1.963×10^{-5} , force for centroid (y) [a.u.]: 3.83×10^{-6} (correct)
What is the free energy of Al ₂ O ₃ ?	I cannot directly provide the free energy value for Al ₂ O ₃ (incorrect)	The free energy of Al ₂ O ₃ is 29.8450654 eV (correct)
Could you tell me the force along z for Na ₂ SO ₄ ?	I cannot provide a specific value (incorrect)	The force along the z-axis for Na ₂ SO ₄ is 1.62197009 atomic units (a.u.) (correct)
Please construct a machine learning potential energy surface model for Al ₁₇₆ Si ₂₄	I cannot provide a specific result for constructing a machine learning potential energy surface model for Al ₁₇₆ Si ₂₄ (incorrect)	I have completed the construction of the machine learning potential energy surface model for Al ₁₇₆ Si ₂₄ and saved it in the model library (correct)
What does the structure of CaCO ₃ look like?	I cannot provide a specific result for the structure of CaCO ₃ (incorrect)	The structural information of CaCO ₃ has been obtained (correct)
What is the charge density distribution acquisition of BaSO ₄ ?	I cannot provide a specific value for the charge density distribution of BaSO ₄ (incorrect)	The charge density distribution acquisition of BaSO ₄ has been obtained (correct)

presented representative queries in Table 1. This comparison is intended to highlight the necessity of external tool integration for eliminating hallucinations and handling domain-specific calculations that raw LLMs cannot perform.

Subsequently, to systematically assess the automation capabilities, we designed four representative tasks crucial to materials property prediction workflows. The definitions of these tasks are as follows:

3.1 T1: material structure acquisition

This task evaluates the agent's ability to interpret textual descriptions and generate accurate three-dimensional molecular geometries (*e.g.*, converting chemical names or compositions into standard 3D coordinate files).

3.2 T2: charge density distribution acquisition

This task requires the agent to plan and conduct high-throughput *ab initio* calculations to derive spatial charge density distributions without human intervention.

3.3 T3: material ML-PES construction

This task tests the cold start capability. The agent must curate datasets, select hyperparameters, and train a ML-PES model from scratch for complex systems.

3.4 T4: material property database operation

This task assesses the system's ability to manage long-term data. The agent is required to retrieve properties, organize calculation results, and operate/maintain a local Material Property Database.

We analyzed the performance of ChatMat across these tasks, with the results summarized in Fig. 4. For T1, the agent successfully constructed the precise 3D crystal structure of Al₁₇₆Si₂₄. Regarding T2, ChatMat executed the calculation workflow to derive the charge density for (H₂O)₆₄. For the

complex T3, the system successfully completed the training of an ML-PES model for the large-scale (HClO₄)₁₆ supercell. Finally, for T4, ChatMat demonstrated its capability to autonomously build, operate, and maintain a comprehensive Material Property Database.

Each task was evaluated using a carefully constructed benchmark dataset, consisting of 100 natural-language queries per task. These queries were specifically designed to cover a diverse range of materials and properties to mitigate potential selection bias, thereby rigorously assessing ChatMat's autonomy, accuracy, and domain-specific competence. ChatMat integrates both DFT simulation software and ML-PES models, combining data-driven predictions with physics-based simulations. This hybrid architecture enables it to perform complex materials science workflows, including both data-driven predictions and physics-based simulations.

Task outcomes were classified into three categories: True (responses that fully met the task requirements), Formatting False (responses containing errors in grammar, spelling, or formatting that impaired readability), and Factual False (responses with incorrect or logically inconsistent content). This labeling framework allows for a comprehensive and nuanced evaluation of ChatMat's reliability and effectiveness in a variety of scientific scenarios. In this work, 'True' refers to an accurate match within an acceptable approximate range.

To quantitatively assess performance, we adopted four evaluation metrics: success rate, factuality rate, formatting compliance rate, and composite reliability score. These metrics were redefined from standard classification metrics to better reflect the three-way outcome structure (True, Factual False, and Formatting False) of our specific task. They are calculated as follows:

$$\text{Success rate} = \frac{\text{True}}{\text{Total samples}} \quad (13)$$



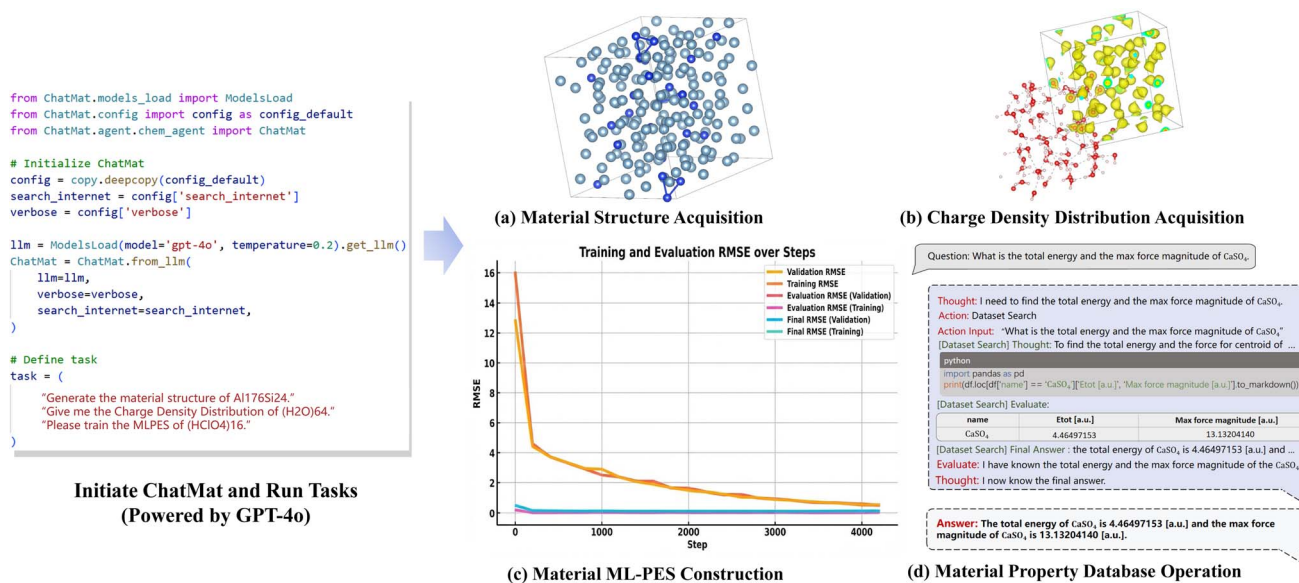


Fig. 4 Representative experimental results across four tasks. The left panel shows the initialization code snippet where the ModelsLoad factory configures GPT-4o as the foundational Large Language Model (LLM) backend for the ChatMat agent. The agent is then equipped with domain-specific tools to execute tasks such as (a) material structure acquisition for $\text{Al}_{176}\text{Si}_{24}$, (b) charge density distribution acquisition for $(\text{H}_2\text{O})_{64}$, (c) material ML-PES construction for $(\text{HClO}_4)_{16}$, and (d) material property database operation. ML-PES: machine learning-driven potential energy surface; RMSE: root mean squared error.

$$\text{Factuality rate} = \frac{\text{True}}{\text{True} + \text{Factual False}} \quad (14)$$

$$\text{Formatting compliance rate} = \frac{\text{True}}{\text{True} + \text{Formatting False}} \quad (15)$$

Composite reliability score =

$$2 \times \frac{\text{Formatting compliance rate} \times \text{Factuality rate}}{\text{Formatting compliance rate} + \text{Factuality rate}} \quad (16)$$

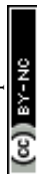
These metrics were selected for their ability to capture complementary aspects of performance. Success rate provides a general measure of overall correctness. The factuality rate reflects the system's ability to avoid factual errors, which is critical in scientific and engineering applications where misinformation can lead to flawed conclusions. The formatting compliance rate captures linguistic and presentation quality by penalizing formatting issues that may hinder human interpretation. Finally, the composite reliability score, calculated as the harmonic mean of the factuality and formatting compliance rates, offers a conservative assessment of performance. Since the harmonic mean is mathematically biased toward the lower value, this metric ensures that the system is heavily penalized if it fails in either content validity or output specifications, thereby enforcing a rigorous standard for scientific reliability.

We conducted each task three times to minimize potential random errors. The total statistical results of the four tasks across the three runs are presented in Fig. 5. This figure provides a macroscopic view of the error distribution, categorizing outcomes into True, Factual False, and Formatting False

to intuitively illustrate the system's overall failure modes. Complementing this, Fig. 6 offers a microscopic view by quantifying the specific performance metrics for each independent run side-by-side. This detailed breakdown demonstrates the system's stability and consistency across repeated experiments, capturing variations that are obscured in the aggregated distribution.

Across all tasks, ChatMat consistently demonstrated high performance, with the success rate ranging from 91% to 96%. In particular, tasks such as Material Property Database Operation and Material Structure Acquisition yielded the highest average success rates of 95% and 94%, respectively, highlighting ChatMat's strong capabilities in retrieving material properties and processing structural information. For more computationally demanding tasks like ML-PES Construction and Charge Density Distribution Acquisition, the system retained robust performance with success rates of approximately 92% and 93%.

Further analysis of factuality rate, with all task scores exceeding 95.8%, indicates that ChatMat effectively minimizes factual inaccuracies, underscoring its high scientific reliability. Similarly, the consistently high Formatting Compliance Rate scores reflect ChatMat's strong adherence to linguistic and formatting standards, which is an essential requirement for seamless integration into human-in-the-loop workflows. The resulting Composite Reliability Scores, all above 95.3%, confirm ChatMat's balanced performance in both producing scientifically valid content and presenting it in a clear, interpretable manner. These results collectively validate ChatMat as a reliable and effective intelligent agent for complex, domain-specific computational workflows.



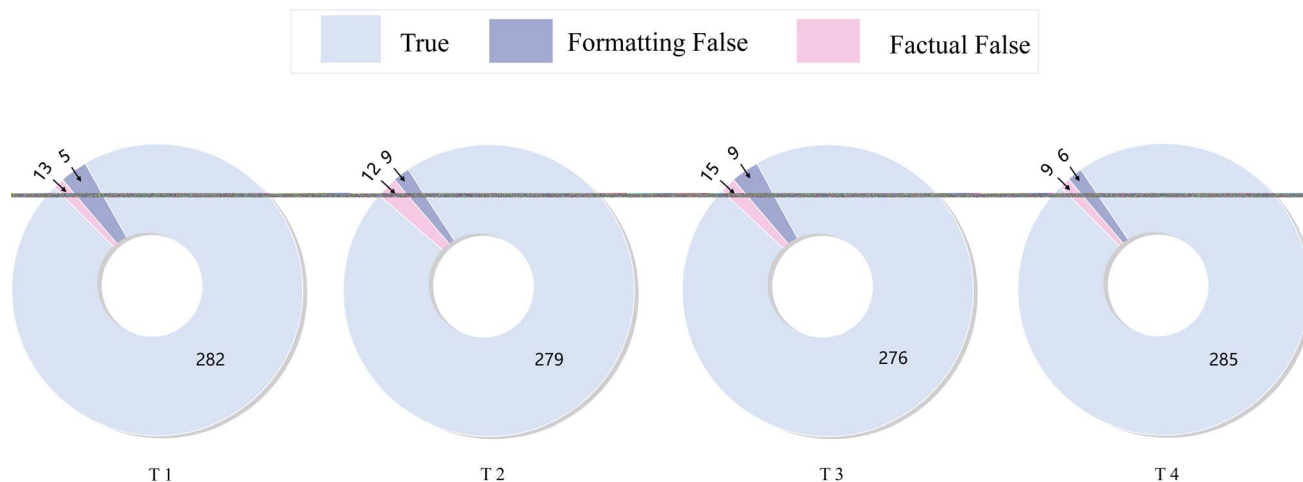


Fig. 5 Total statistical results of the four tasks after three runs, showing the distribution of errors categorized as True, Formatting False, and Factual False.

Traditional methods often struggle to translate the complex structural and chemical characteristics of materials into data that can be effectively processed. ChatMat tackles this challenge by leveraging carefully designed prompts to interpret and convert complex material descriptions into usable data formats. By incorporating these specialized prompts, ChatMat ensures that both textual input and computational models are seamlessly integrated, facilitating the accurate generation of material structures and the execution of physics-based simulations. This

was demonstrated in Material Structure Acquisition (T1) and Charge Density Distribution Acquisition (T2), where ChatMat autonomously generated accurate 3D molecular structures, effectively bridging the gap between textual input and computational predictions.

ChatMat mitigates the challenge of insufficient high-quality, domain-specific datasets in the chemical field by integrating ML-PES and DFT methods. This hybrid approach enables it to generate accurate predictions even when faced with sparse

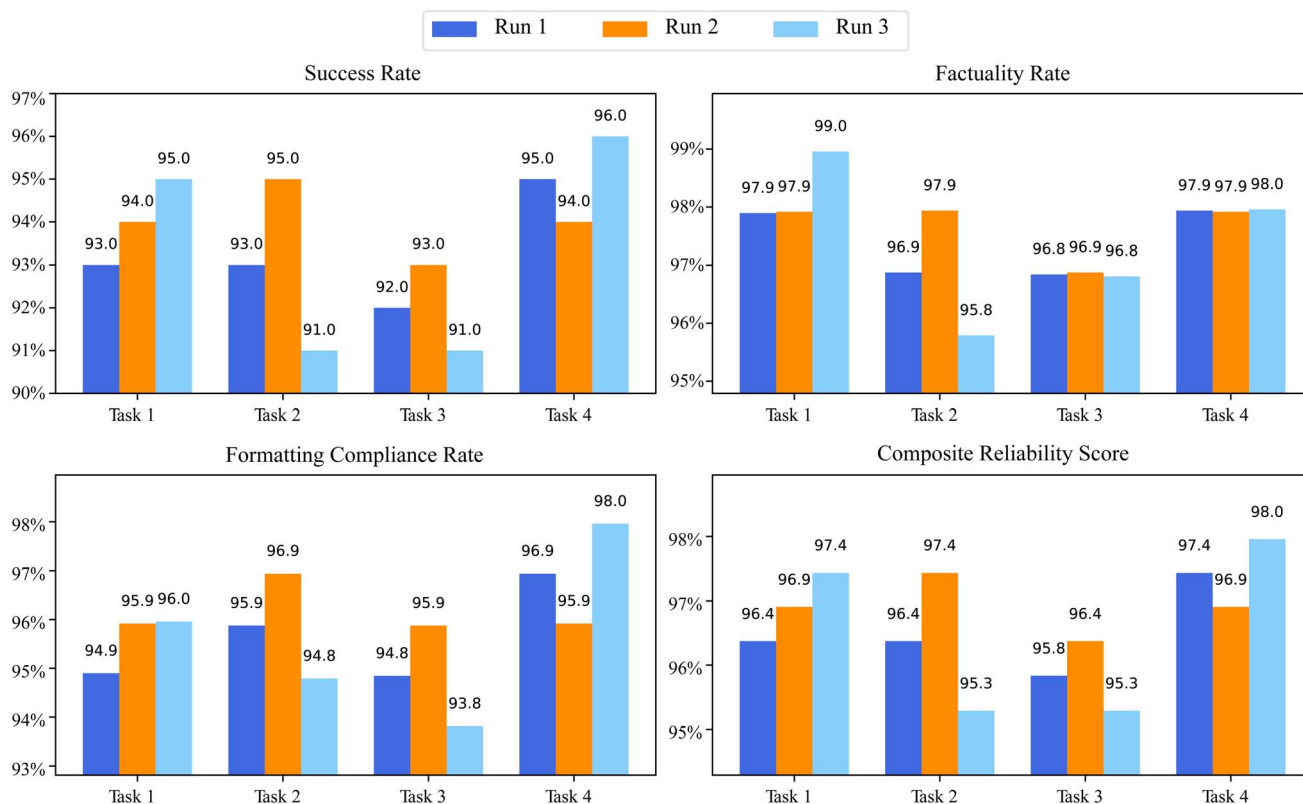


Fig. 6 Performance of GPT-4o-based ChatMat on four tasks across three runs, evaluated using the success rate, factuality rate, formatting compliance rate, and composite reliability score. Each plot compares the performance of the three runs for each metric across the four tasks.



datasets, specifically in the cold start scenario where initial domain-specific training data are non-existent or minimal. The Material ML-PES Construction task (T3) demonstrates ChatMat's ability to autonomously manage datasets and select optimal hyperparameters for model training, overcoming the limitations of sparse, high-quality data. Moreover, the integration of DFT simulations allows for the continuous generation of high-quality data. ChatMat's operation of Material Property Database Operation (T4) highlights its capacity to autonomously build, maintain, and refine material property databases, thereby expanding its knowledge base and improving its predictive accuracy over time.

A longstanding issue in computational materials science has been the heavy reliance on manual operations, resulting in workflows that lag behind the automation seen in other scientific disciplines. This situation often requires researchers to invest a lot of time and effort in performing repetitive and tedious tasks when dealing with complex calculations and experimental processes, greatly limiting the efficiency of material exploration. ChatMat is capable of autonomously executing end-to-end workflows, representing a transformative shift toward more efficient, automated systems. As illustrated in Fig. 7, ChatMat autonomously executed complete end-to-end workflows, integrating database retrieval, ML-based property prediction, and DFT-level simulations to compute and compare total energies and centroid forces. The system's ability to autonomously coordinate these tasks exemplifies its robustness and flexibility in handling complex, real-world materials problems. This case underscores ChatMat's potential to enhance autonomous scientific exploration and greatly accelerate research in computational materials science. ChatMat's ability to perform these tasks with minimal human oversight represents a transformative shift toward more efficient, automated workflows that could accelerate the pace of materials exploration.

Another longstanding challenge in computational materials science is the difficulty in incorporating true physical laws into machine learning models, often resulting in predictions that are accurate in some contexts but fail to generalize across complex or highly variable systems. By integrating DFT simulations, which solve the Schrödinger equation and other fundamental physical models, ChatMat ensures that predictions are derived from established physics while retaining the computational efficiency of ML methods. This hybrid approach enables ChatMat to strike a critical balance between computational efficiency and accuracy, addressing the common trade-off between these two factors in current agent-based systems for materials modeling.

An ablation study further demonstrates ChatMat's advantages by comparing it to traditional methods based solely on either ML-PES or DFT simulations using a benchmark set of 100 randomly selected material property queries. As summarized in Table 2, the ML-only agent exhibits a limited success rate of 48%, constrained by the quality and representativeness of their training data. The DFT-only agent performed more accurately, achieving an 88% success rate, but suffered from high computational costs and limited scalability. Specifically, in unattended

Question: How does the total energy and the max force magnitude of $(\text{H}_2\text{O})_8$ compare with $(\text{NH}_3)_4$.

Thought: I need to find the total energy and the max force magnitude of $(\text{H}_2\text{O})_8$ and $(\text{NH}_3)_4$.

Action: Dataset search

Action Input: "What is the total energy and the max force magnitude of $(\text{H}_2\text{O})_8$ "

[Property Depositor] Thought: To find the total energy and the force for centroid of ...

```
python
import pandas as pd
print(df.loc[df['name'] == '(H2O)8']['Etot [a.u.]', 'Max force magnitude [a.u.'].to_markdown())
print(df.loc[df['name'] == '(NH3)4']['Etot [a.u.]', 'Max force magnitude [a.u.'].to_markdown())
```

[Property Depositor] Evaluate:

name	Etot [a.u.]	Max force magnitude [a.u.]
$(\text{H}_2\text{O})_8$	-19.170570	42.1522951

[Dataset Search] Final Answer: the total energy of $(\text{H}_2\text{O})_8$ is -19.170570 [a.u.] and the ...

Evaluate: I have known the ... of the $(\text{H}_2\text{O})_8$ but I do not know any information of $(\text{NH}_3)_4$.

Thought: I need to find the total energy and the max force magnitude of $(\text{NH}_3)_4$.

Action: Computing Designer

Action Input: I need to determine if there is a suitable model that can calculate...

[Computing Designer] Thought: I need to get the structure information of $(\text{NH}_3)_4$ first.

Final Answer: The structure information of $(\text{NH}_3)_4$ is saved in POSCAR.

[Computing Designer] Final Answer: The total energy of $(\text{NH}_3)_4$ can be calculated by ...

Evaluate: The total energy of $(\text{NH}_3)_4$ can be calculated by the model ...

Thought: I need to calculate the total of $(\text{NH}_3)_4$.

Action: ML Performer

[ML Performer] Thought: I need to calculate the total energy of $(\text{NH}_3)_4$ by pre-trained ...

Final Thought: The total energy of $(\text{NH}_3)_4$ is -13.324207 [a.u.].

Action: DFT Predictor

[DFT Predictor] Thought: I need to calculate the max force magnitude of $(\text{NH}_3)_4$.

Final Thought: Based on the result, the max force magnitude of $(\text{NH}_3)_4$ is 2.94837607 [a.u.].

Final Thought: I now know the final answer.

Answer: The total energy for $(\text{H}_2\text{O})_8$ and $(\text{NH}_3)_4$ are -19.170570 [a.u.] and -13.324207 [a.u.]. Regarding the maximum force magnitude, $(\text{H}_2\text{O})_8$ exhibits a significantly higher value (42.1522951 [a.u.]) compared to that of $(\text{NH}_3)_4$ (2.94837607 [a.u.]).

Fig. 7 Example of a predictor for the question "How do the total energy and the force for the centroid of $(\text{H}_2\text{O})_8$ compare with those of $(\text{NH}_3)_4$?". ChatMat automatically completed this task. a.u.: atomic units.

batch processing, 3% of calculations failed due to SCF non-convergence, 7% due to timeouts, and 2% due to I/O errors. In contrast, ChatMat achieved a superior success rate of 95% by integrating both ML and DFT techniques within a unified LLM-driven framework. This integration enables it to strike a critical balance between computational efficiency and quantum-level accuracy, effectively overcoming a central trade-off in current agent-based systems for materials modeling.

To quantitatively substantiate these efficiency improvements, we evaluated the computational cost, measured using the number of DFT calls, across the aforementioned benchmark of 100 diverse queries. In the DFT-only agent, every property evaluation necessitates a full first-principles calculation. Accounting for the system's error-recovery mechanism, which allows a maximum of three retries per task, the DFT-only agent required an average of 1.15 DFT calls per query. By contrast, ChatMat leverages the Computing Designer agent to dynamically route tasks to the ML Performer when the predicted error falls within the confidence threshold. Through this intelligent routing, ChatMat reduced the average number of DFT calls to 0.35 per query, representing a computational cost saving of approximately 69.6%. Even in the worst-case scenario, such as

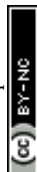


Table 2 Performance comparison of different agent configurations

Agent	Success rate	Formatting compliance rate	Factuality rate	Composite reliability score
ML-only	48%	46%	49%	42%
DFT-only	88%	80%	83%	86%
ChatMat	95%	91%	93%	94%

encountering highly out-of-distribution structures where the ML model completely falls back to the DFT Operator and triggers maximum error retries, ChatMat is capped at exactly 3 DFT calls. This ensures a rigorous lower bound on accuracy without ever exceeding the DFT-only agent's maximum computational cost. This closed-loop quantification indicates that ChatMat enhances the overall success rate while reducing average computational overhead. By intelligently routing tasks to the ML-PES Performer when predictions fall within the defined confidence threshold, the system reduces the average number of DFT calls per query from 1.15 to 0.35. This represents a 69.6% reduction in computational cost, establishing a measurable improvement in efficiency while retaining the established accuracy bounds of quantum mechanical simulations.

4 Conclusion

This study presents our latest advancements in autonomous material prediction and exploration. ChatMat, an intelligent multi-agent chemist system, incorporates a variety of specialized LLM-based agents, each capable of executing complex research tasks both independently and collaboratively. It features a Manager agent that interfaces with human researchers and oversees four role-specific agents: Property Depositor, Computing Designer, ML Performer, and DFT Operator. This architecture amplifies the effectiveness of individual agents through collaboration and coordination, enabling them to jointly address intricate tasks and achieve clearly defined objectives.

By seamlessly integrating diverse computational paradigms, including machine learning, first-principles simulations, and LLM-driven orchestration, ChatMat delivers high-accuracy predictions while retaining computational efficiency. This hybrid architecture addresses key limitations of conventional methods and positions ChatMat as a powerful, efficient, and accessible tool for materials modeling. By supporting both domain experts and non-specialists, ChatMat facilitates collaborative and autonomous scientific exploration, thereby paving the way for accelerated innovation in computational materials science.

Author contributions

S. L. was responsible for conceptualization, methodology, software, investigation, validation, formal analysis, visualization, data curation, project administration, and writing – original draft. L. P., S. J., and W. W. contributed to software, investigation, and writing – review & editing. W. H. was responsible for

conceptualization, methodology, resources, supervision, funding acquisition, and writing – review & editing. All authors contributed to the article and approved the submitted version.

Conflicts of interest

There are no conflicts to declare.

Data availability

The datasets for this work, including data for training and experimental results, are available at <https://doi.org/10.5281/zenodo.18459314>. The ChatMat code implementation can be found at <https://github.com/Xiaobu-USTC/ChatMat.git>.

Supplementary information (SI): the framework outputs and the main agent setup. See DOI: <https://doi.org/10.1039/d5dd00582e>.

Acknowledgements

This study was partly supported by the Anhui Province Science and Technology Innovation Project (Grant No. 202423k09020010), the Strategic Priority Research Program of the Chinese Academy of Sciences (Grant No. XDB0450101), the Innovation Program for Quantum Science and Technology (No. 2021ZD0303306), the National Natural Science Foundation of China (21688102, 22173093, and 22373096), the Anhui Provincial Key Research and Development Program (2022a05020052), and the National Key Research and Development Program of China (2021YFB0300600). This work was partially supported by the Dream Set Off-Kunpeng & Ascend Seed Program.

Notes and references

- 1 D. Jacob, C. Ming-Wei, L. Kenton and T. Kristina, *arXiv*, 2019, e-prints, arXiv:1810.04805, DOI: [10.48550/arXiv.1810.04805](https://doi.org/10.48550/arXiv.1810.04805).
- 2 OpenAI, *arXiv*, 2023, e-prints, arXiv:2303.08774, DOI: [10.48550/arXiv.2303.08774](https://doi.org/10.48550/arXiv.2303.08774).
- 3 A. Chowdhery, S. Narang, J. Devlin, *et al.*, *J. Mach. Learn. Res.*, 2023, **24**, 1–113.
- 4 T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever and D. Amodei, *arXiv*, 2020, e-prints, arXiv:2005.14165, DOI: [10.48550/arXiv.2005.14165](https://doi.org/10.48550/arXiv.2005.14165).



- 5 A. Matarazzo and R. Torlone, *arXiv*, 2025, e-prints, arXiv:2501.04040, DOI: [10.48550/arXiv.2501.04040](https://doi.org/10.48550/arXiv.2501.04040).
- 6 J. Kaddour, J. Harris, M. Mozes, H. Bradley, R. Raileanu and R. McHardy, *arXiv*, 2023, e-prints, arXiv:2307.10169, DOI: [10.48550/arXiv.2307.10169](https://doi.org/10.48550/arXiv.2307.10169).
- 7 Z. Zheng, O. Zhang, C. Borgs, J. T. Chayes and O. M. Yaghi, *J. Chem. Phys.*, 2023, **145**, 18048–18062.
- 8 Z. Zheng, A. H. Alawadhi, S. Chheda, S. E. Neumann, N. Rampal, S. Liu, H. L. Nguyen, Y.-H. Lin, Z. Rong, J. I. Siepmann, *et al.*, *J. Chem. Phys.*, 2023, **145**, 28284–28295.
- 9 K. M. Jablonka, P. Schwaller, A. Ortega-Guerrero and B. Smit, *Nat. Mach. Intell.*, 2024, **6**, 161–169.
- 10 Y. Kang and J. Kim, *Nat. Commun.*, 2024, **15**, 4705.
- 11 M. P. Polak and D. Morgan, *Nat. Commun.*, 2024, **15**, 1569.
- 12 G. M. Hocky, *Nat. Mach. Intell.*, 2024, **6**, 249–250.
- 13 T. Schick, J. Dwivedi-Yu, R. Dessì, *et al.*, *Adv. Neural Inf. Process. Syst.*, 2024, **36**, 68539–68551.
- 14 C. M. Castro Nascimento and A. S. Pimentel, *J. Chem. Inf. Model.*, 2023, **63**, 1649–1655.
- 15 P. Lee, S. Bubeck and J. Petro, *N. Engl. J. Med.*, 2023, **388**, 1233–1239.
- 16 E. Waisberg, J. Ong, M. Masalkhi, S. A. Kamran, N. Zaman, P. Sarker, A. G. Lee and A. Tavakkoli, *Ir. J. Med. Sci.*, 2023, **192**, 3197–3200.
- 17 H. Nori, N. King, S. M. McKinney, D. Carignan and E. Horvitz, *arXiv*, 2023, e-prints, arXiv:2303.13375, DOI: [10.48550/arXiv.2303.13375](https://doi.org/10.48550/arXiv.2303.13375).
- 18 L. Reynolds and K. McDonell, Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm, Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems (CHI EA '21), 2021, p. 314, DOI: [10.1145/3411763.3451760](https://doi.org/10.1145/3411763.3451760).
- 19 Z. Zhong, K. Zhou and D. Mottin, *arXiv*, 2024, e-prints, arXiv:2403.05075, DOI: [10.48550/arXiv.2403.05075](https://doi.org/10.48550/arXiv.2403.05075).
- 20 T. Hu, H. Song, T. Jiang and S. Li, *Symmetry*, 2020, **12**, 1889.
- 21 L. Ward, A. Agrawal, A. Choudhary and C. Wolverton, *npj Comput. Mater.*, 2016, **2**, 1–7.
- 22 J. Dagdelen, A. Dunn, S. Lee, N. Walker, A. S. Rosen, G. Ceder, K. A. Persson and A. Jain, *Nat. Commun.*, 2024, **15**, 1418.
- 23 G. Tom, S. P. Schmid, S. G. Baird, Y. Cao, K. Darvish, H. Hao, S. Lo, S. Pablo-García, E. M. Rajaonson, M. Skreta, *et al.*, *Chem. Rev.*, 2024, **124**, 9633–9732.
- 24 U. Anwar, A. Saparov, J. Rando, D. Paleka, M. Turpin, P. Hase, E. Singh Lubana, E. Jenner, S. Casper, O. Sourbut, B. L. Edelman, Z. Zhang, M. Günther, A. Korinek, J. Hernandez-Orallo, L. Hammond, E. Bigelow, A. Pan, L. Langosco, T. Korbak, H. Zhang, R. Zhong, S. Ó. hÉigeartaigh, G. Recchia, G. Corsi, A. Chan, M. Anderljung, L. Edwards, A. Petrov, C. Schroeder de Witt, S. R. Motwan, Y. Bengio, D. Chen, P. H. S. Torr, S. Albanie, T. Maharaj, J. Foerster, F. Tramer, H. He, A. Kasirzadeh, Y. Choi and D. Krueger, *arXiv*, 2024, e-prints, arXiv:2404.09932, DOI: [10.48550/arXiv.2404.09932](https://doi.org/10.48550/arXiv.2404.09932).
- 25 M.-H. Van, P. Verma, C. Zhao and X. Wu, *arXiv*, 2025, preprint, arXiv:2506.20743, DOI: [10.48550/arXiv.2506.20743](https://doi.org/10.48550/arXiv.2506.20743).
- 26 E. Barbierato and A. Gatti, *Electronics*, 2024, **13**, 416.
- 27 Y. Shen, K. Song, X. Tan, D. Li, W. Lu and Y. Zhuang, *Adv. Neural Inf. Process. Syst.*, 2024, **36**, 38154–38180.
- 28 L. Reynolds and K. McDonell, *arXiv*, 2021, e-prints, arXiv:2102.07350, DOI: [10.48550/arXiv.2102.07350](https://doi.org/10.48550/arXiv.2102.07350).
- 29 Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu, A. H. Awadallah, R. W. White, D. Burger and C. Wang, *arXiv*, 2023, e-prints, arXiv:2308.08155, DOI: [10.48550/arXiv.2308.08155](https://doi.org/10.48550/arXiv.2308.08155).
- 30 J. Liu, X. Huang, Z. Chen and Y. Fang, *CCF International Conference on Natural Language Processing and Chinese Computing*, 2024, pp. 255–267.
- 31 H. Liu, H. Yin, Z. Luo and X. Wang, *Synth. Syst. Biotechnol.*, 2025, **10**, 23–38.
- 32 Z. Yang, L. Li, J. Wang, K. Lin, E. Azarnasab, F. Ahmed, Z. Liu, C. Liu, M. Zeng and L. Wang, *arXiv*, 2023, preprint, arXiv:2303.11381, DOI: [10.48550/arXiv.2303.11381](https://doi.org/10.48550/arXiv.2303.11381).
- 33 S. Lv, L. Peng, W. Wu, Y. Yao, S. Jiao and W. Hu, *Mater. Genome Eng. Adv.*, 2025, **3**, e70013.
- 34 J. Li, W. Liu, Z. Ding, W. Fan, Y. Li and Q. Li, *arXiv*, 2024, e-prints, arXiv:2403.04197, DOI: [10.48550/arXiv.2403.04197](https://doi.org/10.48550/arXiv.2403.04197).
- 35 N. Yoshikawa, M. Skreta, K. Darvish, S. Arellano-Rubach, Z. Ji, L. Bjørn Kristensen, A. Z. Li, Y. Zhao, H. Xu, A. Kuramshin, *et al.*, *Auton. Rob.*, 2023, **47**, 1057–1086.
- 36 K. Chen, J. Lu, J. Li, X. Yang, Y. Du, K. Wang, Q. Shi, J. Yu, L. Li, J. Qiu, J. Pan, Y. Huang, Q. Fang, P. A. Heng and G. Chen, *arXiv*, 2023, e-prints, arXiv:2311.10776, DOI: [10.48550/arXiv.2311.10776](https://doi.org/10.48550/arXiv.2311.10776).
- 37 T. Song, M. Luo, X. Zhang, L. Chen, Y. Huang, J. Cao, Q. Zhu, D. Liu, B. Zhang, G. Zou, *et al.*, *J. Chem. Phys.*, 2025, **147**, 12534–12545.
- 38 D. A. Boiko, R. MacKnight, B. Kline and G. Gomes, *Nature*, 2023, **624**, 570–578.
- 39 A. M. Bran, S. Cox, O. Schilter, C. Baldassari, A. D. White and P. Schwaller, *Nat. Mach. Intell.*, 2024, **6**, 525–535.
- 40 P. Giannozzi, O. Baseggio, P. Bonfà, D. Brunato, R. Car, I. Carnimeo, C. Cavazzoni, S. De Gironcoli, P. Delugas, F. Ferrari Ruffino, *et al.*, *J. Chem. Phys.*, 2020, **152**, 154105.
- 41 T. D. Kühne, M. Iannuzzi, M. Del Ben, V. V. Rybkin, P. Seewald, F. Stein, T. Laino, R. Z. Khaliullin, O. Schütt, F. Schiffmann, *et al.*, *J. Chem. Phys.*, 2020, **152**, 194103.
- 42 G. R. Schleder, A. C. Padilha, C. M. Acosta, M. Costa and A. Fazzio, *J. Phys.: Mater.*, 2019, **2**, 032001.
- 43 D. Lanzoni, F. Rovaris and F. Montalenti, *Sci. Rep.*, 2022, **12**, 3760.
- 44 A. Marcato, D. Marchisio and G. Boccardo, *Can. J. Chem. Eng.*, 2023, **101**, 3013–3018.
- 45 S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan and Y. Cao, *arXiv*, 2022, e-prints, arXiv:2210.03629, DOI: [10.48550/arXiv.2210.03629](https://doi.org/10.48550/arXiv.2210.03629).
- 46 M. K. Horton, P. Huck, R. X. Yang, J. M. Munro, S. Dwaraknath, A. M. Ganose, R. S. Kingsbury, M. Wen, J. X. Shen, T. S. Mathis, *et al.*, *Nat. Mater.*, 2025, **24**, 1–11.
- 47 Y. Zhang, H. Wang, W. Chen, J. Zeng, L. Zhang, H. Wang and E. Weinan, *Comput. Phys. Commun.*, 2020, **253**, 107206.
- 48 L. Zhang, D.-Y. Lin, H. Wang, R. Car and W. E. Phys. Rev. Mater., 2019, **3**, 023804.



- 49 W. Hu, L. Lin, A. S. Banerjee, E. Vecharynski and C. Yang, *J. Chem. Theory Comput.*, 2017, **13**, 1188–1198.
- 50 W. Hu, L. Lin and C. Yang, *J. Chem. Theory Comput.*, 2017, **13**, 5458–5467.
- 51 W. Hu, X. Qin, Q. Jiang, J. Chen, H. An, W. Jia, F. Li, X. Liu, D. Chen, F. Liu, y. Zhao and J. Yang, *Sci. Bull.*, 2021, **66**, 111–119.
- 52 H. Wang, L. Zhang, J. Han and W. E, *Comput. Phys. Commun.*, 2018, **228**, 178–184.
- 53 J. Zeng, D. Zhang, D. Lu, P. Mo, Z. Li, Y. Chen, M. Rynik, L. Huang, Z. Li, S. Shi, Y. Wang, H. Ye, P. Tuo, J. Yang, Y. Ding, Y. Li, D. Tisi, Q. Zeng, H. Bao, Y. Xia, J. Huang, K. Muraoka, Y. Wang, J. Chang, F. Yuan, S. L. Bore, C. Cai, Y. Lin, B. Wang, J. Xu, J.-X. Zhu, C. Luo, Y. Zhang, R. E. A. Goodall, W. Liang, A. K. Singh, S. Yao, J. Zhang, R. Wentzcovitch, J. Han, J. Liu, W. Jia, D. M. York, W. E, R. Car, L. Zhang and H. Wang, *J. Chem. Phys.*, 2023, **159**, 054801.

