TRITONDFT: AN AGENTIC FRAMEWORK AND BENCHMARK FOR AUTOMATED DFT WORKFLOWS

Anonymous authors
Paper under double-blind review

ABSTRACT

Density Functional Theory (DFT) is a cornerstone of materials science, yet executing DFT in practice requires coordinating a complex, multi-step workflow. Existing tools and LLM-based solutions automate parts of the workflow, but lack support for full workflow automation, diverse task adaptation, and accuracy–cost trade-off optimization in DFT parameter choices. To this end, we present TRITONDFT, an agentic framework adopting an expert-curated, extensible workflow design, with Pareto-aware parameter inference and automated parallelization, to enable efficient and accurate DFT execution. We further introduce DFTBENCH, a benchmark for evaluating the agent's multi-dimensional capabilities, spanning accuracy-cost trade-off optimization, high-performance computing knowledge, and monetary cost efficiency. Our code and benchmark suite can be found at `https://anonymous.4open.science/r/TritonDFT-43C7/`.

1 INTRODUCTION

Density Functional Theory (DFT) (Hohenberg & Kohn, 1964; Kohn & Sham, 1965) stands as the computational cornerstone of modern materials science. As a first-principles method, DFT provides high-fidelity predictions to validate theoretical hypotheses and reduce experimental cost.

Executing DFT in practice involves a complex, multi-step workflow. Practitioners must search for structural information, configure input parameters, write DFT software-specific scripts, launch and monitor HPC jobs, and interpret and analyze execution results. As shown in Figure 1, such steps require distinct areas of expertise, including physics and materials science, DFT library software details, and High Performance Computing (HPC). According to our survey of domain researchers, each step can take minutes to hours of manual effort. This overhead slows down the discovery process. While existing DFT tools can handle certain low-level details, such as script generation (Mathew et al., 2017; Larsen et al., 2017) and HPC resource management (Pizzi et al., 2016), users still need to manually handle most of the steps and coordinate the overall workflow.

Such manual overhead gives rise to a natural question: can we leverage Large Language Model (LLM) agents to orchestrate these steps and enable automation? LLM agents have been successfully applied across materials science, including domain-specific tool augmentation (M. Bran et al., 2024; Zhang et al., 2024), theoretical-hypothesis loops (Ding et al., 2024; Kumbhar et al., 2025), and autonomous laboratories (Szymanski et al., 2023; Dai et al., 2025). However, applying LLM-based agents to DFT remains challenging, because DFT workflows are complex, demand high fidelity, and involve cross-domain expertise. These challenges directly motivate our design of TRITONDFT.

First, the complexity of DFT increases the difficulty of implementing a robust and generalizable agent framework. Modern DFT packages such as `Quantum ESPRESSO` (Giannozzi et al., 2009) distribute their functionality across specialized executables and expose dozens of interdependent parameters, with different invocation patterns and analysis methods across tasks. While prior works have demonstrated feasibility on specific tasks such as structural relaxation or adsorption (Wang et al., 2025; Liu et al., 2025), they typically rely on task-specific workflows. In contrast, TRITONDFT adopts an expert-informed Plan–Execute–Refine workflow design, coupled with an explicit task–to–executable mapping mechanism for extensibility. TRITONDFT now supports a broad set of tasks, ranging from structural optimization to complex tasks such as phonon properties.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

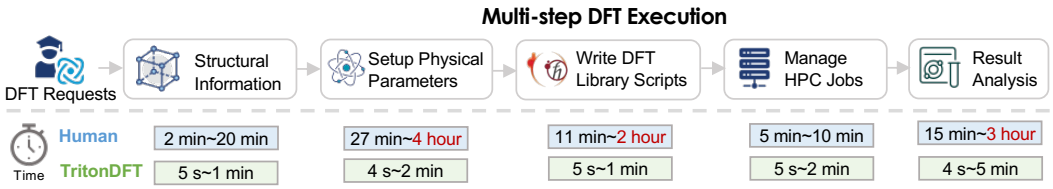

Figure 1: DFT execution is a complex, multi-step process requiring heterogeneous domain expertise. Based on an internal survey of 24 domain researchers at the PhD level or above (see Appendix A.1 for the survey procedure), manually handling each step typically takes minutes to hours. TRITONDFT reduces the per-step time to the scale of seconds to minutes, and provides automation across the entire workflow.

Second, as a numerical simulation, DFT requires results to be obtained both accurately and efficiently. Thus, DFT parameter configurations simultaneously affect numerical fidelity and computational workload, introducing an inherent accuracy–cost trade-off. An existing agent study on parameter configuration (Xia et al., 2025) primarily focuses on accuracy while leaving this trade-off problem unsolved. TRITONDFT introduces a Pareto-aware parameter inference method, enabling the LLM to estimate the accuracy–cost Pareto frontier and iteratively refine configurations.

Third, despite extensive benchmarks like graduate-level materials-domain knowledge (Zaki et al., 2024; Mirza et al., 2024), key capabilities in end-to-end DFT workflows, involving finding Pareto-optimality, HPC parallelization, and cost efficiency, remain unevaluated. We present DFTBENCH to evaluate these capabilities. DFTBENCH comprises 67 materials spanning 10 distinct types to ensure diversity in physical properties and computational complexity, with expert-curated convergence tests to obtain Pareto-optimal parameter configurations and ground-truth results.

In summary, our contributions can be summarized as follows:

- We present TRITONDFT, an expert-informed agentic framework for automating complete DFT workflow execution and accelerating theoretical simulation setup.
- We present DFTBENCH, a benchmark suite for evaluating agentic capabilities in automated DFT, spanning accuracy-cost trade-offs, HPC parallelization, and monetary costs.
- Comprehensive experiments demonstrate the potential of TRITONDFT as a practical building block for theoretical simulation research. We also reveal substantial capability differences of LLMs across different task types in the automation.

2 BACKGROUND AND RELATED WORK

2.1 DFT EXECUTION AND CHALLENGES

DFT predicts material properties from atomic structures through a sequence of calculations, such as structural relaxation and self-consistent field (SCF) calculations. Each calculation requires two types of parameters. Physical parameters $\boldsymbol{\theta}_{\text{phy}}$, such as energy cutoffs, k-point grids, and spin-orbit coupling, control accuracy and convergence, while HPC parameters $\boldsymbol{\theta}_{\text{hpc}}$, such as parallelization over k-points or images, determine how the calculation maps to hardware. Efficient DFT execution therefore requires expertise in both materials science and high-performance computing.

Parameters with Accuracy–Cost Trade-off. DFT execution can take from minutes to hours or even days, depending strongly on the parameter configuration $\boldsymbol{\theta}$. As illustrated in Figure 2, increasing physical parameters such as the k-point density and energy cutoff can reduce calculation error but also increase computational cost. Practitioners must therefore identify *Pareto-optimal* configurations that meet the required accuracy at low cost. This becomes more difficult when multiple parameters interact, requiring convergence tests or prior experience.

DFT Interfaces and Automation Tools. DFT libraries such as `Quantum ESPRESSO` (Giannozzi et al., 2009) and `VASP` (Hafner, 2008) require users to prepare inputs, submit jobs, and inspect outputs manually. Existing tools automate parts of this process: Atomate (Mathew et al., 2017) and

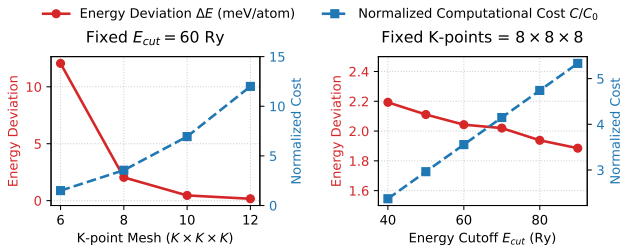

Figure 2: Energy deviation and computational cost variations with different DFT parameters for Silicon (Space Group $\mathrm{Fd\bar{3}m}$). Normalized cost is calculated as C/C_0, where C denotes estimated computational cost and C_0 corresponds to the cost at the minimum k-grid (left panel) or minimum energy cutoff (right panel). Both panels reveal a trade-off between accuracy and cost.

ASE (Larsen et al., 2017) assist with script generation, while AiiDA (Pizzi et al., 2016) manages job execution and provides interfaces for a small set of $\boldsymbol{\theta}_{\text{phy}}$. However, advanced physical settings such as magnetic order still require user expertise. Moreover, these protocols are fixed, whereas TRITONDFT leverages the reasoning ability of LLMs to adapt configurations to the user demand.

2.2 LLM AGENTS FOR AUTOMATED MATERIAL DISCOVERY

LLM agents have been used for scientific workflows through tool use and multi-step coordination (Wang et al., 2023; Yao et al., 2024; Hong et al., 2023). In materials science, prior agents use tools for paper extraction (Cheung et al., 2024; Hira et al., 2024; Song et al., 2023), retrieval (Schilling-Wilhelmi et al., 2025; Chiang et al., 2024; McNaughton et al., 2024), surrogate models (Liu et al., 2024), and tool hubs (Zhang et al., 2024; M. Bran et al., 2024). These tools usually expose simple input-output interfaces. DFT does not fit this single-call pattern: a single calculation is a multi-stage process with complex configuration and execution. Treating DFT as a tool therefore requires a more elaborate workflow to handle these stages and their interactions.

Benchmarking Agents in Materials Science. A growing number of LLM benchmarks have been proposed in material science, including graduate-level question answering (Zaki et al., 2024; Mirza et al., 2024; Zhang et al., 2025; Cheung et al., 2025), research-level task completion (Miret & Krishnan, 2024; Guo et al., 2023), specific tool usage (Huang et al., 2025), retrieval-augmented reasoning (Zhong et al., 2025), and wet-lab experimental design and analysis (Mandal et al., 2025). However, DFT requires multi-dimensional capabilities including scientific knowledge, HPC expertise, and constrained optimization, which are not fully captured by existing benchmarks. The above gaps motivate the design of TRITONDFT and DFTBENCH.

Recent agents can execute useful DFT workflows, but their evaluations usually focus on individual tasks or workflow reliability. TRITONDFT focuses on the decisions that determine practical DFT use: selecting accurate but efficient parameters, configuring parallel execution, completing multi-step workflows, and tracking model cost. Table 1 compares the evaluation scope reported in the cited papers. Among these studies, TRITONDFT is the only framework that evaluates these dimensions together on a common 67-material Quantum ESPRESSO benchmark. Because these systems use different DFT engines and datasets, we compare evaluation scope, not raw performance.

3 TRITONDFT: METHOD AND IMPLEMENTATION

We propose TRITONDFT, an agentic framework for end-to-end DFT workflow automation. As shown in Figure 3, TRITONDFT translates natural-language requests into complete DFT workflows through a **Plan–Execute–Refine** process (Section 3.1), with **Pareto-aware parameter inference** (Section 3.2) to balance accuracy and computational cost and **hardware-aware parallelization** (Section 3.3) to efficiently map computations onto available hardware. We support both `Quantum ESPRESSO` (Giannozzi et al., 2009) and `VASP` (Hafner, 2008) as DFT backends.

Table 1: Comparison on evaluation with recent agentic DFT frameworks.

	DREAMS (Wang et al., 2025)	VASPilot (Liu et al., 2025)	AgenticDFT (Xia et al., 2025)	AutoDFT (Yang et al., 2026)	**TRITONDFT (Ours)**
Evaluation Details					
DFT Engine	QE	VASP	VASP	VASP	**QE + VASP**
Parameter Configuration	Physics Only	Physics Only	Physics Only	Physics Only	**Physics + HPC**
Evaluation Set	27 Sol27LC crystals and adsorption cases	4 materials (2H-TMD only)	27 Sol27LC crystals and adsorption cases	20 materials	**67 materials in 10 categories**
Capabilities Measured in the Evaluation					
Accuracy–Cost Tradeoff	✗	✗	✗	✗	✓
HPC Efficiency	✗	✗	✗	✗	✓
API Cost Report	✗	✗	✗	✓	✓

3.1 PLAN–EXECUTE–REFINE WORKFLOW

The workflow is anchored by a Planner that decomposes a user query into a sequence of DFT subproblems, each grounded in a specific executable. For example, structural relaxation (VC-relax) and self-consistent-field (SCF) calculations are grounded in `pw.x`, while phonon analysis is grounded in `ph.x`. Rather than letting the LLM directly generate executable commands, this grounding constrains each subproblem to a registered executable with structured metadata describing its input/output formats and execution constraints. Adding support for new executables only requires registering new metadata, without modifying the workflow logic.

Within each subproblem, TRITONDFT executes a closed loop with four stages: (1) **Parameter Inference** produces physical parameters $\boldsymbol{\theta}_{\text{phy}}$ (e.g., `ecutwfc`, k-point grid) using a Pareto-aware reasoning mechanism that targets a user-specified accuracy budget while keeping computational cost low (Section 3.2); (2) **Script Generator** synthesizes $\boldsymbol{\theta}_{\text{phy}}$ into a syntactically valid input file conforming to the executable metadata; (3) **Executor** configures hardware parallelization parameters $\boldsymbol{\theta}_{\text{hpc}}$ and submits the job to the underlying compute backend (Section 3.3); (4) **Interpreter** parses the raw output, validates it against the convergence criteria in the metadata, and produces either a result summary on success or a refinement signal on failure.

The workflow is augmented with external domain tools. The parameter inference stage queries the Materials Project (Jain et al., 2013) to retrieve physical information such as initial crystal structures and reference lattice constants. The interpreter stage integrates `pymatgen` (Ong et al., 2013) for symmetry analysis and space group verification. These integrations ground the agent's outputs in established references, complementing the LLM's prior knowledge with verified structural data.

3.2 PARETO-AWARE PARAMETER INFERENCE

DFT parameter configuration is a bi-objective optimization problem: each setting of $\boldsymbol{\theta}_{\text{phy}}$ produces a point in accuracy-cost space, and the user typically requires the cheapest configuration that satisfies a given accuracy target. To navigate this trade-off, the agent must reason about both objectives jointly. We explore the following three forms of *Pareto-aware Reasoning*:

Single-pass estimation. The agent produces a configuration in a single LLM call given the task description and the user-specified accuracy target ΔE^*. The prompt explicitly asks the agent to consider both accuracy and cost when choosing parameters, but no estimation or refinement step is performed. This variant relies entirely on the agent's prior knowledge of DFT parameter behavior.

Self-reflection with lightweight probes. The agent first estimates a candidate configuration, then runs lightweight probes to obtain execution feedback, and finally refines the parameters based on the observed accuracy-cost behavior. For expensive stages such as VC-relax, TRITONDFT uses lightweight SCF probes to test parameter quality. The agent reads the feedback and revises key convergence parameters, such as E_{cut} and the k-point mesh.

Reference-guided self-reflection. The agent receives a category-level reference from a similar material class. This reference gives a plausible scale for E_{cut} and k-point sampling. The agent does not copy it directly. Instead, it adjusts the configuration based on the target structure, dimensionality,

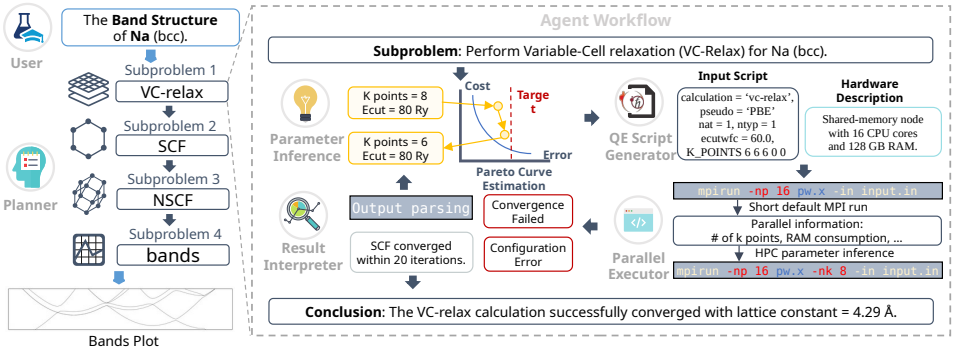

Figure 3: An overview of the TRITONDFT framework.

symmetry, metallicity, and ΔE^*. The reference serves as a soft regularizer to prevent unstable execution or overly aggressive cost cutting. It also helps the agent identify advanced parameters.

3.3 AUTOMATED PARALLELIZATION

The executor automatically sets up parallelization parameters to improve execution efficiency. It takes as input a natural-language hardware specification (e.g., "32 CPU cores on a 128 GB shared-memory node"), together with the generated input scripts and an estimation of the resource cost to finish the calculation. Following common practices adopted by DFT experts, the agent estimates the resource cost via a short default MPI run (within 30 seconds), and extracts key signals such as number of k-points and total DRAM consumption. These signals provide lightweight estimates of parallel scalability and memory footprint to guide setup and avoid oversubscription.

We also find that a common failure mode is that the model tends to use every available parallelization knob once it is exposed in the prompt. This over-parallelization can hurt efficiency, especially for small and medium-sized systems. We therefore add conservative guardrails against over-configuration, avoiding aggressive k-point splitting and unnecessary OpenMP threading.

3.4 USER INTERFACE AND INTERACTION

TRITONDFT is designed as a practical interaction layer for running DFT workflows across different computing environments. We decouple the agent backend from the DFT computation backend, allowing the same agent workflow to operate on both local servers and Slurm-managed clusters.

We provide two interfaces for different usage scenarios. The web interface is intended for relatively small-scale calculations, providing execution visualization at key stages. The advanced user interface allows users to connect TRITONDFT to a Slurm cluster and run the generated workflows. Both interfaces support human-in-the-loop interaction, allowing users to inspect key intermediate outputs, including the execution plan and generated scripts, and provide feedback before proceeding. The interfaces and their interaction workflows are illustrated in Appendix D.

4 DFTBENCH: BENCHMARK FOR LLM AGENTS ON AUTOMATED DFT

DFTBENCH comprises 67 unique crystalline materials, exhibiting diversity in two aspects:

Physical Diversity: DFTBENCH covers 10 distinct material categories, ranging from fundamental electronic phases (Metals, Insulators, Semiconductors) to complex functional and quantum materials (van der Waals, Topological insulators, Ferroelectrics). The material set contains 47 chemical elements, and 23 crystallographic space groups, spanning multiple electronic phases and magnetic ground states. Such diversity in composition, symmetry, and electronic structure requires the model to understand different material properties and precisely configure $\boldsymbol{\theta}_{\text{phy}}$.

Computational Complexity Diversity: DFTBENCH spans system sizes that differ by orders of magnitude in computational cost. The benchmark includes systems ranging from single-atom

primitive cells to polyatomic unit cells, with unit cell volumes from 16.0 to 192.0 Å^3 and total valence electron counts from 1 to 64. Such diversity requires the model to efficiently configure $\boldsymbol{\theta}_{\text{hpc}}$ to handle workloads spanning multiple orders of magnitude.

Evaluation Design. We evaluate the agent along three protocol dimensions:

(i) Accuracy-Cost Trade-off and Pareto-Optimal Parameter Reference. To evaluate numerical accuracy, we define three target energy deviation levels: $\Delta E < 1$, 10, and 20 meV/atom. These thresholds reflect commonly adopted DFT accuracy levels. The stringent threshold of 1 meV/atom is computationally expensive, targeting energy-sensitive properties (e.g., relative phase stability and defect energetics). 10 meV/atom is commonly used for standard high-throughput accuracy (e.g., equilibrium structures and band properties). 20 meV/atom offers the lowest computational cost for coarse-grained screening, structure filtering, and exploratory data generation.

For each material, we provide expert-curated input configurations that satisfy these targets and serve as Pareto-optimal references. Building these references requires material-specific convergence tests over kinetic energy cutoff and k-point grids, as well as advanced parameters for magnetic, topological, and correlated-electron systems. We evaluate agents by comparing their generated configurations with these references under different energy-error tolerances.

(ii) HPC Parallelization. We evaluate the execution time for each test case under a default `mpirun` setup, where `np` denotes the total number of parallel processes, and compare it against agent-generated configurations that leverage advanced DFT parallelization parameters such as `nk` and `ntg`. This evaluation assesses the agent's capability of effective parallel acceleration.

(iii) Cost evaluation. We also measure the number of tokens and the total runtime of LLM calls. These quantities are converted into the corresponding API costs and end-to-end workflow throughput, to evaluate both time and monetary costs of the agent in real-world usage.

5 Evaluation

5.1 Experimental Setup

Implementation and Platform. The main experiments integrate `Quantum ESPRESSO` (v7.4) with a unified API interface for accessing diverse commercial LLMs. The framework is deployed on a high-performance CPU node equipped with an AMD EPYC 9534 64-Core Processor clocked at 3.48 GHz. The CPU supports vector extensions like `AVX-512`, ensuring optimized execution.

Test Cases. We perform benchmarking primarily on four DFT tasks: variable-cell relaxation (VC-relax), self-consistent field (SCF), band gap, and density of states (DOS). These tasks cover the most fundamental steps in DFT. We further include case studies on broader task types, including Γ-point phonon calculations, full phonon dispersion, Raman spectra, and IR spectra.

5.2 DFT Parameter Configuration and Execution

Accuracy and Cost Trade-off Analysis. We evaluate the accuracy–cost trade-off on VC-relax, the key step that determines the relaxed geometry. Table 2 compares TritonDFT using GPT 5.2 with static parameter-selection baselines. Atomate uses its recommended k-point setting and pseudopotential-suggested cutoff, while AiiDA provides `fast`, `balanced`, and `stringent` protocols at different accuracy levels. All configurations are executed and evaluated using the same TritonDFT runtime system. Pass Rate measures whether a configuration satisfies the target energy tolerance, while Normalized Cost C_{norm} (Appendix B.1) estimates computational work from the energy cutoff and k-point grid rather than wall-clock runtime.

The results show a clear tradeoff between accuracy and computation cost. More conservative AiiDA protocols improve accuracy but incur much higher cost, while cheaper static configurations fail more often under stricter accuracy requirements. For strict accuracy targets such as 1 meV/atom, however, single-pass generation is often insufficient, motivating reference guidance and lightweight probes.

We also compare the single-pass parameter configuration capability of different GPT and other foundation models, revealing distinct tendencies in parameter selection. Results are in Appendix

Table 2: Comparison across parameter selection methods under varying energy error thresholds ΔE, which denotes the energy deviation from the highest-accuracy reference calculation.

Method	$\Delta E < 20$ meV/atom		$\Delta E < 10$ meV/atom		$\Delta E < 1$ meV/atom	
	Pass Rate	Cost Factor	Pass Rate	Cost Factor	Pass Rate	Cost Factor
Baselines						
Atomate	78.9%	6.87	74.6%	5.58	35.2%	5.38
AiiDA `fast`	78.9%	4.35	66.2%	3.74	35.2%	3.20
AiiDA `balanced`	83.1%	31.55	80.3%	25.02	50.7%	18.08
AiiDA `stringent`	83.1%	102.60	81.7%	80.34	57.7%	53.39
TritonDFT						
Single-Pass	70.5%	14.29	67.0%	8.95	47.1%	4.23
Reference-Guided	92.3%	3.29	92.3%	3.74	79.1%	4.59
Lightweight Probe	94.0%	4.85	92.5%	3.20	85.1%	2.16

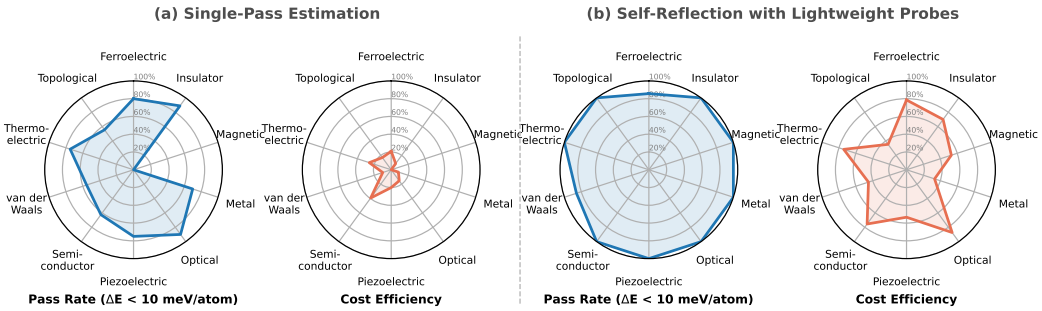

Figure 4: Performance Analysis with Pass Rate and Cost Efficiency across different material types. Cost Efficiency is measured as (1 / Cost Factor), averaged over all passed cases within each type.

C.1. For stricter accuracy targets such as 1 meV/atom, however, single-pass generation is often insufficient, motivating the use of reference guidance and lightweight probes for refinement.

Performance across Material Types. As shown in Figure 4(a), single-pass estimation performs unevenly across material types, with low accuracy on challenging categories such as magnetic materials and metals. This is mainly because these systems are more sensitive to parameter choices, and single-pass estimation often underestimates the required k-point density or convergence settings. At the same time, their cost efficiency is also lower. When lightweight probes and self-reflection are introduced, Figure 4(b) shows that both pass rate and cost efficiency improve across most material types. The remaining failures mainly arise from the limited probe budget, which prevents the model from exploring different parameters and identifying issues such as an insufficient energy cutoff.

Self-Reflection with Probe Analysis. We further analyze how lightweight probes improve parameter selection through iterative feedback. As shown in Figure 5(a), static AiiDA protocols and single-pass generation can be inaccurate or overly expensive on ZnO. With probe feedback, TRITONDFT progressively approaches the Pareto frontier with 5 iterations. Figure 5(b) illustrates the exploration process. We compare TRITONDFT with two deterministic search strategies. *Scan* starts from the AiiDA `fast` configuration and adjusts the k-point density with a fixed step of 4. When the search crosses the energy-error threshold, it reverses direction and halves the step size. *Bisection* uses binary search between the AiiDA `balanced` and `stringent` configurations, progressively narrowing the range toward the target accuracy. Among the two deterministic baselines, *Scan* performs better and more closely resembles the sweep-based convergence tests commonly used by scientists. In comparison, TRITONDFT achieves higher accuracy at a modestly higher cost by using agent reasoning to select the next probe. It typically reaches the high-accuracy region within the first few probes, then refines nearby configurations to reduce cost.

Figure 5(c) shows the end-to-end overhead of probing. Since VC-relax itself requires multiple SCF calls, the six additional probe runs can account for a sizable fraction of the total computation. This

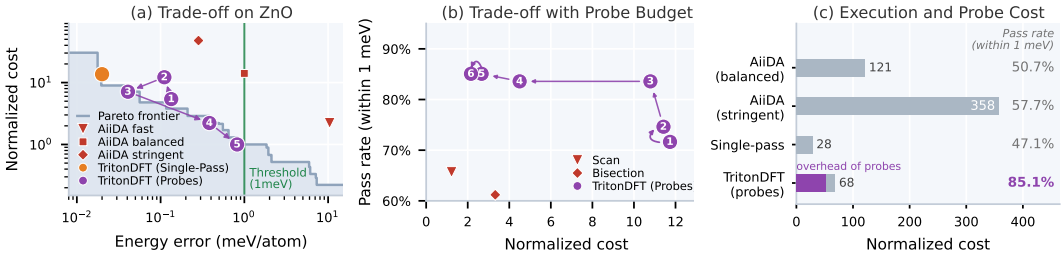

Figure 5: (a) Accuracy–cost tradeoff on ZnO. (b) Tradeoff of TRITONDFT and deterministic baselines across probe budgets. (c) End-to-end cost breakdown of VC-relax with and without probes.

overhead substantially improves the final accuracy, while the more accurate AiiDA protocols incur much higher VC-relax costs due to their expensive configurations.

Advanced Parameters. We also evaluate advanced physical settings, including vdW corrections, Hubbard U, spin polarization, and spin–orbit coupling. Spin polarization is handled reliably by most stronger models, while vdW and Hubbard U require more material-specific knowledge. SOC is treated separately because its use depends on the target property, so we recommend specifying it explicitly when needed. Detailed results are provided in Appendix C.2.

Automatic Parallelization. We further evaluate the execution efficiency of LLM-generated parallel configurations during structure relaxation. GPT 5.2 and the Claude 4.5 models perform consistently better than other models, achieving up to 21.7% speedup over the default MPI configuration at 64 cores. We attribute this advantage to their stronger code-generation capabilities, which help them better apply parallelization rules, such as selecting k-point parallelism based on the number of available k-points. Detailed results are provided in Appendix C.3.

End-to-End Workflow Evaluation. We evaluate four tasks (VC-relax, SCF, Bandgap, DOS) against the highest-accuracy human-configured reference, using relative lattice-parameter deviation, relative total-energy-per-atom error, absolute band-gap error, and absolute Fermi-level error, respectively. Gemini 2.5 Pro achieves the lowest VC-relax error (5.5%, vs. 5.7% for GPT 5.2), while GPT 5.2 achieves the lowest SCF (0.04%), band-gap (0.34 eV), and DOS Fermi-level (1.32 eV) errors. Claude models are comparable on most tasks. Overall, stronger-reasoning models tend to produce smaller errors. See Appendix C.5 for details.

Workflow Throughput. We evaluate the end-to-end throughput of the DFT setup workflow, excluding DFT execution time and focusing only on automation overhead. TRITONDFT achieves an effective request rate of approximately 10–100 queries/hour, depending on task complexity and API latency. In contrast, our survey on PhD-level practitioners indicates that manual setup sustains less than 1 request/hour, giving TRITONDFT a $> 10\times$ efficiency improvement. See Appendix C.6.

Monetary Cost. We compute input and output tokens of TRITONDFT across three tasks and estimate cost based on official API pricing. Gemini 2.5 Flash is the cheapest at \$0.01–0.03 per query. GPT-5 series and Gemini 2.5 Pro fall at \$0.04–0.18 per query. The Claude series reaches the highest \$0.44 per query for complex calculations. Overall, the cost of TRITONDFT to setup a DFT workflow remains modest, see Appendix C.7.

5.3 CASE STUDY 1: COMPARISON WITH DREAMS

We further compare TRITONDFT with DREAMS (Wang et al., 2025), a prior open-source agentic DFT framework, on Sol27LC, one of its main evaluation benchmarks. DREAMS is effective for its target property: both methods reproduce the converged lattice constants with small errors. We further evaluate whether the selected parameters remain converged for subsequent calculations. As shown in Table 3, DREAMS-selected parameters show median deviations much larger than TRITONDFT. This shows that convergence for the target property does not necessarily imply high-fidelity convergence for other properties.

The comparison highlights the difference between a fixed prompt template and task-dependent plan and execution. DREAMS provides an effective policy for its target calculation, while TRITONDFT

Table 3: Comparison with DREAMS on Sol27LC. We report the median deviation from the converged reference using the configuration selected by each method.

Method	Deviation from the target property		
	VC-relax lattice a (%)	SCF energy (meV/atom)	DOS Fermi level (meV)
DREAMS	0.272	6.75	25.5
TRITONDFT	0.183	0.50	0.80

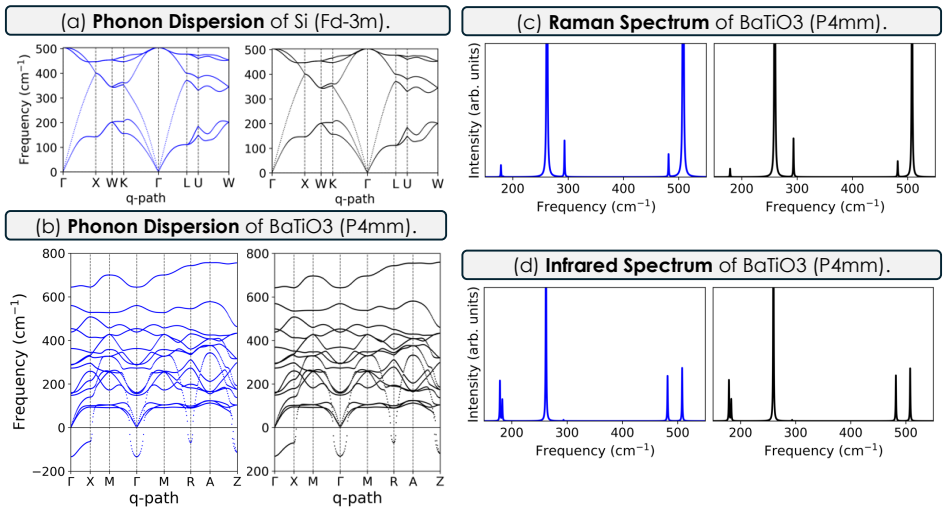

Figure 6: Comparison of TRITONDFT's DFT results (marked as **Blue**) with human-curated DFT results (marked as **Black**) on phonon dispersions, Raman spectra and infrared spectra.

configures and probes parameters according to the requested task and accuracy target. DFTBench evaluates this capability across 67 materials from 10 categories and multiple DFT tasks.

5.4 CASE STUDY 2: BROADER TASK TYPES WITH HUMAN RESULT COMPARISON

TRITONDFT also generalizes beyond the main benchmark to broader DFT tasks. Figure 6 shows representative phonon, Raman, and IR calculations, where the agent automatically constructs and executes the required workflows. For phonon calculations, it organizes the QE pipeline using `pw.x`, `ph.x`, `q2r.x`, and `matdyn.x`, while selecting q-point sampling and task-specific settings. These examples show that TRITONDFT can automate diverse DFT workflows without relying on fixed task-specific templates (Wang et al., 2025).

We further compare the results with human-curated DFT calculations in Appendix C.8. For $BaTiO_3$, the nonzero Γ-point phonon frequencies agree within approximately 2%, while other materials show larger and mode-dependent deviations. These differences coincide with changes in material-specific settings and relaxed structures; for example, the relaxed c parameter of FeSe differs between the two calculations. Since multiple settings differ simultaneously, we do not attribute the deviations to any individual choice. These results demonstrate successful automation of the workflows while showing that material-specific configuration and validation of the relaxed structure remain important for quantitative phonon accuracy.

6 CONCLUSION

We presented TRITONDFT, an agentic framework for automating DFT workflows with expert-informed planning, Pareto-aware parameter inference, and HPC parallelization. We also introduced DFTBENCH to evaluate accuracy–cost trade-off, parallel efficiency, and cost-effectiveness across diverse materials. Our results show that TRITONDFT exhibits good capabilities for parameter configuration and execution across different materials and tasks.

AI USE STATEMENT

In this work, we used generative AI tools to assist with translation. We have not used generative AI tools to propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, clean and reformat dataset, support qualitative and thematic data analysis, interpret results, and generate synthetic data sets, help develop theoretical models or conceptual frameworks. To formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs are not applicable to this work. Additionally, we used generative AI tools to create or edit software code. We have reviewed all AI-assisted work. AI-edited code was inspected, tested, and used by the authors in the actual experimental workflow, while AI-edited manuscript text was manually checked and further refined by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide the source code, experimental configurations, prompts, and evaluation scripts needed to reproduce our results at an anonymous repository: `https://anonymous.4open.science/r/TritonDFT-43C7/`.

REFERENCES

Jerry Cheung, Yuchen Zhuang, Yinghao Li, Pranav Shetty, Wantian Zhao, Sanjeev Grampurohit, Rampi Ramprasad, and Chao Zhang. Polyie: A dataset of information extraction from polymer material scientific literature. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 2370–2385, 2024.

Jerry Junyang Cheung, Shiyao Shen, Yuchen Zhuang, Yinghao Li, Rampi Ramprasad, and Chao Zhang. Msqa: Benchmarking llms on graduate-level materials science reasoning and knowledge. *arXiv preprint arXiv:2505.23982*, 2025.

Yuan Chiang, Elvis Hsieh, Chia-Hong Chou, and Janosh Riebesell. Llamp: Large language model made powerful for high-fidelity materials knowledge retrieval and distillation. *arXiv preprint arXiv:2401.17244*, 2024.

Yahao Dai, Henry Chan, Aikaterini Vriza, Jingyuan Fan, Fredrick Kim, Yunfei Wang, Wei Liu, Naisong Shan, Jing Xu, Max Weires, et al. Adaptive ai decision interface for autonomous electronic material discovery. *Nature Chemical Engineering*, pp. 1–11, 2025.

Qianggang Ding, Santiago Miret, and Bang Liu. Matexpert: Decomposing materials discovery by mimicking human experts. *arXiv preprint arXiv:2410.21317*, 2024.

Paolo Giannozzi, Stefano Baroni, Nicola Bonini, Matteo Calandra, Roberto Car, Carlo Cavazzoni, Davide Ceresoli, Guido L Chiarotti, Matteo Cococcioni, Ismaila Dabo, et al. Quantum espresso: a modular and open-source software project for quantum simulations of materials. *Journal of physics: Condensed matter*, 21(39):395502, 2009.

Taicheng Guo, Bozhao Nan, Zhenwen Liang, Zhichun Guo, Nitesh Chawla, Olaf Wiest, Xiangliang Zhang, et al. What can large language models do in chemistry? a comprehensive benchmark on eight tasks. *Advances in Neural Information Processing Systems*, 36:59662–59688, 2023.

Jürgen Hafner. Ab-initio simulations of materials using vasp: Density-functional theory and beyond. *Journal of computational chemistry*, 29(13):2044–2078, 2008.

Kausik Hira, Mohd Zaki, Dhruvil Sheth, NM Anoop Krishnan, et al. Reconstructing the materials tetrahedron: challenges in materials information extraction. *Digital Discovery*, 3(5):1021–1037, 2024.

Pierre Hohenberg and Walter Kohn. Inhomogeneous electron gas. *Physical review*, 136(3B):B864, 1964.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *The twelfth international conference on learning representations*, 2023.

Xu Huang, Junwu Chen, Yuxing Fei, Zhuohan Li, Philippe Schwaller, and Gerbrand Ceder. Cascade: Cumulative agentic skill creation through autonomous development and evolution. *arXiv preprint arXiv:2512.23880*, 2025.

Anubhav Jain, Shyue Ping Ong, Geoffroy Hautier, Wei Chen, William Davidson Richards, Stephen Dacek, Shreyas Cholia, Dan Gunter, David Skinner, Gerbrand Ceder, et al. Commentary: The materials project: A materials genome approach to accelerating materials innovation. *APL materials*, 1(1), 2013.

Walter Kohn and Lu Jeu Sham. Self-consistent equations including exchange and correlation effects. *Physical review*, 140(4A):A1133, 1965.

Shrinidhi Kumbhar, Venkatesh Mishra, Kevin Coutinho, Divij Handa, Ashif Iquebal, and Chitta Baral. Hypothesis generation for materials discovery and design using goal-driven and constraint-guided llm agents. *arXiv preprint arXiv:2501.13299*, 2025.

Ask Hjorth Larsen, Jens Jørgen Mortensen, Jakob Blomqvist, Ivano E Castelli, Rune Christensen, Marcin Dułak, Jesper Friis, Michael N Groves, Bjørk Hammer, Cory Hargus, et al. The atomic simulation environment—a python library for working with atoms. *Journal of Physics: Condensed Matter*, 29(27):273002, 2017.

Jiaxuan Liu, Tiannian Zhu, Caiyuan Ye, Zhong Fang, Hongming Weng, and Quansheng Wu. Vaspilot: Mcp-facilitated multi-agent intelligence for autonomous vasp simulations. *Chinese Physics B*, 34(11):117106, 2025.

Siyu Liu, Tongqi Wen, ASL Subrahmanyam Pattamatta, and David J Srolovitz. A prompt-engineered large language model, deep learning workflow for materials classification. *Materials Today*, 80:240–249, 2024.

Andres M. Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, and Philippe Schwaller. Augmenting large language models with chemistry tools. *Nature Machine Intelligence*, 6(5):525–535, 2024.

Indrajeet Mandal, Jitendra Soni, Mohd Zaki, Morten M Smedskjaer, Katrin Wondraczek, Lothar Wondraczek, Nitya Nand Gosvami, and NM Anoop Krishnan. Evaluating large language model agents for automation of atomic force microscopy. *Nature Communications*, 16(1):9104, 2025.

Kiran Mathew, Joseph H Montoya, Alireza Faghaninia, Shyam Dwarakanath, Muratahan Aykol, Hanmei Tang, Iek-heng Chu, Tess Smidt, Brandon Bocklund, Matthew Horton, et al. Atomate: A high-level interface to generate, execute, and analyze computational materials science workflows. *Computational Materials Science*, 139:140–152, 2017.

Andrew D McNaughton, Gautham Krishna Sankar Ramalaxmi, Agustin Kruel, Carter R Knutson, Rohith A Varikoti, and Neeraj Kumar. Cactus: Chemistry agent connecting tool usage to science. *ACS omega*, 9(46):46563–46573, 2024.

Santiago Miret and Nandan M Krishnan. Are llms ready for real-world materials discovery? *arXiv preprint arXiv:2402.05200*, 2024.

Adrian Mirza, Nawaf Alampara, Sreekanth Kunchapu, Martiño Ríos-García, Benedict Emoekabu, Aswanth Krishnan, Tanya Gupta, Mara Schilling-Wilhelmi, Macjonathan Okereke, Anagha Aneesh, et al. Are large language models superhuman chemists? *arXiv preprint arXiv:2404.01475*, 2024.

Shyue Ping Ong, William Davidson Richards, Anubhav Jain, Geoffroy Hautier, Michael Kocher, Shreyas Cholia, Dan Gunter, Vincent L Chevrier, Kristin A Persson, and Gerbrand Ceder. Python materials genomics (pymatgen): A robust, open-source python library for materials analysis. *Computational Materials Science*, 68:314–319, 2013.

Giovanni Pizzi, Andrea Cepellotti, Riccardo Sabatini, Nicola Marzari, and Boris Kozinsky. Aiida: automated interactive infrastructure and database for computational science. *Computational Materials Science*, 111:218–230, 2016.

Mara Schilling-Wilhelmi, Martiño Ríos-García, Sherjeel Shabih, María Victoria Gil, Santiago Miret, Christoph T Koch, José A Márquez, and Kevin Maik Jablonka. From text to insight: large language models for chemical data extraction. *Chemical Society Reviews*, 2025.

Yu Song, Santiago Miret, and Bang Liu. Matsci-nlp: Evaluating scientific language models on materials science language tasks using text-to-schema modeling. *arXiv preprint arXiv:2305.08264*, 2023.

Nathan J Szymanski, Bernardus Rendy, Yuxing Fei, Rishi E Kumar, Tanjin He, David Milsted, Matthew J McDermott, Max Gallant, Ekin Dogus Cubuk, Amil Merchant, et al. An autonomous laboratory for the accelerated synthesis of novel materials. *Nature*, 624(7990):86–91, 2023.

Hanchen Wang, Tianfan Fu, Yuanqi Du, Wenhao Gao, Kexin Huang, Ziming Liu, Payal Chandak, Shengchao Liu, Peter Van Katwyk, Andreea Deac, et al. Scientific discovery in the age of artificial intelligence. *Nature*, 620(7972):47–60, 2023.

Ziqi Wang, Hongshuo Huang, Hancheng Zhao, Changwen Xu, Shang Zhu, Jan Janssen, and Venkatasubramanian Viswanathan. Dreams: Density functional theory based research engine for agentic materials simulation. *arXiv preprint arXiv:2507.14267*, 2025.

Zeyu Xia, Jinzhe Ma, Congjie Zheng, Shufei Zhang, Yuqiang Li, Hang Su, P Hu, Changshui Zhang, Xingao Gong, Wanli Ouyang, et al. An agentic framework for autonomous materials computation. *arXiv preprint arXiv:2512.19458*, 2025.

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

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Mohd Zaki, NM Anoop Krishnan, et al. Mascqa: investigating materials science knowledge of large language models. *Digital Discovery*, 3(2):313–327, 2024.

Huan Zhang, Yu Song, Ziyu Hou, Santiago Miret, and Bang Liu. Honeycomb: A flexible llm-based agent system for materials science. *arXiv preprint arXiv:2409.00135*, 2024.

Junkai Zhang, Jingru Gan, Xiaoxuan Wang, Zian Jia, Changquan Gu, Jianpeng Chen, Yanqiao Zhu, Mingyu Derek Ma, Dawei Zhou, Ling Li, et al. Matscibench: Benchmarking the reasoning ability of large language models in materials science. *arXiv preprint arXiv:2510.12171*, 2025.

Xianrui Zhong, Bowen Jin, Siru Ouyang, Yanzhen Shen, Qiao Jin, Yin Fang, Zhiyong Lu, and Jiawei Han. Benchmarking retrieval-augmented generation for chemistry. *arXiv preprint arXiv:2505.07671*, 2025.

A Details of DFTBench

A.1 Human Baseline Survey Methodology

Purpose and participants. We conducted an internal survey of 24 researchers with experience in computational materials science to estimate the manual effort required for an end-to-end DFT band-structure workflow. The survey was administered through Google Forms and covered participant background, structure preparation, parameter selection, input generation, job submission, and post-processing. The questionnaire is summarized in Table 4.

Participants estimated the time normally spent on each stage of their usual workflow. The total-workflow estimate excluded queue waiting and DFT execution time. Because participants

were not asked to complete a common timed calculation, the survey provides estimates of typical manual effort rather than a controlled human-versus-agent benchmark. Responses were collected using predefined time intervals and analyzed only in aggregate.

Questionnaire design. The survey contained 36 questions organized into the following parts:

Table 4: Summary of the human-baseline survey questionnaire.

Survey section	Information collected
Participant background	DFT experience level, primary DFT code, and approximate number of completed projects (Questions 1–3).
Structure acquisition	Structure source, time required to obtain a usable structure, and additional time required for cleaning or correcting the structure (Questions 4–6).
Symmetry verification	Whether symmetry was checked, tools used, verification time, and time required to resolve symmetry inconsistencies (Questions 7–10).
Input conversion	Method used to convert the structure into a DFT input file, conversion and debugging time, and time required to adapt automatically generated inputs (Questions 11–13).
Pseudopotentials	Pseudopotential source, selection time, and whether the initial pseudopotential had to be replaced (Questions 14–16).
Convergence testing	Whether cutoff-energy and k-point convergence tests were performed and the researcher time spent setting up and analyzing these tests (Questions 17–20).
Workflow setup	Time required to prepare relaxation, SCF, and NSCF inputs; method used to generate the high-symmetry k-path; and time spent preparing the path (Questions 21–26).
Job submission	Computing environment, hardware, scheduler, and time spent preparing and submitting job scripts (Questions 27–30).
Post-processing	Plotting tools and time spent extracting and plotting the band structure, band gap, valence-band maximum, and conduction-band minimum (Questions 31–32).
Overall assessment	Total manual time, most time-consuming step, most error-prone step, and estimated time saved through automation (Questions 33–36).

A.2 Information of Materials

DFTBench comprises 67 distinct materials spanning 10 different material categories. Some materials belong to multiple categories (e.g., semiconductor, optical). All material information is summarized in Table 5.

Table 5: Benchmark dataset of materials used for DFT workflow evaluation, including material category (multiple categories separated by commas), space group, and number of atoms in the conventional unit cell.

#	Material	Category	Space Group	Atoms
1	CaO	Insulator	$Fm\bar{3}m$	2
2	KCl	Insulator	$Fm\bar{3}m$	2
3	LiF	Insulator	$Fm\bar{3}m$	2
4	MgF_2	Insulator	$P4_2/mnm$	6
5	MgO	Insulator	$Fm\bar{3}m$	2
6	NaCl	Insulator	$Fm\bar{3}m$	2
7	NaF	Insulator	$Fm\bar{3}m$	2
8	SiO_2	Insulator	$P3_121$	9
9	BN (hex)	Insulator	$P6_3/mmc$	4
10	$C_{diamond}$	Insulator	$Fd\bar{3}m$	2
11	BN	Insulator, Piezoelectric	$Fm\bar{3}m$	2
12	AlN	Insulator, Piezoelectric	$P6_3mc$	4
13	BeO	Insulator, Piezoelectric	$P6_3mc$	4

Continued on next page

#	Material	Category	Space Group	Atoms
14	$BaTiO_3$	Insulator, Ferroelectric	P4mm	5
15	HfO_2	Insulator, Ferroelectric	$Pca2_1$	12
16	$NaNbO_3$	Insulator, Ferroelectric	$Pm\bar{3}m$	5
17	$LiTaO_3$	Insulator, Ferroelectric, Piezoelectric	R3c	10
18	$PbTiO_3$	Insulator, Ferroelectric, Piezoelectric	P4mm	5
19	Cr_2O_3	Insulator, Magnetic	R3c	10
20	FeO	Insulator, Magnetic	P4/mmm	8
21	MnO	Insulator, Magnetic	P4/mmm	8
22	NiO	Insulator, Magnetic	P4/mmm	8
23	Al_2O_3	Insulator, Optical	$R\bar{3}c$	10
24	BaF_2	Insulator, Optical	$Fm\bar{3}m$	3
25	CaF_2	Insulator, Optical	$Fm\bar{3}m$	3
26	Bi_2Se_3	Insulator, van der Waals, Topological	$R\bar{3}m$	5
27	Ag	Metal	$Fm\bar{3}m$	1
28	Al	Metal	$Fm\bar{3}m$	1
29	Au	Metal	$Fm\bar{3}m$	1
30	Cu	Metal	$Fm\bar{3}m$	1
31	Fe	Metal	I4/mmm	1
32	K	Metal	$Im\bar{3}m$	1
33	Li	Metal	$Im\bar{3}m$	1
34	Mo	Metal	$Im\bar{3}m$	1
35	Na	Metal	$Im\bar{3}m$	1
36	W	Metal	$Im\bar{3}m$	1
37	MgB_2	Metal	P6/mmm	3
38	Nb	Metal	$Im\bar{3}m$	1
39	Pb	Metal	$Fm\bar{3}m$	1
40	Sn (β-Sn)	Metal	$I4_1/amd$	2
41	V	Metal	$Im\bar{3}m$	1
42	Co	Metal, Magnetic	$P6_3/mmc$	2
43	Fe (bcc)	Metal, Magnetic	$Im\bar{3}m$	1
44	Fe_3O_4	Metal, Magnetic	$Fd\bar{3}m$	14
45	Ni	Metal, Magnetic	$Fm\bar{3}m$	1
46	FeSe	Metal, van der Waals	P4/nmm	4
47	$NbSe_2$	Metal, van der Waals	$P6_3/mmc$	6
48	AlP	Semiconductor	$F\bar{4}3m$	2
49	CdTe	Semiconductor	$F\bar{4}3m$	2
50	GaAs	Semiconductor	$F\bar{4}3m$	2
51	Ge	Semiconductor	$Fd\bar{3}m$	2
52	InP	Semiconductor	$F\bar{4}3m$	2
53	Si	Semiconductor	$Fd\bar{3}m$	2
54	ZnS	Semiconductor	$F\bar{4}3m$	2
55	ZnSe	Semiconductor	$F\bar{4}3m$	2
56	PbTe	Semiconductor, Topological	$Fm\bar{3}m$	2
57	SnTe	Semiconductor, Topological	$Fm\bar{3}m$	2
58	Mg_2Ge	Semiconductor, Thermoelectric	$Fm\bar{3}m$	3
59	Mg_2Si	Semiconductor, Thermoelectric	$Fm\bar{3}m$	3
60	SiGe	Semiconductor, Thermoelectric	$F\bar{4}3m$	2
61	GeTe	Semiconductor, Ferroelectric	R3m	2
62	CdS	Semiconductor, Piezoelectric	$P6_3mc$	4
63	CdSe	Semiconductor, Piezoelectric	$P6_3mc$	4
64	GaN	Semiconductor, Piezoelectric	$P6_3mc$	4
65	ZnO	Semiconductor, Piezoelectric	$P6_3mc$	4
66	Bi	Topological	$R\bar{3}m$	2
67	Na_3Bi	Topological	$P6_3/mmc$	8

A.3 QUESTION TEMPLATES

DFTBENCH provides question templates for a diverse set of tasks. For each material, the placeholders {formula} and {space group} in the templates are instantiated accordingly. The

required accuracy is also parameterized as {dEnergy}, which specifies the target energy tolerance for the task. The question templates used for evaluation are listed below:

VC-relax: For {formula} with space group {space group}, perform a variable-cell relaxation (vc-relax) calculation. Require an energy convergence of {dEnergy}. Use PBE pseudopotentials. Return the fully relaxed structure (atomic parameters).

SCF: For {formula} with space group {space group}, run a self-consistent field (SCF) calculation and report the converged total energy per formula unit in eV. Require an energy convergence of {dEnergy}. Use PBE pseudopotentials.

Band Gap: For {formula} with space group {space group}, use a full workflow consisting of: 1) vc-relax, 2) scf, 3) nscf, 4) band structure calculation and 5) band post-processing. Require an energy convergence of {dEnergy}. Use PBE pseudopotentials. Do not need to explicitly return band structure. Just finish the calculation.

DOS: For {formula} with space group {space group}, use a full workflow consisting of: 1) vc-relax, 2) scf, 3) nscf, 4) band structure calculation, 5) band post-processing, and 6) DOS post-processing. Require an energy convergence of {dEnergy}. Use PBE pseudopotentials. Do not need to explicitly return band structure and DOS. Just finish the calculation.

Currently, our evaluation covers the four most essential DFT task types. In the future, we will support additional question templates and datasets. Notably, new tasks can be incorporated in a modular manner, making such extensions fully feasible.

B Computational details for human-performed DFT calculations

All density functional theory (DFT) calculations were performed using Quantum ESPRESSO. Structural relaxations were carried out with a total energy convergence threshold of 1.0×10^{-9} Ry and a force convergence threshold of 1.0×10^{-8} Ry/Bohr. For the self-consistent field (SCF) cycles, the electronic convergence threshold was set to 1.0×10^{-10} Ry, and a mixing parameter $\beta = 0.7$ was employed to ensure stable convergence. The plane-wave kinetic energy cutoff and Monkhorst–Pack k-point grids were determined from systematic convergence tests, requiring the total energy to be converged within 1 meV per atom. The converged cutoff energy and k-point mesh were subsequently used for all subsequent SCF calculations. For non-self-consistent field (NSCF) calculations, denser k-point meshes were adopted to obtain accurate electronic properties such as band structures and density of states. The k-point density for NSCF calculations was increased on a case-by-case basis depending on the specific property being evaluated. Complete convergence data, including cutoff energies, k-point grids, and total energies, are provided in the accompanying CSV file to ensure reproducibility.

B.1 Estimation of Normalized cost

The number of plane waves in a DFT calculation is controlled by the kinetic energy cutoff (E_{cut}), which defines the radius of the reciprocal-space sphere used in the plane-wave expansion. The number of plane waves scales as

$$N_{\mathrm{PW}} \propto E_{\mathrm{cut}}^{3/2}, \tag{1}$$

while the computational cost scales linearly with the total number of k-points, $N_k = k_x k_y k_z$.

Accordingly, the overall computational cost scales approximately as

$$C \propto E_{\mathrm{cut}}^{3/2}\, N_k. \tag{2}$$

To compare different parameter choices, we define a normalized cost metric as

$$C_{\mathrm{norm}} = \frac{(E_{\mathrm{cut,agent}})^{3/2}\,(k_x k_y k_z)_{\mathrm{agent}}}{(E_{\mathrm{cut,ground\,truth}})^{3/2}\,(k_x k_y k_z)_{\mathrm{ground\,truth}}}. \tag{3}$$

Table 6: Single-pass model performance on the accuracy-cost tradeoff under varying energy error thresholds ΔE, which denotes the energy deviation from the highest-accuracy reference calculation.

Model	$\Delta E < 20$ meV/atom		$\Delta E < 10$ meV/atom		$\Delta E < 1$ meV/atom	
	Pass Rate	Cost Factor	Pass Rate	Cost Factor	Pass Rate	Cost Factor
GPT 5.2	70.5%	14.29	67.0%	8.95	47.1%	4.23
GPT 5.1	39.3%	6.22	32.9%	4.21	9.8%	2.24
GPT 4o	52.8%	1.85	38.2%	1.28	13.6%	0.50
Gemini 2.5 Pro	59.6%	3.77	53.9%	2.95	14.9%	1.24
Gemini 2.5 Flash	23.6%	1.85	16.9%	1.68	2.3%	0.78
Claude Opus 4.5	9.0%	1.62	5.6%	1.33	4.5%	0.58
Claude Sonnet 4.5	30.3%	2.38	25.8%	1.93	21.6%	0.87

B.2 Calculation of Band Gap

Band gaps were computed from the highest occupied molecular orbital (HOMO) and lowest unoccupied molecular orbital (LUMO) energies obtained from the NSCF calculations. For certain materials, electronic smearing was required to achieve stable convergence. In such cases, Quantum ESPRESSO does not explicitly report the HOMO and LUMO energies and instead provides only the Fermi energy. As a result, the band gaps extracted for these systems may appear as zero in the accompanying supplementary dataset. These materials require additional post-processing or alternative workflows for an accurate determination of the band gap.

In this work, our primary objective was to benchmark the agent's performance within a standard automated workflow. Implementation of specialized procedures for robust band gap extraction in smeared calculations is planned for future versions of the framework.

C Additional Experimental Results

C.1 Result analysis across different models

We benchmark the performance on Single-Pass parameter configuration generation of eight state-of-the-art models across three major families: OpenAI's GPT 5.2, GPT 5.1, and GPT 4o; Google's Gemini 2.5 Pro and Gemini 2.5 Flash; and Anthropic's Claude Opus 4.5 and Claude Sonnet 4.5. This selection spans a broad spectrum of model architectures and scales, from flagship frontier models to more cost-efficient ones.

Among the models in Table 6, GPT 5.2 achieves the highest pass rates across all thresholds, but with higher cost, reaching an average Normalized Cost of 14.29. GPT 5.1, GPT 4o, and Gemini 2.5 Pro are more cost-balanced at the 20 and 10 meV/atom thresholds, but their pass rates fall below 15% under the stricter 1 meV/atom threshold. The Claude 4.5 family performs worse, with Opus underperforming Sonnet. Parameter-level analysis shows that Opus 4.5 tends to choose aggressively low-cost configurations, with Cost below 1.62, but leading to frequent accuracy violations.

C.2 Advanced Parameters

Table 7 reports the accuracy of advanced-parameter setup across models. Spin polarization is handled reliably by most stronger models, while vdW and Hubbard U remain more model-dependent because they require additional material-specific knowledge beyond basic composition. Claude Opus 4.5 performs best overall, while Gemini 2.5 Pro is strongest on vdW settings. These results suggest that advanced physical settings are generally more challenging than numerical parameters such as cutoff and k-point density, motivating explicit user guidance or external references when such settings are important.

We treat SOC separately, since its use depends strongly on the target property rather than on the material alone. For the same set of SOC-relevant materials, GPT-5.2 enables SOC more

Table 7: Model accuracy on advanced-parameter setup. # denotes the number of materials in DFTBENCH for which the corresponding advanced parameter is required to be set.

Model	Total (#=15)	vdW (#=5)	Hubbard U (#=5)	Spin (#=10)
GPT 5.2	33.3%	0.0%	0.0%	100.0%
GPT 5.1	33.3%	0.0%	0.0%	100.0%
GPT 4o	0.0%	0.0%	0.0%	0.0%
Gemini 2.5 Pro	60.0%	60.0%	20.0%	100.0%
Gemini 2.5 Flash	46.7%	40.0%	0.0%	100.0%
Claude Opus 4.5	66.7%	0.0%	100.0%	100.0%
Claude Sonnet 4.5	33.3%	0.0%	0.0%	100.0%

Table 8: Average speedup (%) of LLM-generated parallel configurations over the default MPI baseline. Simple uses a basic prompt, while Improved adds empirical guardrails to prevent over-configuration.

Model	16 Cores		32 Cores		64 Cores	
	Simple	Improved	Simple	Improved	Simple	Improved
GPT 5.2	+5.8	+12.1	-6.8	+0.8	**+21.2**	+21.2
GPT 5.1	-7.7	-2.1	-20.1	+0.8	-32.0	+8.3
GPT 4o	-31.9	+2.0	-77.7	-30.9	-31.3	+3.5
Gemini 2.5 Pro	-13.9	+10.4	-50.6	-29.3	-9.0	+10.8
Gemini 2.5 Flash	-42.0	-14.4	-62.5	-22.1	-24.9	+10.4
Claude Opus 4.5	**+11.1**	**+15.0**	**+7.7**	**+16.0**	+13.8	+21.6
Claude Sonnet 4.5	+6.7	+14.7	-7.8	+10.2	-4.8	**+21.7**

often for band-gap calculations than for relaxation. We therefore recommend treating SOC as a user-configurable setting and specifying it explicitly when relativistic effects are relevant.

C.3 AUTOMATED PARALLELIZATION

We evaluate the execution efficiency of the structure relaxation phase, which accounts for over 50% of the end-to-end runtime. Table 8 reports the average speedup of LLM-generated parallel configurations over the default MPI baseline, `mpirun -np <cores>`. We only include cases with baseline runtime exceeding 30 s, so that the comparison excludes MPI startup noise data. This yields 24, 20, and 15 cases at 16, 32, and 64 cores, respectively.

Under the naive prompt, stronger models already achieve useful gains: Claude Opus 4.5 remains positive at all core counts, and GPT 5.2 reaches up to $+21.2\%$ at 64 cores. In contrast, weaker models are much less reliable. For example, GPT 4o drops to -77.7% at 32 cores, while Gemini 2.5 Flash drops to -62.5% at 32 cores. These losses come from missing probe-derived constraints, such as matching `-nk` to the IBZ k-point structure and avoiding excessive pools, task groups, or threads when the system is too small for them.

The prompt with empirical guardrails addresses part of the failures: unnecessary over-configuration. Instead of letting the executor set many pools, threads, or decomposition parameters at once, it keeps the command conservative by default and only adds extra parallel settings when the probe supports further decomposition. This improves robustness across models, with the largest per-setting gain reaching 33.9% at 16 cores, 46.8% at 32 cores, and 40.3% at 64 cores.

C.4 RESULT ANALYSIS OF REFERENCE-GUIDED SELF-REFLECTION

Among the evaluated models, structured self-reflection with category-level reference anchors substantially improves pass rates across all accuracy thresholds, as shown in Table 9. It also produces more consistent results across models. At the strictest threshold, for example, the range of pass rates narrows from 4.5–47.1% under Single-Pass estimation to 78.8–82.0% with reference guidance.

Table 9: Model performance using Reference-guided self-reflection Strategy.

Model	$\Delta E < 20$ meV/atom		$\Delta E < 10$ meV/atom		$\Delta E < 1$ meV/atom	
	Pass Rate	Cost Factor	Pass Rate	Cost Factor	Pass Rate	Cost Factor
GPT 5.2	92.3%	3.29	92.3%	3.74	79.1%	4.59
GPT 5.1	92.2%	3.72	91.1%	4.39	79.8%	4.45
GPT 4o	90.1%	2.82	91.2%	3.71	81.3%	4.39
Gemini 2.5 Pro	95.6%	3.34	95.5%	3.97	82.0%	4.81
Gemini 2.5 Flash	91.5%	2.84	93.4%	3.41	78.8%	4.25
Claude Opus 4.5	90.1%	2.74	90.1%	3.50	79.1%	4.20
Claude Sonnet 4.5	88.1%	2.89	87.2%	3.66	81.0%	4.43

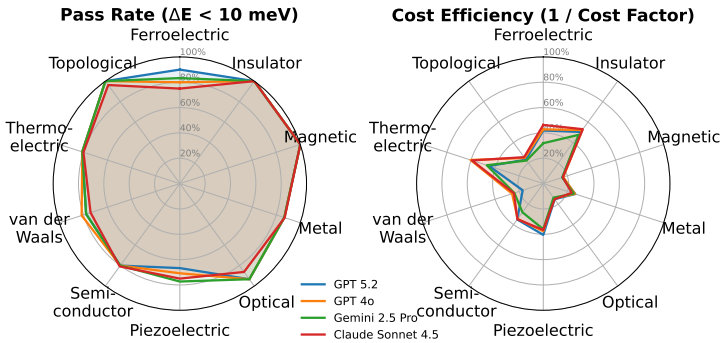

Figure 7: Performance analysis of the model using reference-guided self-reflection strategy across different material types. The left radar plot shows pass rate at $\Delta E < 10$ meV, while the right radar plot shows cost efficiency, measured as $1/$Cost Factor and averaged over passed cases within each material type.

This consistency reflects the use of shared parameter tendencies within each material category. Materials in the same broad category often have similar numerical requirements; for example, metallic systems generally require denser k-point sampling. The category reference therefore provides a reasonable scale for E_{cut} and k-point density, while the model can still adjust the parameters according to the specific material and accuracy target. To prevent answer leakage, reference-guided refinement uses only one similar material as guidance for each test case, and that reference material is excluded from the test set. Gemini 2.5 Pro achieves the highest pass rates across the three thresholds, while GPT 4o and Claude Opus 4.5 obtain slightly lower pass rates with lower cost factors.

The benefit of this strategy depends on how well the reference categories represent the target material. For materials with structures or electronic characteristics that differ substantially from those in DFTBENCH, the category reference may be less informative. These results are currently limited to the material categories represented in DFTBENCH, and the effectiveness of category-level references is not guaranteed for more complex systems or materials outside these categories. Within this benchmark, however, the results demonstrate the value of cross-material reasoning and motivate a harness that maintains historical calculation results as references for future parameter configuration.

We further evaluate advanced-parameter setup under such model type in Table 10. In comparison, the reference-guided refinement process substantially improves overall advanced-parameter accuracy for most models, indicating that the anchored iterative workflow helps preserve physically appropriate auxiliary settings in addition to improving E_{cut} and k-point selection. Claude Opus 4.5 achieves the strongest overall performance, reaching 92.3% on the total metric, while GPT 5.2, Claude Sonnet 4.5, GPT 5.1, GPT 4o, and Gemini 2.5 Flash also show clear gains over the single-pass setting. Spin polarization remains the easiest advanced parameter. By contrast, Hubbard U remains challenging for most models: Claude Opus 4.5 is the only model that reaches higher than 90%, whereas the other models frequently fail to identify correlated-electron cases consistently

Table 10: Model accuracy on advanced-parameter setup using reference-guided self-reflection model. # denotes the number of materials in DFTBENCH for which the corresponding advanced parameter is required to be set.

Model	Total (#=13)	vdW (#=5)	Hubbard U (#=4)	Spin (#=8)
GPT 5.2	76.9%	80.0%	50.0%	100.0%
GPT 5.1	69.2%	80.0%	50.0%	87.5%
GPT 4o	46.2%	100.0%	25.0%	25.0%
Gemini 2.5 Pro	69.2%	20.0%	0.0%	100%
Gemini 2.5 Flash	76.9%	40.0%	0.0%	100%
Claude Opus 4.5	92.3%	80.0%	100%	100%
Claude Sonnet 4.5	53.8%	60.0%	25.0%	87.5%

Table 11: Mean relative error (MRE, %) for VC-relax and SCF, and mean absolute error (MAE) for Band Gap and Fermi energy E_F (extracted from DOS calculations), computed over successfully finished execution results.

Model	VC-relax (Lattice)	SCF (E/atom)	Bandgap (eV)	E_F (eV)
GPT 5.2	5.7%	0.0%	0.34	1.32
GPT 5.1	7.4%	0.1%	0.45	2.34
GPT 4o	10.1%	1.2%	1.90	8.17
Gemini 2.5 Pro	5.5%	0.1%	1.02	1.45
Gemini 2.5 Flash	8.1%	0.5%	1.19	7.09
Claude Opus 4.5	8.0%	0.1%	1.84	3.01
Claude Sonnet 4.5	9.1%	0.1%	1.83	1.75

across all three accuracy levels. Overall, Table 10 shows that reference-guided self-reflection narrows the gap in advanced-parameter setup.

C.5 END-TO-END WORKFLOWS

Table 11 reports per-model results on the four end-to-end tasks (VC-relax, SCF, Bandgap, DOS), including both completion rate and accuracy on completed runs. Completion rate measures whether the agent produces a workflow that runs to termination; accuracy is computed only over completed runs, using mean relative error (MRE) for VC-relax and SCF, and mean absolute error (MAE) for Bandgap and Fermi energy. This table supports the analysis discussed in Section 5.2.

C.6 END-TO-END QUERY THROUGHPUT

Figure 8 reports end-to-end workflow throughput (queries per hour, excluding DFT execution time) for each evaluated LLM across three representative tasks. Throughput reflects the agent overhead introduced by LLM inference and tool calls, providing a practical measure of how many queries an agent can dispatch per unit time.

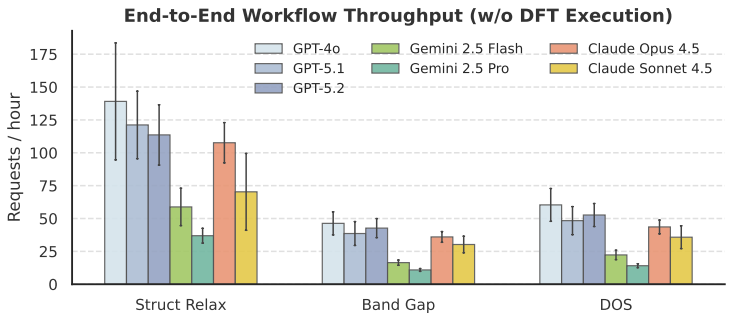

Figure 8: Comparison of TRITONDFT's workflow throughput with different LLMs across different DFT tasks.

C.7 MONETARY COST

Table 12 reports the average API cost per query across the three tasks for each evaluated LLM. Costs are computed from the official API pricing of input and output tokens consumed by TRITONDFT during the agentic workflow.

Table 12: Average cost consumption (USD) per query regarding API usage across different tasks.

Model	Struct Relax	Band Gap	DOS
GPT 5.2	0.05 ± 0.02	0.15 ± 0.04	0.13 ± 0.04
GPT 5.1	0.04 ± 0.02	0.13 ± 0.04	0.10 ± 0.03
GPT 4o	0.06 ± 0.03	0.18 ± 0.04	0.14 ± 0.03
Gemini 2.5 Pro	0.05 ± 0.03	0.13 ± 0.04	0.11 ± 0.04
Gemini 2.5 Flash	0.01 ± 0.01	0.03 ± 0.01	0.03 ± 0.01
Claude Opus 4.5	0.15 ± 0.08	0.44 ± 0.13	0.37 ± 0.11
Claude Sonnet 4.5	0.15 ± 0.06	0.34 ± 0.09	0.28 ± 0.08

C.8 Γ-POINT PHONONS AND PHONON DISPERSION

Table 13 summarizes the calculation parameters used for the Γ-point phonon calculations across five benchmark materials, comparing the TritonDFT-generated settings with the corresponding human-curated settings. Table 14 reports the resulting Γ-point phonon frequencies obtained from both workflows, enabling a direct mode-by-mode comparison between TritonDFT and the human-curated reference calculations. Phonon frequencies are highly sensitive to the relaxed crystal structure, which in turn depends on the exchange–correlation functional, convergence thresholds, k-point sampling, and the treatment of van der Waals interactions. As shown in Table 13, the TritonDFT-generated and human-curated calculations use slightly different computational settings, leading to small differences in the relaxed lattice parameters. For HfO_2, the human-curated calculation used the PBEsol functional, whereas the TritonDFT calculation used PBE. For the layered materials, the human-curated calculations included van der Waals corrections, while the TritonDFT calculations did not. These differences in computational setup and the resulting relaxed structures provide a physically reasonable explanation for the deviations observed in the Γ-point phonon frequencies reported in Table 14.

Table 13: Comparison of TritonDFT generated and human-curated calculation parameters for the benchmark phonon calculations.

TritonDFT calculation parameters								
Material	a (Å)	b (Å)	c (Å)	ecutwfc (Ry)	K-Grid	`etot_conv_thr` (a.u.)	`forc_conv_thr` (a.u.)	vdw_corr
$BaTiO_3$	3.9844	3.9844	4.178	80	$8 \times 8 \times 8$	3.7×10^{-4}	1.0×10^{-3}	none
Bi_2Se_3	10.434	10.434	10.434	80	$8 \times 8 \times 8$	1.837×10^{-4}	1.0×10^{-4}	none
FeSe	3.5549	3.5549	5.206	60	$12 \times 12 \times 8$	7.350×10^{-5}	1.0×10^{-4}	none
HfO_2	5.1437	5.1949	5.3349	90	$6 \times 6 \times 4$	4.409×10^{-6}	1.0×10^{-4}	none
$NbSe_2$	3.4353	3.4353	12.7797	70	$16 \times 16 \times 6$	3.0×10^{-4}	1.0×10^{-3}	none
Human Curated								
Material	a (Å)	b (Å)	c (Å)	ecutwfc (Ry)	K-Grid	`etot_conv_thr` (a.u.)	`forc_conv_thr` (a.u.)	vdw_corr
$BaTiO_3$	3.986	3.986	4.2043	80	$8 \times 8 \times 8$	1×10^{-9}	1×10^{-5}	none
Bi_2Se_3	10.134	10.134	10.134	80	$6 \times 6 \times 6$	1×10^{-9}	1×10^{-8}	Grimme-D2
FeSe	3.503	3.503	5.7869	90	$6 \times 6 \times 6$	1×10^{-8}	1×10^{-7}	Grimme-D2
HfO_2	5.0752	5.1541	5.2491	80	$8 \times 8 \times 6$	1×10^{-9}	1×10^{-8}	none
$NbSe_2$	3.456	3.456	12.7632	90	$8 \times 8 \times 8$	1×10^{-9}	1×10^{-8}	Grimme-D2

Table 14: Gamma-point phonon frequencies (in cm^{-1}) for five representative materials. Human-curated results are compared with TritonDFT results. TritonDFT used PBE functional for all the calculations whereas the human curated one is done with PBESol for HfO_2 and PBE for the rest.

Mode	$BaTiO_3$		Bi_2Se_3		FeSe		HfO_2		$NbSe_2$	
	Human	TritonDFT	Human	TritonDFT	Human	TritonDFT	Human	TritonDFT	Human	TritonDFT
1	-131.49	-131.47	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	-131.49	-131.47	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	42.64	39.71	123.70	144.79	115.65	106.63	26.80	21.59
5	0.00	0.00	42.64	41.89	123.70	144.79	128.63	121.87	26.80	21.86
6	149.31	147.98	71.71	66.37	200.66	235.34	130.60	122.45	51.27	40.62
7	162.33	160.43	85.61	84.34	224.02	287.55	131.10	122.56	122.92	125.39
8	162.33	160.43	85.61	87.82	224.02	293.63	144.98	141.24	122.92	125.59
9	273.34	272.92	133.39	133.19	241.38	293.63	164.14	157.47	124.06	128.27
10	295.75	292.30	133.39	135.18	303.76	345.79	180.03	172.38	124.06	128.41
11	295.75	292.30	138.55	139.02	303.76	350.96	226.81	208.44	214.11	216.92
12	369.41	362.50	138.55	142.57	325.21	350.96	240.21	232.73	223.94	224.00
13	452.84	453.99	145.62	148.34	–	–	247.61	235.21	223.94	226.32
14	452.84	453.99	163.45	168.86	–	–	252.05	243.54	226.42	226.43
15	569.14	558.60	183.17	174.71	–	–	253.05	248.17	228.16	231.22
16	–	–	–	–	–	–	310.57	295.84	228.16	231.29
17	–	–	–	–	–	–	324.30	295.87	270.47	275.51
18	–	–	–	–	–	–	329.52	312.14	294.98	300.04
19	–	–	–	–	–	–	330.93	316.14	–	–
20	–	–	–	–	–	–	342.95	325.20	–	–
21	–	–	–	–	–	–	377.88	358.73	–	–
22	–	–	–	–	–	–	385.87	358.96	–	–
23	–	–	–	–	–	–	392.89	376.17	–	–
24	–	–	–	–	–	–	403.81	390.66	–	–
25	–	–	–	–	–	–	486.06	474.54	–	–
26	–	–	–	–	–	–	493.68	484.36	–	–
27	–	–	–	–	–	–	505.25	484.67	–	–
28	–	–	–	–	–	–	505.45	494.91	–	–
29	–	–	–	–	–	–	542.77	524.77	–	–
30	–	–	–	–	–	–	561.24	544.18	–	–
31	–	–	–	–	–	–	595.99	571.91	–	–
32	–	–	–	–	–	–	621.99	601.40	–	–
33	–	–	–	–	–	–	647.86	628.41	–	–
34	–	–	–	–	–	–	657.27	635.65	–	–
35	–	–	–	–	–	–	727.09	705.27	–	–
36	–	–	–	–	–	–	764.25	744.66	–	–

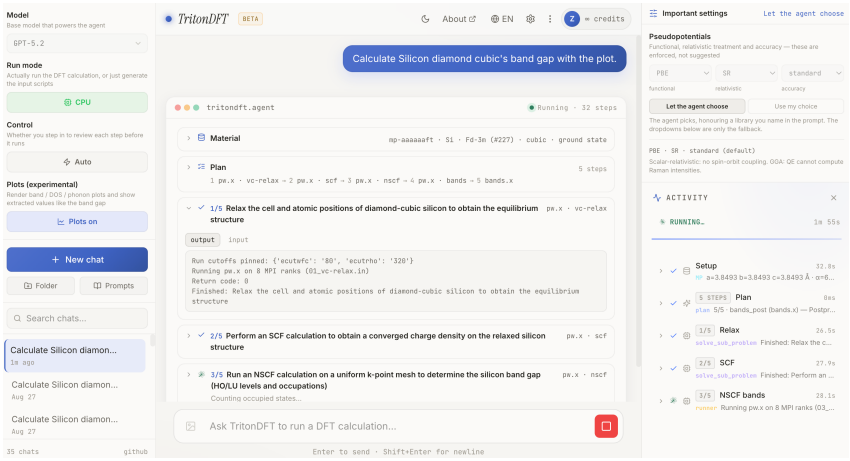

Figure 9: Web interface shows task configuration and the step-by-step DFT execution process.

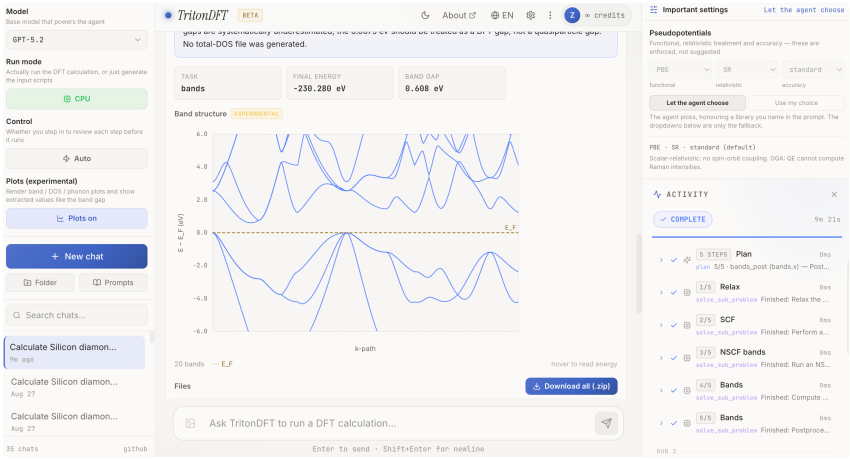

Figure 10: Web interface shows computed quantities, visualizations, and result data and files.

D USER INTERFACE

D.1 WEB INTERFACE

The web interface of TRITONDFT is shown in Fig. 9. Users can log in and submit natural-language instructions through a chat-style interface. The central panel presents the complete execution process, including the retrieved initial material structure, the decomposition of the task into subproblems, the execution progress, and the analysis of each result. From the left panel, users can select the underlying model and choose between a fully autonomous mode and an interactive mode that allows them to review intermediate steps. The right panel provides advanced settings, such as the exchange–correlation functional, relativistic treatment, and pseudopotential accuracy.

After execution, TRITONDFT reports the computed quantities together with the corresponding visualizations, as shown in Fig. 10. It also provides all generated files for download, including the input scripts and output logs. Alternatively, users can choose to generate the inputs without executing the calculation. In this mode, TRITONDFT produces the complete input files and a ready-to-run launch script, which users can copy to and execute on their own computing systems.

D.2 Advanced User Interface

The advanced user interface guides users through logging in to remote systems, inspecting workflows, selecting computational resources when necessary, submitting jobs, and monitoring their execution from a single interface. It supports both X11 forwarding and browser-based access through HTTPS or SSH port forwarding. X11 forwarding provides direct access to the graphical interface, whereas browser-based access renders the interface locally in the user's browser and generally reduces the display-communication overhead associated with remote graphical sessions. Users can therefore choose the access method best suited to their local environment, network conditions, and institutional security policies.

DFT calculations frequently fail at different stages for reasons related not to the code itself, but to the computational environment or system configuration. To accommodate such failures, the interface implements stateful workflows that track the status of each step. Users can inspect completed and active stages, identify failed calculations, and restart a workflow from a selected step without repeating the entire calculation. Whenever possible, outputs from successfully completed upstream steps are retained and reused, while the selected step and its dependent downstream stages are regenerated or re-executed. This step-level restart capability is particularly valuable for long-running HPC calculations, in which repeating previously completed stages can consume substantial computational resources.

The advanced user interface provides human oversight throughout both workflow preparation and execution. The interaction is divided into two main stages: (1) scientific assessment and plan approval before calculations are submitted, and (2) execution monitoring and result inspection after submission.

D.3 Scientific Assessment and Plan Approval

Figure 11 shows the scientific assessment and plan-review interface. Before any calculation is submitted, TritonDFT presents its interpretation of the user's request, including the requested observables, material configuration, and proposed computational strategy. The generated workflow is decomposed into individual calculation and post-processing steps, with the purpose, required inputs, expected outputs, and computational method identified for each step.

The interface provides separate views of the complete plan, workflow graph, source structure, available computational resources, catch mistakes through validation, and input files. Users can therefore inspect both the scientific reasoning and the specific simulation parameters before allocating computational resources. Input files for individual workflow stages, such as structural relaxation, self-consistent-field calculations, band-structure calculations, and density-of-states calculations, can also be reviewed and edited directly.

After reviewing the proposed workflow, the user may validate the plan, request revisions, modify the generated inputs, or approve the workflow for execution. Cluster submission remains locked until the user explicitly selects *Approve & Run*. This design introduces a human-in-the-loop checkpoint that helps identify inappropriate physical assumptions or numerical settings before expensive calculations are launched.

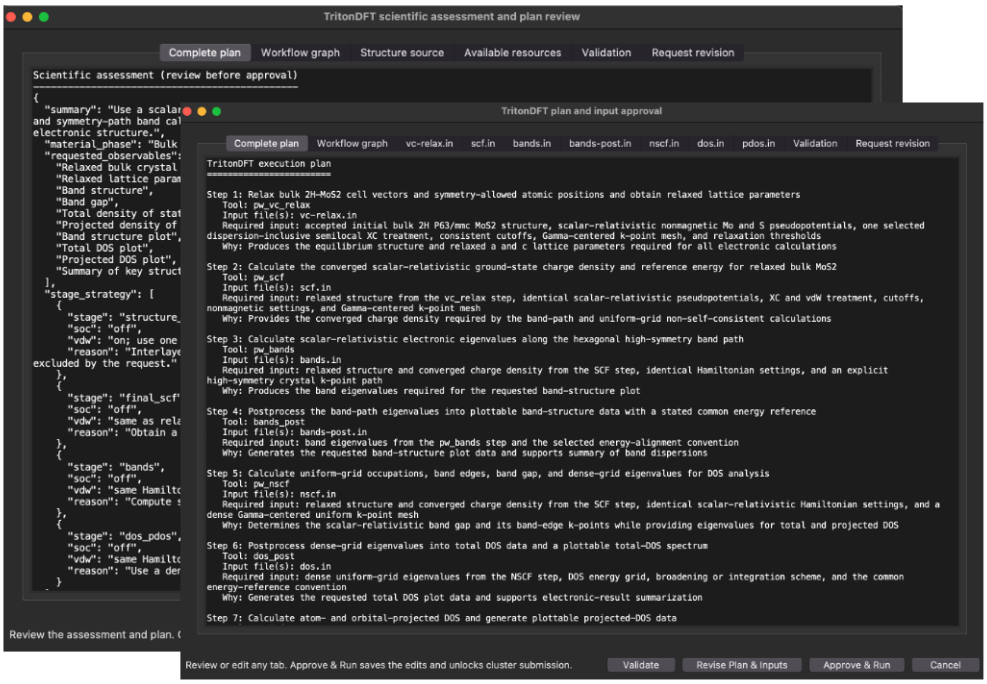

Figure 11: Advanced user interface for scientific assessment and workflow approval. The assessment window summarizes TritonDFT's interpretation of the request and proposed computational strategy, while the plan-and-input approval window allows users to inspect, validate, revise, and approve the individual workflow stages and their corresponding input files before cluster submission.

D.4 Execution Monitoring and Result Inspection

After approval and submission, the interface switches to the execution and results view shown in Figure 12. The *Execution status* tab presents the workflow as an ordered collection of tasks and reports the branch, current state, execution attempt, and associated scheduler job identifier for each task. This information allows users to determine which calculations have completed, which are currently running, and which require further attention.

The workflow state is retained at the level of individual tasks. If a task fails, the user can revise its inputs and restart the workflow from the appropriate stage. Successfully completed upstream tasks and their outputs are preserved whenever they remain compatible with the revised calculation. Dependent downstream tasks are then regenerated or executed again as needed. This approach avoids unnecessarily repeating completed calculations and is particularly valuable for long-running workflows on high-performance computing systems.

The remaining tabs provide direct access to validation information, generated input files, relaxed structures, electronic band structures, density-of-states data, and summarized results. Structural information can be inspected through lattice parameters, atomic coordinates, an embedded three-dimensional visualization, or files exported in commonly used formats. Electronic-structure results are presented as interactive band-structure and density-of-states plots, with controls for energy referencing, plot ranges, and visualization options. Together, these capabilities provide a unified environment for tracking workflow execution and examining the resulting scientific data.



Figure 12: Advanced user interface for workflow monitoring and result inspection. The execution-status table records the state, execution attempt, and scheduler job identifier of each workflow task. Additional views provide access to the relaxed crystal structure, structural parameters, band structure, density of states, validation information, and generated input and result files.

D.5 Support for VASP

Beside running Quantum ESPRESSO, TritonDFT also supports generating inputs and running VASP (Hafner, 2008) calculations on computing clusters. Because VASP is proprietary software, users must have a valid VASP license and authorized access to a VASP installation and pseudopotential library on the cluster to execute calculations. Anyone can use TritonDFT to generate VASP input files without a license. To protect licensed data, TritonDFT does not read, store, or transfer POTCAR contents. Instead, it generates a shell script that assembles the required POTCAR file directly on the compute cluster from the user's authorized VASP POTCAR library immediately before the calculation runs.

E Details about TritonDFT Framework Design

E.1 QE-aware Multi-step Planner

The planner currently incorporates 13 calculation tools or workflow modules. The prompt is designed as follows:

You are a strict planning assistant for Quantum ESPRESSO.
Output requirements:
- Decompose the user query into 1..N subproblems.
- Each subproblem must be wrapped as <subproblem1>...</subproblem1>, <subproblem2>...</subproblem2>, etc. (in order).
- Each subproblem must contain four fields:
Problem: What to calculate, Tool: Tool to use, Required input: Required input parameters (Do not give any concrete parameter value here, just describe what is needed). These fields MUST appear on separate lines, each separated by a newline; otherwise, the output is considered incorrect.
- Keep each subproblem short (2–3 lines).
- Do not output anything outside <subproblem> blocks.
Core rules:
Allowed tools: pw_scf, pw_nscf, pw_relax, pw_vc_relax, pw_bands, bands_post, dos_post, projwfc_post, pp_post, q2r_post, matdyn_post, pw_phonon_gamma, elastic_post.
In-Context Example 1 (band structure with given lattice constant)
Query: Calculate the band structure of silicon in the diamond structure (a0 = 5.43 Å).
<subproblem1>
Problem: Do an SCF calculation to converge charge density
Tool: pw_scf
Required input: diamond Si structure
</subproblem1>
<subproblem2>
Problem: Perform NSCF calculation along the high-symmetry path
Tool: pw_nscf
Required input: same structure, SCF charge density
</subproblem2>
<subproblem3>
Problem: Post-process bands to obtain band structure
Tool: bands_post
Required input: NSCF results
</subproblem3>
In-Context Example 2 (equilibrium lattice constant unknown)
Query: Calculate the equilibrium lattice constant of Na in the BCC structure.
<subproblem1>
Problem: Find equilibrium lattice constant by relaxing the cell volume and atomic positions
Tool: pw_vc_relax
Required input: BCC Na structure
Output: equilibrium lattice constant (Å)
</subproblem1>
Now handle this query:
Query: {question}
- Do NOT include reasoning, explanations, or justification.
- The output must ONLY be <subproblemN>...</subproblemN> blocks, nothing else.

TRITONDFT will parse these <subproblem> blocks and execute each subproblem accordingly. Further extension of additional executables can be incorporated by adding corresponding options explicitly in the prompt, along with executable descriptions and parameter specifications, as described in Section E.2.

E.2 PARETO-AWARE PARAMETER GUESS AND SCRIPT GENERATION

Single-pass estimation. For each subproblem, a parameter-guessing stage is first performed to estimate the key input parameters required for execution. By explicitly guiding the model to generate Pareto-aware parameter estimates, TRITONDFT achieves a principled trade-off between numerical accuracy and computational efficiency. Additionally, TRITONDFT aggregates the execution results of previous subproblems together with relevant internal memory, forming the previous_memory field to enable enhanced context-aware parameter estimation and decision making. The parameter guessing prompt is designed as follows:

You are a computational materials scientist solving a DFT subproblem. Now you are asked to propose ONLY the minimal necessary input parameter guesses for this subproblem.
Instructions
1) Propose only the parameters that are directly required or explicitly important for this query.
2) Make reasonable guesses based on existing data. Avoid performing convergence tests; instead, provide an educated estimate.
3) When guessing parameters, aim for a Pareto-efficient trade-off: choose values that are accurate and numerically stable (i.e., likely to converge), while keeping computational cost (time and memory) reasonably low.
Output Schema

```
{
  "material": "<element/compound>",
  "structure": {
    "prototype": "<fcc|bcc|rocksalt|diamond|...>"
  },
  "parameter_guesses": {
    "<only the parameters directly relevant to the query>": "<value>"
  }
}
```

The whole user query is:
{query}
{previous_memory}
Current Problem to Solve:
{subproblem}
Please respond strictly in the required JSON schema, without any additional explanation.

Once the parameter guesses are determined, TRITONDFT further generates executable-specific scripts for each subproblem. We design a dedicated set of executable requirements, provided in the `tool_requirements` field, where the script format is carefully specified by domain experts. For example, for `pw.x`, we require the model to explicitly generate the standard namelists such as `&control`, `&system`, `&electrons`, `&ions`, and `&cell`. This modular and extensible prompt integration enables TRITONDFT to support additional executables and functionalities with minimal changes.

You are a computational materials scientist.
Given the parameter JSON below, GENERATE ONLY input files in Quantum ESPRESSO. Wrap ALL generated inputs inside ONE `<scripts>...  </scripts>` block; each file inside a `<script>...  </script>` tag.
{tool_requirements}
{previous_memory}
{previous_run}
Only apply targeted corrections to the script based on the errors above; do not change any other guessed parameters unnecessarily.
The whole user query is:
{query}.
Input (parameter JSON)
{params_json}
{query_info}
Current Problem to Solve:
{subproblem}.
Output Schema
`<scripts>\n <script>...one input...  </script>\n <script>...next...  </script>\n ...</scripts>`
Pitfall
Do NOT regenerate or reinterpret the entire query. Only generate new or corrected scripts corresponding to the specific subproblem(s) currently being handled. For example, if the overall query involves multiple tasks on the same system (e.g., `vc-relax`, `scf`, and `bandgap`), generate scripts **only** for the current subproblem (e.g., `vc-relax`). Future subproblems like `scf` or `bandgap` will be generated later.
Please provide your output containing only the text wrapped in `<scripts>` and `<script>` tags.
You MUST output raw plain text. Do NOT format or render anything. Output must be copy-paste ready for a Unix shell or input file.
Do NOT escape or encode any characters. In particular, NEVER output "&".
Use literal Quantum ESPRESSO namelists like "&control", "&system", "&electrons", "&ions", "&cell".

E.3 AUTOMATIC PARALLELIZATION

For the generated executable script, TRITONDFT further performs automatic parallelization by configuring hybrid MPI/OpenMP parameters for `mpirun`. The computational scale and workload characteristics of the target program are estimated from the `probe_output` field, which is obtained by running a default `mpirun` configuration for two minutes and extracting profiling information from the output log. Based on this runtime probe, the model selects appropriate parallel parameters to balance communication overhead and computational scalability. Meanwhile, the hardware information is integrated into the `hardware_description` field in natural language, enabling the model to infer physical core counts and resource constraints. The prompt is designed as follows.

You are an expert in Quantum ESPRESSO (QE) parallelization for HPC systems. Your goal is to choose the optimal Hybrid MPI/OpenMP configuration (`OMP_NUM_THREADS`, `-nk`, `-nb`, `-ntg`).
PARAMETER RULES
1. **Resource Limits**:
- Identify Physical Cores from the description. IGNORE logical/hyper-threads.
- Constraint: (MPI_RANKS) * (OMP_NUM_THREADS) $\leq$ Total Physical Cores.
- Under-subscription: It is valid to leave some cores idle to satisfy divisibility requirements.
2. **Parallel Flags**:
- `-nk` (Pools): Parallelizes over k-points. Constraint: MPI_RANKS and k-points must be divisible by nk.
- `-nb` (Band Groups): Parallelizes linear algebra (diagonalization). Usage: Typically for systems with many electronic bands.
- `-ntg` (Task Groups): Parallelizes 3D FFT grids. Constraint: MPI_RANKS must be divisible by ntg. Usage: Helps distribute large FFT grids across many ranks.
3. **General Optimization Strategy (Reasoning Required)**:
- Assess System Scale: Look at `nat` (number of atoms) and k-points in the input.
- For Small Systems (few atoms), MPI communication overhead often outweighs the benefit of complex parallelization logic. Excessive splitting (high `-nb`, `-ntg` or OpenMP) can degrade performance.
- For Large Systems (many atoms), calculation and memory are the bottlenecks. Advanced flags (`-ntg`, `-nb`) and Hybrid MPI/OpenMP become essential to scale.
- Decision Process: Balance the need for parallelism against the cost of communication.
Inputs
1) QE input script (verbatim): {input_script}
2) Hardware resources (natural language description): {hardware_description}
3) Probe output (from a default run): {probe_output}
 # Output Format (STRICT)
Command: export OMP_NUM_THREADS=<int>; mpirun --allow-run-as-root -np <int> {exec_path} -nk <int> -nb <int> -ntg <int> -in {input_filename} | tee {output_filename}