

Evaluating LLM-based AI agents integrated with materials synthesis tools: the case of atomic layer deposition

Angel Yanguas-Gil^{1*}

¹Applied Materials Division, Argonne National Laboratory, 9700 S Cass Ave, Lemont, 60062, Illinois, USA.

Contributing authors: ayg@anl.gov;

Abstract

This review provides an overview of the different strategies that can be used to evaluate the performance of AI models and agents based on large language models (LLMs) for materials synthesis. After providing a brief overview of the key technologies behind the current generation of AI agents based on LLMs, we summarize the different approaches to evaluating these models in the context of materials science and in particular on materials synthesis, with a specific emphasis on scenarios in which the models are directly integrated with experimental tools. We discuss evaluation strategies spanning knowledge and reasoning benchmarks, tool-use benchmarks, and closed loop benchmarks involving the interaction with experimental systems or realistic virtual tools. We use atomic layer deposition (ALD) as a case study, emphasizing how existing approaches in the literature both build from general approaches used beyond materials science and can be generalized to other materials synthesis techniques. Finally, we provide a practical evaluation framework to evaluate LLMs in the context of materials synthesis

Keywords: large language models, materials synthesis, artificial intelligence, atomic layer deposition

1 Introduction

In recent years, there has been growing interest in integrating generative AI models and agents with materials synthesis tools to realize autonomous platforms for materials

discovery and process optimization[1–4]. The underlying vision is that agents powered by the most recent generation of large language models (LLMs) could orchestrate complex research workflows[5] involving multiple synthesis and characterization tools, simulations and prior data (Fig. 1), much as they already excel in tasks such as code generation. Unlike conventional machine learning (ML) models designed for specific tasks, LLM-based agents can potentially address a much broader set of tasks, interpreting natural-language instructions, accessing and searching for data, and managing multi-step workflows involving both simulations and experiments.

Current efforts to integrate AI with experiments fall into two broad categories: the development of end-to-end complex autonomous platforms mentioned above and the augmentation of existing instruments with AI agents. Examples of the latter include characterization techniques such as atomic force microscopy and thin-film growth techniques such as pulsed laser deposition and atomic layer deposition[6–8]. While the ability to augment or retrofit existing equipment and tools allows us to tap into the wealth of experimental tools in materials science labs and consequently have a broad impact, it has its own unique challenges, including how to design effective interfaces to experimental equipment that were not originally designed for automation, and how to make these interfaces compatible with various types of AI/ML algorithms, which can range from deterministic space searches to the latest generation of LLM-enabled agents.

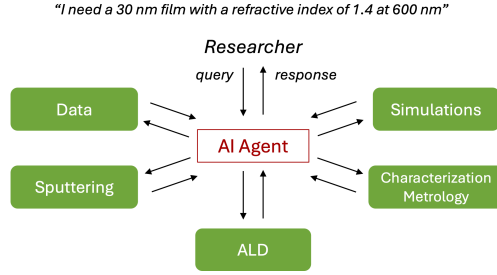


Fig. 1 AI agents powered by LLMs can potentially orchestrate multiple complex research workflows involving the interaction with multiple materials synthesis, characterization, and simulation tools as well as databases and prior data

In this review, we focus on the problem of how to quantify the performance of AI agents integrated with experimental tools. Evaluating LLM-based agents is critical to understand how well they will perform in real research tasks, identify existing limitations, and quantify the actual value of AI integration. The importance of benchmarks and evaluation is bound to increase as LLMs and AI agents progress to higher technology readiness levels and are consequently reused for multiple projects and applications. This is something that we are observing today, with commercial and open-weight LLMs being actively explored for different research tasks[9]. In the context of materials synthesis, the cost and time associated with running experiments and the risk of inadvertently damaging valuable equipment make quantifying the performance of AI models particularly relevant. The stakes are even higher in manufacturing settings

such as semiconductor processing, where the impact of cross-contamination or simply low yields can propagate downstream over multiple processing steps before being detected.

Developing evaluation strategies and benchmarks to help us understand the capabilities and shortcomings of AI agents and models is therefore crucial to pre-empt any problems and derisk their integration with experiments. This is a responsibility that is shared between AI researchers and domain experts: AI researchers make extensive use of benchmarks to quantify model performance during the training or fine-tuning of large language models[10], but most of the benchmarks are meant to be run in environments that are isolated from experimental labs. Materials scientists can play a critical role in the development of targeted benchmarks to ensure that the models can be evaluated in scenarios that are relevant for downstream uses in materials research.

This review has therefore two goals: to provide an overview of how experimental tools are integrated with LLM-based agents, and to summarize strategies used for evaluating the performance of these agents. This work approaches the subject from a materials science perspective. The emphasis is not so much on the training of new models as on how we can evaluate the performance of these models and agents. In Section 2 we provide a quick overview of LLM-based technologies, how they have been leveraged for materials science and related scientific applications, and the main evaluation strategies used in the literature. In Section 3, we review the evaluation strategies used to understand how well LLMs perform for materials science, and for materials synthesis in particular. We focus on atomic layer deposition as a model system to explore how some of the main strategies used for LLM evaluation are particularized for materials synthesis. Finally, in Section 4 we conclude with some recommendations, including a practical evaluation framework for materials synthesis.

2 LLM-based agents and their integration with experiments

2.1 Overview of LLMs, multimodal LLMs, and AI agents

Large language models are a class of deep learning models trained on large corpora of text to predict the probability distribution over the next token in a sequence. A key breakthrough was the introduction of the transformer architecture, which provided a self-attention mechanism capable of representing long-range correlations present in complex texts[11]. The generative pre-trained transformer (GPT) family is arguably the best-known example[12].

A defining characteristic of LLMs is the probabilistic nature of their outputs: given a prompt, the model samples from a learned probability distribution over possible continuations, producing outputs that are stochastic even for identical inputs. This stochasticity can have important implications for applications in science and in materials synthesis in particular, where reproducibility and consistency are critically important as these technologies mature from research artifacts to real-world use, particularly when integrated with experimental and simulation capabilities.

Applications built on top of LLMs typically rely on an orchestration layer, or harness: this is a software layer that manages the interaction between the model and the user. For instance, the harness is what gives the perception of having a conversation with a model. After each new user input, the harness sends the model the conversation history up to that point, therefore priming the LLM to provide a response built on top of the conversation history.

Two key capabilities developed using a combination of model training and harnesses are multi-step reasoning strategies and tool use. In the context of multi-step reasoning, strategies such as chain-of-thought[13] and self-consistency sampling[14] have powered a new generation of reasoning LLMs that are capable of solving complex logical problems. Examples of such reasoning models include OpenAI’s o3 as well as open-weight models such as DeepSeek-R1[15] and Qwen[16]. These techniques, which also require their own specific training strategies, rely on breaking down queries into intermediate reasoning steps that are addressed by LLMs before a final answer is returned to the user.

In tool use, or function calling, LLMs are trained to generate structured output indicating which tool to use when their prompt includes a structured description of available resources [17, 18] (for instance a Python interpreter or a database query interface). The harness then interprets the model output, runs any tool as needed, and generates a new prompt that is passed to the model with the response. This model-tool-updated prompt cycle can run many times before the information is sent back to the user.

One limitation of conventional LLMs is that they are limited to text. Multimodal LLMs (MLLMs) are designed to overcome this limitation, and they are capable of ingesting one or more additional modalities (image, video, audio, etc.) besides text. MLLMs contain other pre-trained components to encode the main features of the additional input modalities. They also have a cross-modal connector that bridges the output of the different encoders with the language model. For instance, the LLaVA architecture uses a multilayer network to project features encoded by a pre-trained vision encoder into the language embedding space[19]. The model is trained using a visual instruction-tuning methodology[19].

AI agents based on LLMs leverage all these individual tools to both reason about complex, multi-step tasks and interact with tools and their environment[20]. The concept of an *agent* predates the rise of LLMs [21], and it broadly refers to systems that interact with their environment to achieve concrete goals. A key difference of LLM-based agents is that they do not require task-specific training or programming.

The different capabilities of models and agents matter for evaluation because they require different types of benchmarks.

2.2 Integrating LLM-based agents with experimental tools

At a high level, integrating LLM-based agents with experimental tools typically involve the generation of API calls or other structured instructions. In this section, we will exemplify this through three representative examples in the literature:

A first representative example is Coscientist, one of the earliest frameworks to demonstrate an LLM-powered agent capable of autonomously designing and executing

chemical experiments[5]. In its original design, Coscientist comprises an LLM-planner that has access to four different commands: GOOGLE, PYTHON, DOCUMENTATION, and EXPERIMENT. A simple harness uses the LLM to generate calls to any of the four modules as needed or until a maximum number of iterations is reached. Interpretation and execution are handled by an automation layer that does not rely on LLMs. The results of these experiments are then sent back to the harness and used to generate subsequent calls.

A second example is the integration of an atomic force microscopy (AFM) tool[6]. Mandal et al. implement two agents: an AFM Handler Agent that is used for experimental control and a Data Handler Agent for analysis. The interface with the experimental system is done through a Python-based API, enabling direct control of the experimental system. The AFM Handler Agent has control of a code-execution tool, which directly runs Python code generated by the LLM. This allows the generation of complex sequences of instructions through Python scripts.

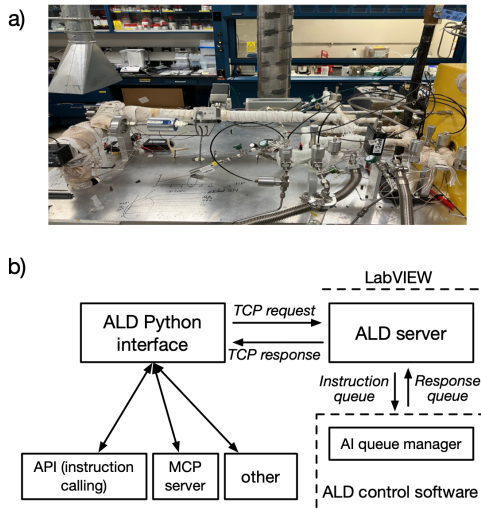


Fig. 2 Example of LLM integration with an atomic layer deposition tool: (a) Picture of the cross-flow ALD reactor, including in-situ ellipsometer; (b) Scheme of the interface design, where the LabVIEW software controlling the ALD tool is accessed by a Python API via TCP sockets. This API allows multiple types of integration with both conventional ML algorithms and LLMs, including through MCP servers or direct tool use. Figure adapted from Ref. [8] with permission.

The third example is the integration of an LLM-based agent with an atomic layer deposition tool (Fig. 2)[8]. In this case, a translation layer sits between the outward-facing Python API, which defines a small set of operations, and the LabVIEW backend, which is used to operate the tool. This Python API is designed to allow for three types of integration with LLMs: through tool use, the Model Context Protocol (MCP), and a custom harness similar to the execution loop of the Coscientist architecture from Boiko et al.[5]. In contrast to the AFM case, in the specific example reported in Ref.

[8], the LLM outputs text in structured format, in this case JSON, that is then parsed and translated into a specific tool call.

While there are other examples in the literature, these three cases showcase one of the most common design patterns to integrate LLMs with experimental tools:

1. LLMs generate tool calls in some sort of structured format, such as JSON, Python or an application-specific schema.
2. A harness translates model outputs into executable actions, often with an iterative control loop. The exact structure of the agent can change from case to case and involve multiple modules and even multiple agents each specialized in a part of the project. This harness may include non-LLM modules such as parsers, safety checks, schedulers, or conventional ML components.
3. A query-response-action loop is common, especially when the agent must incorporate feedback from tools, databases, or experiments before deciding on the next step.

Figure 3 shows the high-level design of a simple LLM agent interacting with experimental tools that incorporates these key ideas.

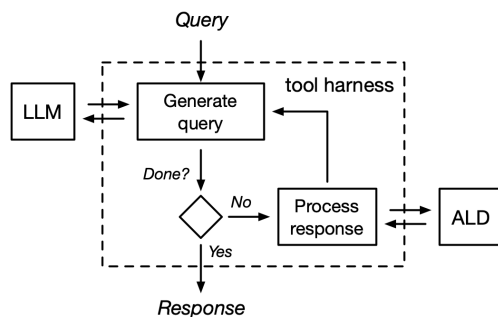


Fig. 3 A high-level structure of a simple LLM agent interacting with an experimental tool: the LLM-based agent comprises a harness that implements a query-response-action loop until the LLM considers that the task is done or a maximum number of iterations has taken place. This harness is responsible for translating the model response into the instructions that are used to communicate with various experimental tools and resources available to the agent. Examples such as Coscientist[5] or agents used for ALD process optimization share a similar top-level architecture.

3 Evaluating LLMs and LLM-based agents in materials science

In this section, we summarize the main evaluation strategies relevant to materials science and materials synthesis. Section 3.1 provides an overview of evaluation strategies for LLMs in scientific applications and materials science. Section 3.2 then focuses on evaluation for materials synthesis through the case study of atomic layer deposition.

3.1 General evaluation strategies and benchmarks in materials science

3.1.1 Evaluating knowledge, reasoning, and text generation

As shown in the prior section, regardless of whether the output is produced directly by the LLM or through a harness with some sort of internal logic, LLMs are designed to generate output text from an input prompt. At the highest level, both general-purpose and materials-science-specific LLM benchmarks comprise a collection of questions or inputs together with a scoring method for individual responses and the overall score for the benchmark. These benchmarks can be run either directly calling the model or through an evaluation harness that standardizes the evaluation process. Most of the relevant examples are designed to be run by directly calling the model via an API without the need to use a harness.

To evaluate knowledge and reasoning capabilities, LLM benchmarks typically use a few standard approaches: multiple choice questions (MCQ) are a common approach, being extensively used as part of general purpose and materials science and scientific benchmarks. A useful aspect of MCQs is that evaluation can be carried out either by directly computing the probability for each answer option after inputting the question, or by evaluating an LLM-as-judge approach, in which an open-ended response is evaluated by an LLM that determines whether it corresponds to the right answer. This ease of evaluation and of encoding the right response makes MCQs one of the most common approaches for early benchmarks. One such example is GPQA[22], whose main set comprises 448 expert-level questions in biology, physics, and chemistry. These questions were designed by domain experts to require a significant amount of domain-specific reasoning and to resist retrieval through internet searches. Within the context of materials science, MCQs are part of benchmarks such as MaScQA[23], which includes more than 350 MCQs comprising 14 subdomains within materials science, from materials manufacturing to electrical properties and mechanical behavior. Finally, the structured nature of MCQs is ideally suited to explore automatic question generation[24].

Questions requiring open-ended responses are used when probing the ability to generate numerical values, for instance to evaluate the model’s ability to retrieve physical properties of materials or to provide the solutions to numerical problems. They are also used to answer questions requiring the generation of text in specific format, such as chemical formulas or SMILES or even to generate code. They are evaluated using a combination of rule-based and LLM-based evaluations. For instance, MatSciBench is a benchmark comprising 1340 questions curated from materials science textbooks comprising 1236 numerical questions and 104 formula questions[25]. The authors evaluate the answers using primarily a rule-based approach based on parsing the output using regular expressions based on that used to evaluate LLM’s answers to math questions. Another example is ChemBench, which integrates MCQs with open response questions in the field of chemistry[26]. Benchmarks requiring free-text generative outputs typically use either LLM-as-a-judge scoring approaches or human Likert-scale ratings. One example of the second case is ALDBench, which was developed to evaluate the

ability of LLMs to reason about atomic layer deposition[27]. This will be described in detail in Section 3.2.2.

Finally, recent benchmarks have started to incorporate modalities beyond texts to evaluate the potential of multimodal models. MaCBench is a benchmark that focuses on evaluating the model performance on research tasks[28]. Each entry in MaCBench comprises pairs of images and text-based questions. They employ a rule-based parser that uses regex-based functions to handle both MCQs and numerical values. As in the case of MatSciBench, an LLM extractor is used to handle cases where the parser fails. Circi et al evaluated the ability of LLMs to understand tables in materials science, creating a dataset of 2,512 entries and comparing the performance of using image vs text (OCR and structured format) inputs on data extraction tasks[29].

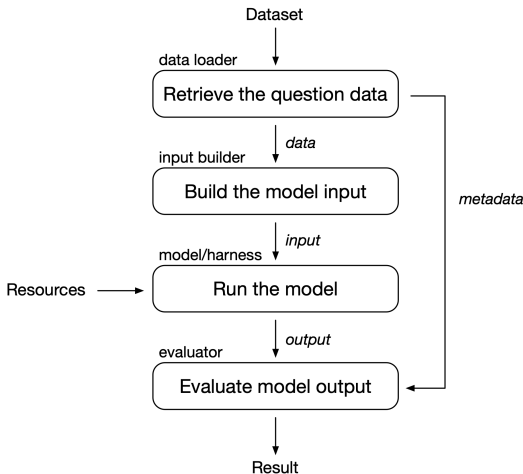


Fig. 4 Basic LLM evaluation pipeline: for each question in the benchmark, a *data loader* takes data and the *input builder* transform it into a input that is passed to the model. The *data loader* can also extract any relevant metadata, such the difficulty or the topic of the question. The output of the model is then passed through an *evaluator* to extract the results and compute the model score.

Despite their diversity, these benchmarks share a common structure: 1) a *dataset* containing the information required to build each input to the model; 2) a *builder* function that transforms each entry into a model input, often a prompt and, for multimodal benchmarks, additional modalities such as images or spectra; 3) an *evaluator* function that parses the model output and compute a score. The resulting evaluation pipeline is summarized in Figure 4. The data for each question may include relevant metadata, such as the difficulty, subcategory or additional context for the evaluator. In some cases, additional resources are made available to the model, such as the ability to access the open internet or a collection of tools. These components can be implemented in a few lines of Python code, shared as their own code repositories, or be part of complex testing harnesses that leverage online repositories to download models and the data required to run a specific benchmark[30].

3.1.2 Evaluating tool use and interactive agent behavior

As tool use has become an increasingly important application of LLMs, a number of benchmarks have emerged that evaluate the ability of AI agents and models to interact with external tools. Some of these general purpose benchmarks provide conceptual models that are useful in the context of materials science. For instance, in the work from Xu et al introducing the ToolBench benchmark[31], the authors provide a simple conceptual model for tool use comprising two different scenarios (Figure 5) in the *single step manipulation scenario*, a model or agent has to generate one or more calls to external tools accomplish a specific goal. Models can be evaluated using the general evaluation pipeline described in Section 3.1.1 (Figure 4). In contrast, in the *multistep manipulation scenario*, the model or agent must interact iteratively with an environment, generating tool calls and incorporating the environment’s feedback until it reaches an exit step.

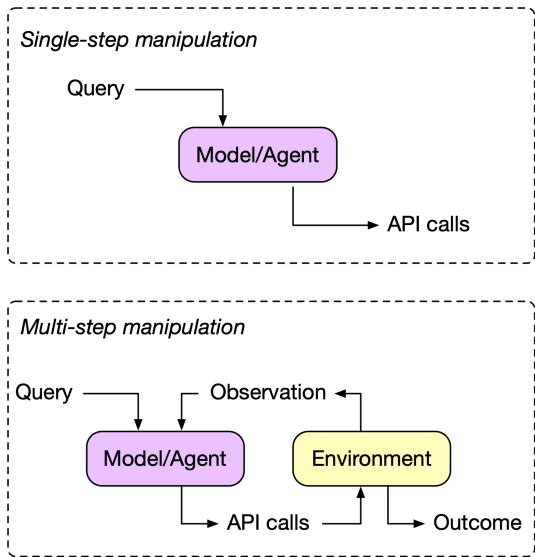


Fig. 5 Tool manipulation examples fall into two broad scenarios: in the single step manipulation case, the model or agent responds with a sequence of API calls; in the multi-step manipulation scenario, the agent interacts with the environment, receiving feedback to a sequence of tool calls until a desired outcome takes place.

This framework can be directly applied to the integration of AI with scientific tools. From a materials science perspective, this has a series of advantages. First, the ToolBench framework treats the model or AI agent as a black box, being broadly defined as a system that takes a text prompt as input and executes a series of actions. Second, communication takes place through an API or interface. This interface is the natural place where the experiment interfaces with the AI model. Consequently, the design of benchmarks that are specific to a given tool requires thinking about the interface.

The single step manipulation scenario computes the accuracy by comparing the function calls with the desired function call. This means that it is possible to evaluate model without having to physically connect them to the instruments. The benchmark can therefore be the first step in understanding the feasibility of using AI to drive a specific tool. This provides a practical way to iterate on interface design and improve the performance of the model or agent before connecting it to a real piece of equipment.

The evaluation of the multistep generation scenario requires the interaction with a tool or an environment[20, 32]. However, as we show in Section 3.2.4, in some cases it is possible to replace the real tool with a virtual tool that is an approximation of a real system. Even if that system is not tied to a specific tool, since the main use case of generative AI is its applicability to a wide range of conditions, a model’s ability to solve challenges even in an idealized case can provide insights on the expected performance when connected to a real system.

3.1.3 Benchmarks for materials synthesis

While some general-purpose and scientific benchmarks do contain materials synthesis queries, materials synthesis is not covered in depth in chemistry and materials science benchmarks. For instance, MaScQA includes 91 questions on materials manufacturing and 35 questions on materials processing, focused primarily on metallurgy[23]. MatSciBench focuses on materials processing also in the context of metallurgy[25]. Overall, most materials science-specific benchmarks do not address materials synthesis in an explicit way (see, for instance, Ref. [33] for a perspective on LLMs for materials science).

Likewise, the number of benchmarks that address specific materials synthesis techniques is also still very limited. In addition to the ALD case addressed in Section 3.2 [8, 27, 34], there are only a few examples in the literature. For instance, Cho et al developed a twenty open-response question benchmark to evaluate the performance of retrieval augmented generation (RAG) systems using the specific case of graphene synthesis. These questions were created by domain experts and evaluated using four different methods, a metric of similarity with the ground truth based on BERTscore, LLM and human judges, and the automated RAG evaluation framework RAGAS[35].

In the broader context of inorganic chemistry, Kim et al explored LLM’s ability to predict the synthesizability of chemical compounds[36]. Out of 393,053 unique compositions contained in the Open Quantum Materials Database and the Materials Project, they defined as synthesizable the subset with references in the Inorganic Crystal Structure Database (40,817). The structure was similar to the MCQ approach, with the models being asked to generate if the compound was synthesizable or not based on the chemical formula. They also developed a benchmark that asked the model to identify precursors for materials synthesis based on 11,923 unique reactions and 311 precursors based on the the synthesis dataset from Kononova et al[37]. In this case, each question was open ended and the response was parsed against a predefined template and the correct list of precursors.

Overall, the current state of the literature points out to significant gaps that need to be addressed if we want to fully understand the capabilities and limitations of LLMs and LLM-based agents in materials synthesis.

3.2 Atomic layer deposition as a case study

The benchmarks described in Section 3.1 have helped illuminate the potential and limitations of existing LLMs in the context of material science and the approaches to evaluate the performance of LLMs and agents. While not trying to be exhaustive, many of the benchmarks mentioned in Section 3.1 provide a great starting point to understand the potential of LLMs. They also play a key role in AI research, assisting in the development of a new generation of models specialized for materials science research.

However, there is still an important gap between the general capabilities evaluated by general purpose benchmarks such as MatSciBench or ChemBench, and the level of granularity required to evaluate models meant to be deployed along with materials synthesis tools to assist in the research process. This has led to the development of more specialized benchmarks that target specific subdomains and tasks.

In this section we use atomic layer deposition (ALD) as a model system to understand how to apply different evaluation strategies to materials synthesis. The examples in this section adapt the general evaluation ideas introduced in Sections 3.1.1 and 3.1.2 and particularize them for this thin film growth technique. They provide a great case study that can inform how to approach evaluation of models and agents in the context of materials synthesis.

3.2.1 Atomic layer deposition

Atomic layer deposition is a thin film growth technique based on the self-limited reactions of a precursor and a co-reactant with surfaces. In contrast to chemical vapor deposition, which is predominantly a continuous process, in ALD exposures to precursor and co-reactant are separated by purge times to prevent their simultaneous presence in the reactor. An ALD cycle involving one precursor and one co-reactant is defined by four time intervals: t_1, t_2, t_3, t_4 , where t_1 and t_3 are the precursor and co-reactant dose times, respectively, and t_2 and t_4 are the precursor and co-reactant purge times. A repetition of multiple cycles produces an ALD growth.

The optimization of an ALD process typically involves identifying the shortest dose and purge times that still achieve saturated growth per cycle. If the dose times are too small, the growth per cycle is lower than the saturated growth per cycle. If the dose times are too long, precursor is wasted and the process takes longer. An ALD reactor frequently has multiple precursor and co-reactant channels. Executing a growth therefore involves picking a set of precursors and co-reactant, setting conditions such as the reactor temperature, and the right dose and purge times. ALD can be extended to more complex processes in which different precursors are alternated to grow ternary, alloyed, or doped materials. It may also use surface functionalization steps to achieve growth selectivity or modulate the saturated growth per cycle. There are a number of reviews in the literature that explore the range of ALD processes and applications in depth[38].

ALD is a particularly useful model system for evaluating LLMs in materials synthesis. First, its high degree of automation makes it a natural candidate for integration

with both conventional ML methods and LLM-based agents. Second, it allows evaluation at multiple levels of granularity, from identifying relevant chemistries and selecting available precursors in a tool, to carrying out closed-loop process optimization.

In the following sections, we will introduce three different methodologies for evaluating LLM-based models and agents in the context of atomic layer deposition: an open response benchmark meant to be evaluated by human experts, a benchmark for the use of ALD experimental tool, and a benchmark for ALD process optimization based on reasoning LLMs.

3.2.2 ALDBench: an expert-reviewed open-response benchmark

ALDBench provides an example of an open response benchmark developed specifically to understand LLM’s capabilities in the context of materials synthesis[27]. ALDBench comprised 70 questions grouped into four categories: "how to grow", "specific questions about ALD processes", "general ALD knowledge", and "applications". LLM responses were graded by seven human experts across four criteria: Quality, Specificity, Relevance, and Accuracy, using a 1 to 5 Likert scale. Questions were also graded in terms of their Difficulty (1-Easy, early graduate, 5-Hard, top expert) and Specificity (1-General, 5-Specific, quantitative). This design made it possible both to compute scores for each category and explore correlations among question and response attributes. An example is shown in Figure 6 showing the correlation between the average quality and the other three criteria. Relevance, in particular, tracks more closely with the perception of quality of the human reviewers.

Grading both questions and answers also allowed exploring correlations between each of the dimensions. The analysis split the scores into two groups, one for high scores and another one for below average, and built contingency tables for all possible pairs of different criteria. Through the application of the Fisher exact test to each contingency table allowed them to identify significant differences (p-values < 0.05). The results were consistent with the presence of statistically significant correlations for three different cases: the overall quality of the response and the difficulty of the question (higher difficulty led to lower quality responses), the relevance of the response and the difficulty of the question (more difficult questions lead to less relevant response), and the accuracy of the response and the specificity of the question (more specific questions led to less accurate answers).

A key advantage of using open ended responses is that experts can apply their expertise to study in detail the model responses, for instance to find evidence of hallucination, implausible recommendations or overly vague but reasonably sounding responses, all of which are described in Ref. [27]. In fact, while three models, GPT-3.5, GPT-4.0 and GPT-4o were tested, only GPT-4o produced responses of high enough quality to merit a dedicated study. On the other hand, the use of human experts as reviewers limits the scale at which these experiments have been run. This has led to the exploration of model-as-a-judge approaches.

3.2.3 Evaluating tool use in atomic layer deposition

As mentioned in Section 2.2, Ref [8] presents the design of an experimental atomic layer deposition chamber that has been integrated with an AI agent. At the core

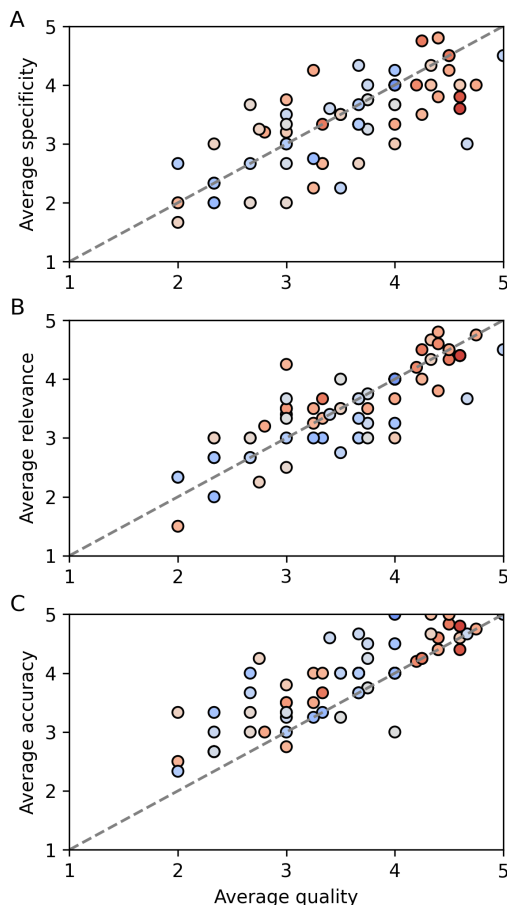


Fig. 6 Correlation between the average quality of the GPT4o responses to the ALDBench benchmark and (A) the specificity of the response; (B) the relevance of the response; (C) the accuracy of the response. Points are colored by average difficulty of each question (red-easy, blue-hard). Reproduced from Ref. [27] with permission.

of that integration is a simple API that provides a small number of functions that models can use to request information and execute growths. In order to understand the performance of the AI agents, the work presents a systematic comparison of a common agent architecture using different underlying LLMs.

The benchmark comprised a series of challenges that each required a single tool use call, in this case the call required to execute a given growth. For each challenge, a query was passed to the model together with the required background information, which in this case comprised the ALD tool configuration and potentially a list of prior experiments. Similar to the approach used in ToolBench [31], the prompt information describing the tool was generated dynamically from the reactor configuration. The model was then required to provide one or more tool calls to the reactor API required

to carry out the growth experiments that fulfilled the query. These responses were encoded using JSON.

Each benchmark challenge in required the model to choose the channels corresponding to the precursor and the co-reactant and the number of cycles. Some more complex challenges required growing doped materials, where $n \times \text{AB} + n \times \text{CD}$ super-cycles are used to form a doped or multicomponent material. A third type of challenge involved a functionalization step requiring only one precursor[8].

As in the ToolBench protocol, responses are scored based on how closely they match the target solution. Errors could be due to the model failing to provide the right JSON (corresponding to the failure to construct the right tool call response), selecting the wrong process (different channel numbers), or whether the right process was used but the wrong number of channels were provided. These two cases correspond to the case of using the wrong arguments. Furthermore, the study broke down the challenges in two broad categories: the first were instruction challenges, in which the agent were required to translate a request, for instance "Grow ten cycles of TMA/water" into the right function call given the reactor configuration. The second were process-discovery challenges, in which the query provided the desired outcome, for instance "Grow ten cycles of alumina", and the model had to infer both the right precursor and the channels.

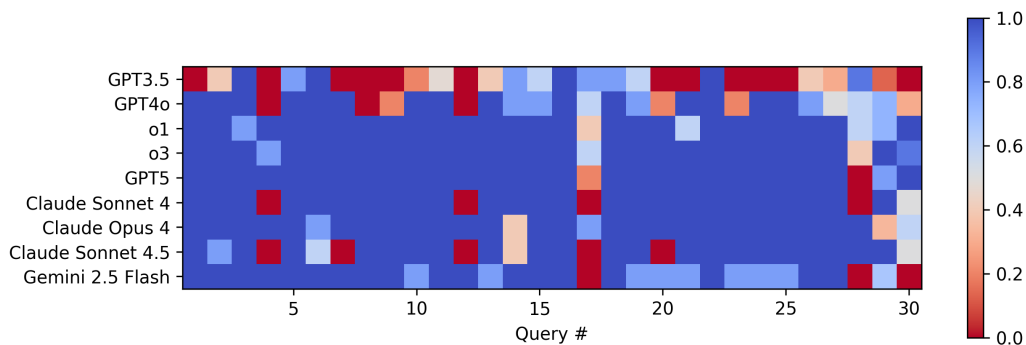


Fig. 7 Performance across 30 ALD process identification challenges of a simple agent built on different large language models. Each datapoint is the average of 5 independent runs. The scale is from 0 (complete failure) to 1 (perfect score). Reproduced from Ref. [8] with permission.

In Figure 7 we show the performance for identical agents relying on different LLMs for the process discovery challenges and in Table 1 we reproduce the key results. All scores are between 0 (bad) to 1 (perfect). The results show that, while the majority of the model excelled at instruction challenges, models comparatively struggled on process discovery challenges. This highlights the need to structure benchmarks in a way that highlight the model’s performance in specific skills. The results in Table 1 suggest that the models are good at translating direct instructions into function

Table 1 Average scores in instruction and process identification tasks as a function of the underlying large language model. The scale is from 0 (complete failure) to 1 (perfect score). Data from [8]

Model	Instruction	Process Identification
GPT3.5	0.80(0.09)	0.39(0.04)
GPT4o	1.0(0)	0.72(0.04)
o1	1.0(0)	0.94(0.04)
o3	1.0(0)	0.96(0.03)
GPT5	1.0(0)	0.93(0.01)
Claude Sonnet 4	1.0(0)	0.85(0.02)
Claude Opus 4	1.0(0)	0.93(0.02)
Claude Sonnet 4.5	0.96(0.05)	0.78(0.02)
Gemini 2.5 Flash	0.88(0.07)	0.84(0.06)
Average	0.96(0.07)	0.82(0.17)

calls to an experimental tool, but that they find identifying the right processes more challenging.

In fact, a more in-depth analysis carried out in Ref [8] showed that, on aggregate, models tend to perform worse on challenges involving processes that are not broadly represented in the literature as well as those involving counter intuitive chemistries, such as the use of a typical oxidant as a co-reactant to grow a pure metal. These insights were obtained because the process identification benchmark comprises 30 challenges tailored to a given materials synthesis technique. The results are also specific to the type of agent and the interface used in that work. This highlights the importance of designing benchmarks that are specific to each experimental tool and interface and that can cover a wide range of different challenges and conditions.

3.2.4 Closed-loop ALD process optimization benchmarks

Benchmarks focused on process optimization require the agent to interact with the experimental tool, similar to the multi-step manipulation scenario discussed in Section 3.1.2. In the case of ALD, this evaluation was done using a virtual tool, a simulation that, in addition to modeling an ALD process, returns feedback representative of an in-situ characterization technique such as quartz crystal microbalance or spectroscopic ellipsometry, incorporating realistic constraints such as the addition of noise and a reduced number of significant digits.

While Paulson et al used this approach to evaluate various machine learning approaches[4], Ref. [34] applied this methodology to evaluate the ability of agents built on top of reasoning large language models to optimize five different atomic layer deposition processes, chosen to represent a range of prototypical behaviors, including differences in how rapidly saturation is reached, how sharp the saturation curve is, different values of the growth per cycle, and whether they had a non-self limited component. For most of the cases explored, the agent did not have any prior information about the ALD process.

As mentioned in Section 3.2.1, optimizing an ALD process involves identifying the shortest possible dose times that still yield a growth per cycle close to the saturated

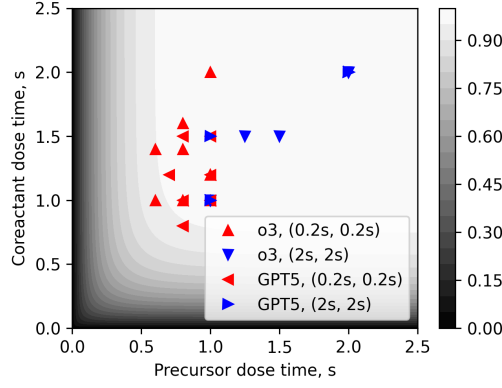


Fig. 8 Optimized dose time for one of the ALD process optimization baselines. In addition to the impact of the initial conditions, independent runs return different optimal conditions, highlighting the importance of considering the stochasticity of LLM’s response. Fig. from Ref. [34] with permission.

value. Consequently, the outcome of the optimization process is the final precursor and co-reactant dose times together with the corresponding growth per cycle. The total number of experiments requested and the number of iterations required by the agent to complete the optimization process were also tracked. These two numbers differ because the agent often requests more than one experiment per iteration.

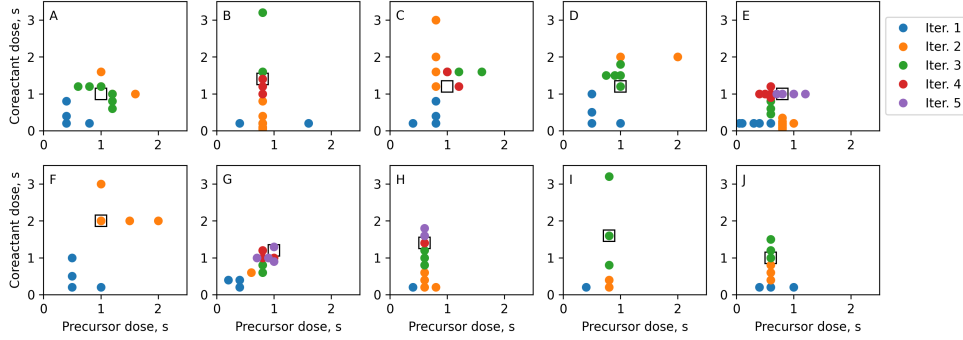


Fig. 9 Each plot in this figure represents an independent run of the agent trying to optimize the same ALD process. Points represents different conditions explored by the agent, with the color reflecting the iteration in which that specific growth was last requested. The plot shows a large variation in search strategies across ten different runs. Reproduced from Ref. [34] with permission.

To address the inherent stochasticity of LLMs, ten independent runs were carried out per condition. While the agent almost always reported that the optimization had concluded satisfactorily, and the growth per cycle were close to the saturation growth per cycle, a significant variability in the dose times was observed even when starting from the same initial condition. In Figure 8 we show an example of such variability

in the optimization outcomes. Similarly, the search strategy changed from run to run. In Fig. 9, the explored conditions are colored by the iteration in which each growth was last requested. These results highlight the importance of designing evaluation strategies that can evaluate the impact of stochasticity in the way an agent interact with its environment.

4 Discussion and practical recommendations

4.1 A practical evaluation framework for materials synthesis

As mentioned in Section 3, there is a huge gap in our understanding of the performance of LLMs for materials synthesis. Only a few techniques have specific benchmarks, and in most cases they have a limited coverage that does not provide a complete picture of the performance of LLM-based agents for that specific synthesis technique. Even in the case of ALD treated in Section 3.2, the range of benchmarks is still incomplete. On the other hand, there are a number of well tested strategies to evaluate LLM-based models and agents that are directly applicable to materials synthesis.

This leads us to propose an evaluation strategy for materials synthesis, one that integrates different methodologies to evaluate LLM-based models and agents, including their integration with experiments. This multistep approach is geared towards maximizing our understanding of the potential of these techniques in a way that is accessible to non-AI researchers:

1. **Expert-reviewed open-response benchmark:** The first step is developing a set of questions to help gain a sense of the model’s abilities and knowledge around a specific technique. The quality of the responses can be reviewed by domain experts, for instance using the methodology described in Section 3.2.2. From a methodological standpoint, the key consideration is to ensure that each question is answered independently. For instance, when using chatbot type of interfaces, the model’s context window grows with prior questions, which can affect the model’s response. Using separate API calls for each question sidesteps this issue.
2. **Multiple choice and open ended benchmark with LLM-as-a-judge evaluation:** Multiple choice questions and numerical open ended questions offer a scalable pathway to evaluate LLMs, particularly if LLMs are used as a judge to disambiguate open ended responses and extract any numerical value. This approach allows scaling up to thousands of questions, as shown for the case of MatSciBench. The methodology should follow a workflow similar to that described in described in Figure 4. Given a corpus of research papers and other background materials, approaches such as that of Li and Cole can be used to generate question sets using LLMs[24].
3. **Low-level tool use benchmark:** These benchmarks should be directed at quantifying an agent’s ability to generate one or more calls to the experimental tool, essentially the ability to translate instructions into concrete interactions with the experiment (e.g. "set the power of target 3 to 100 W for ten minutes"). These challenges can also be evaluated using the same workflow described in Fig. 4.

4. **Synthesis-challenge benchmark:** A variation of the previous case, it is possible to apply a similar methodology to materials synthesis challenges where, in addition to gauging the model’s ability to execute specific instructions, it quantifies the model’s ability to infer the instructions from the case. An example are the process identification challenges described in Section 3.1.2 for the case of ALD.
5. **Closed-loop, multistep interaction benchmark:** This type of benchmark requires a virtual tool, a simulation or model of the experimental system capable of returning adequate feedback in a way that is consistent with specific characterization techniques, to allow the agent interact with the experimental system until a certain requirement is met. Since the goal is evaluating the performance of an agent in as general conditions as possible, virtual tools do not have to refer to any specific process or tool. Instead, they should be able to sample a wide enough range of behaviors to maximize the confidence on the agent’s performance when facing a relevant research challenge in the lab.

4.2 Gaps, limitations, and opportunities

A comparison of the state of the art with the evaluation strategy described in Section 4.1 shows that there are huge gaps in our current understanding of the capabilities and limitations of models for materials synthesis. Even in the case of atomic layer deposition, arguably one of the best characterized techniques thus far, the picture is still far from complete. On the other hand, as emphasized in this review, over the past five years there has been a substantial development and refinement of evaluation methodologies for LLM models and agents[10]. There is therefore an opportunity to leverage those advances to develop benchmarks that are specifically tailored to different synthesis techniques.

One particularly important characteristic of LLMs is the stochastic nature of its output. In problems like ALD process optimization this leads to large run-to-run variabilities, as shown in Figs 8 and 9. Evaluation strategies targeting LLM-based systems need to address the impact of stochasticity if we want these techniques to become part of scientific research and materials synthesis workflows. A second challenge is understanding how robust the agent’s response is to small perturbations in the way the question is phrased. We are currently working on simple tools to amplify existing materials science benchmarks to address this question. For experimentally integrated agents, evaluation should not only assess task success but also safe failure behavior, including the ability to interrupt, request clarifications, and avoid potentially harmful actions under uncertainty.

Finally, it is important to highlight that the specific LLM, generation hyperparameters, prompting strategy, tools and information accessible to the agent, and the design of the harness itself will all affect its performance on any benchmark. Consequently, it is important to consistently keep evaluating the performance of any new agent on the relevant benchmarks to ensure that the performance does not regress as the design evolved. This ability to quickly iterate on an agent design is one of the main advantages of developing a robust set of technique and tool specific benchmarks, providing a safe pathway towards fast design iterations without having to compromise on safety or having to wait for the outcome of the experiments they are controlling.

5 Conclusions

The integration of LLM-based agents with materials synthesis tools creates new opportunities for autonomous experimentation, but it also raises key questions in terms of the performance, robustness, reproducibility, and safety of the AI models. In this review, we have provided an overview of evaluation strategies, and argued that a robust evaluation in the context of materials synthesis requires a careful approach spanning multiple facets, such as knowledge and reasoning, tool use, synthesis-specific challenges, and closed-loop interactions with experimental or virtual tools.

Atomic layer deposition provides a valuable case study because its high degree of automation enables evaluation at multiple levels, from process identification and tool use to full process optimization. The examples discussed here highlight both the promise of current LLM-based agents and their limitations, particularly with respect to their ability to infer new processes, their robustness, and their run-to-run variability. Developing more comprehensive, synthesis-specific benchmarks will be essential for safely integrating these systems into research workflows and accelerate the development of new models and agents that are tailored to materials synthesis needs.

Acknowledgements. This research is based upon work supported by Laboratory Directed Research and Development (LDRD) funding from Argonne National Laboratory, provided by the Director, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-06CH11357. The author also thanks the U.S. Department of Energy’s Genesis Mission and the Office of Science, Office of Advanced Scientific Computing Research’s ModCon.

Declarations

Funding

DE-AC02-06CH11357

Conflict of interest

On behalf of all authors, the corresponding author states that there is no conflict of interest

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Data availability

All the data used to generate the figures is available from the author upon request.

Materials availability

Not applicable.

Code availability

Not applicable.

Author contribution

The sole author responsible for the conception, literature review, and writing of this manuscript is AY

References

- [1] Häse, F., Roch, L.M., Aspuru-Guzik, A.: Next-generation experimentation with self-driving laboratories. *Trends in Chemistry* **1**(3), 282–291 (2019) <https://doi.org/10.1016/j.trechm.2019.02.007>
- [2] MacLeod, B.P., Parlane, F.G.L., Morrissey, T.D., Häse, F., Roch, L.M., Dettelbach, K.E., Moreira, R., Yunker, L.P.E., Rooney, M.B., Deeth, J.R., Lai, V., Ng, G.J., Situ, H., Zhang, R.H., Elliott, M.S., Haley, T.H., Dvorak, D.J., Aspuru-Guzik, A., Hein, J.E., Berlinguette, C.P.: Self-driving laboratory for accelerated discovery of thin-film materials. *Science Advances* **6**(20), 8867 (2020) <https://doi.org/10.1126/sciadv.aaz8867>
<https://www.science.org/doi/pdf/10.1126/sciadv.aaz8867>
- [3] Stach, E., DeCost, B., Kusne, A.G., Hattrick-Simpers, J., Brown, K.A., Reyes, K.G., Schrier, J., Billinge, S., Buonassisi, T., Foster, I., Gomes, C.P., Gregoire, J.M., Mehta, A., Montoya, J., Olivetti, E., Park, C., Rotenberg, E., Saikin, S.K., Smullin, S., Stanev, V., Maruyama, B.: Autonomous experimentation systems for materials development: A community perspective. *Matter* **4**(9), 2702–2726 (2021) <https://doi.org/10.1016/j.matt.2021.06.036>
- [4] Paulson, N.H., Yanguas-Gil, A., Abuomar, O.Y., Elam, J.W.: Intelligent agents for the optimization of atomic layer deposition. *ACS Applied Materials & Interfaces* **13**(14), 17022–17033 (2021) <https://doi.org/10.1021/acsami.1c00649>
- [5] Boiko, D.A., MacKnight, R., Kline, B., Gomes, G.: Autonomous chemical research with large language models. *Nature* **624**(7992), 570–578 (2023) <https://doi.org/10.1038/s41586-023-06792-0>
- [6] Mandal, I., Soni, J., Zaki, M., Smedskjaer, M., Wondraczek, K., Wondraczek, L., Gosvami, N., Krishnan, N.: Evaluating large language model agents for automation of atomic force microscopy. *Nature Communications* **16**(1) (2025) <https://doi.org/10.1038/s41467-025-64105-7>

- [7] Harris, S.B., Biswas, A., Yun, S.J., Roccapiore, K.M., Rouleau, C.M., Poretzky, A.A., Vasudevan, R.K., Geohegan, D.B., Xiao, K.: Autonomous synthesis of thin film materials with pulsed laser deposition enabled by in situ spectroscopy and automation. *Small Methods* **8**(9), 2301763 (2024) <https://doi.org/10.1002/smtd.202301763> <https://onlinelibrary.wiley.com/doi/pdf/10.1002/smtd.202301763>
- [8] Yanguas-Gil, A., Jones, J.C., Kim, S., Nguyen, C.T., Elam, J.W.: Design and performance of ai agents interfacing with an atomic layer deposition tool. *Review of Scientific Instruments* **97**(5), 053903 (2026) <https://doi.org/10.1063/5.0318770>
- [9] Zhang, Y., Khan, S.A., Mahmud, A., Yang, H., Lavin, A., Levin, M., Frey, J., Dunnmon, J., Evans, J., Bundy, A., Dzeroski, S., Tegner, J., Zenil, H.: Exploring the role of large language models in the scientific method: from hypothesis to discovery. *npj Artificial Intelligence* **1**(1), 14 (2025) <https://doi.org/10.1038/s44387-025-00019-5>
- [10] Chang, Y., Wang, X., Wang, J., Wu, Y., Yang, L., Zhu, K., Chen, H., Yi, X., Wang, C., Wang, Y., Ye, W., Zhang, Y., Chang, Y., Yu, P., Yang, Q., Xie, X.: A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology* **15**(3) (2024) <https://doi.org/10.1145/3641289>
- [11] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems. NIPS'17*, pp. 6000–6010. Curran Associates Inc., Red Hook, NY, USA (2017). <https://doi.org/10.48550/arXiv.1706.03762>
- [12] Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D.M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D.: Language models are few-shot learners. In: *Proceedings of the 34th International Conference on Neural Information Processing Systems. NIPS '20*. Curran Associates Inc., Red Hook, NY, USA (2020). <https://doi.org/10.48550/arXiv.2005.14165>
- [13] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E.H., Le, Q.V., Zhou, D.: Chain-of-thought prompting elicits reasoning in large language models. In: *Proceedings of the 36th International Conference on Neural Information Processing Systems. NIPS '22*. Curran Associates Inc., Red Hook, NY, USA (2022). <https://doi.org/10.48550/arXiv.2201.11903>
- [14] Wang, X., Wei, J., Schuurmans, D., Le, Q.V., Chi, E.H., Narang, S., Chowdhery, A., Zhou, D.: Self-consistency improves chain of thought reasoning in language models. In: *The Eleventh International Conference on Learning Representations* (2023). <https://doi.org/10.48550/arXiv.2203.11171>

- [15] Guo, D., Yang, D., Zhang, H., Song, J., Wang, P., Zhu, Q., Xu, R., Zhang, R., Ma, S., Bi, X., Zhang, X., Yu, X., Wu, Y., Wu, Z.F., Gou, Z., Shao, Z., Li, Z., Gao, Z., Liu, A., Xue, B., Wang, B., Wu, B., Feng, B., Lu, C., Zhao, C., Deng, C., Ruan, C., Dai, D., Chen, D., Ji, D., Li, E., Lin, F., Dai, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Xu, H., Ding, H., Gao, H., Qu, H., Li, H., Guo, J., Li, J., Chen, J., Yuan, J., Tu, J., Qiu, J., Li, J., Cai, J.L., Ni, J., Liang, J., Chen, J., Dong, K., Hu, K., You, K., Gao, K., Guan, K., Huang, K., Yu, K., Wang, L., Zhang, L., Zhao, L., Wang, L., Zhang, L., Xu, L., Xia, L., Zhang, M., Zhang, M., Tang, M., Zhou, M., Li, M., Wang, M., Li, M., Tian, N., Huang, P., Zhang, P., Wang, Q., Chen, Q., Du, Q., Ge, R., Zhang, R., Pan, R., Wang, R., Chen, R.J., Jin, R.L., Chen, R., Lu, S., Zhou, S., Chen, S., Ye, S., Wang, S., Yu, S., Zhou, S., Pan, S., Li, S.S., Zhou, S., Wu, S., Yun, T., Pei, T., Sun, T., Wang, T., Zeng, W., Liu, W., Liang, W., Gao, W., Yu, W., Zhang, W., Xiao, W.L., An, W., Liu, X., Wang, X., Chen, X., Nie, X., Cheng, X., Liu, X., Xie, X., Liu, X., Yang, X., Li, X., Su, X., Lin, X., Li, X.Q., Jin, X., Shen, X., Chen, X., Sun, X., Wang, X., Song, X., Zhou, X., Wang, X., Shan, X., Li, Y.K., Wang, Y.Q., Wei, Y.X., Zhang, Y., Xu, Y., Li, Y., Zhao, Y., Sun, Y., Wang, Y., Yu, Y., Zhang, Y., Shi, Y., Xiong, Y., He, Y., Piao, Y., Wang, Y., Tan, Y., Ma, Y., Liu, Y., Guo, Y., Ou, Y., Wang, Y., Gong, Y., Zou, Y., He, Y., Xiong, Y., Luo, Y., You, Y., Liu, Y., Zhou, Y., Zhu, Y.X., Huang, Y., Li, Y., Zheng, Y., Zhu, Y., Ma, Y., Tang, Y., Zha, Y., Yan, Y., Ren, Z.Z., Ren, Z., Sha, Z., Fu, Z., Xu, Z., Xie, Z., Zhang, Z., Hao, Z., Ma, Z., Yan, Z., Wu, Z., Gu, Z., Zhu, Z., Liu, Z., Li, Z., Xie, Z., Song, Z., Pan, Z., Huang, Z., Xu, Z., Zhang, Z., Zhang, Z.: DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature* **645**(8081), 633–638 (2025) <https://doi.org/10.1038/s41586-025-09422-z>
- [16] Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., Zheng, C., Liu, D., Zhou, F., Huang, F., Hu, F., Ge, H., Wei, H., Lin, H., Tang, J., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Zhou, J., Lin, J., Dang, K., Bao, K., Yang, K., Yu, L., Deng, L., Li, M., Xue, M., Li, M., Zhang, P., Wang, P., Zhu, Q., Men, R., Gao, R., Liu, S., Luo, S., Li, T., Tang, T., Yin, W., Ren, X., Wang, X., Zhang, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Zhang, Y., Wan, Y., Liu, Y., Wang, Z., Cui, Z., Zhang, Z., Zhou, Z., Qiu, Z.: Qwen3 Technical Report (2025). <https://arxiv.org/abs/2505.09388>
- [17] Schick, T., Dwivedi-Yu, J., Dessí, R., Raileanu, R., Lomeli, M., Hambro, E., Zettlemoyer, L., Cancedda, N., Scialom, T.: Toolformer: language models can teach themselves to use tools. In: Proceedings of the 37th International Conference on Neural Information Processing Systems. NIPS ’23. Curran Associates Inc., Red Hook, NY, USA (2023). <https://doi.org/10.48550/arXiv.2302.04761>
- [18] Patil, S.G., Zhang, T., Wang, X., Gonzalez, J.E.: Gorilla: Large Language Model Connected with Massive APIs (2023). <https://arxiv.org/abs/2305.15334>
- [19] Liu, H., Li, C., Wu, Q., Lee, Y.J.: Visual Instruction Tuning (2023). <https://arxiv.org/abs/2304.08485>

- [20] Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., Cao, Y.: ReAct: Synergizing Reasoning and Acting in Language Models (2023). <https://arxiv.org/abs/2210.03629>
- [21] Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach, 3rd edn. Prentice Hall Press, USA (2009)
- [22] Rein, D., Hou, B.L., Stickland, A.C., Petty, J., Pang, R.Y., Dirani, J., Michael, J., Bowman, S.R.: GPQA: A Graduate-Level Google-Proof Q&A Benchmark (2023). <https://arxiv.org/abs/2311.12022>
- [23] Zaki, M., Jayadeva, Mausam, Krishnan, N.: MaScQA: investigating materials science knowledge of large language models. Digital Discovery **3**(2), 313–327 (2024) <https://doi.org/10.1039/d3dd00188a>
- [24] Li, Z., Cole, J.: Auto-generating question-answering datasets with domain-specific knowledge for language models in scientific tasks. Digital Discovery **4**(4), 998–1005 (2025) <https://doi.org/10.1039/d4dd00307a>
- [25] Zhang, J., Gan, J., Wang, X., Jia, Z., Gu, C., Chen, J., Zhu, Y., Ma, M.D., Zhou, D., Li, L., Wang, W.: MatSciBench: Benchmarking the Reasoning Ability of Large Language Models in Materials Science (2026). <https://arxiv.org/abs/2510.12171>
- [26] Mirza, A., Alampara, N., Kunchapu, S., Ríos-García, M., Emoekabu, B., Krishnan, A., Gupta, T., Schilling-Wilhelmi, M., Okereke, M., Aneesh, A., Elahi, A.M., Asgari, M., Eberhardt, J., Elbeheiry, H.M., Gil, M.V., Greiner, M., Holick, C.T., Glaubitz, C., Hoffmann, T., Ibrahim, A., Klepsch, L.C., Köster, Y., Kreth, F.A., Meyer, J., Miret, S., Peschel, J.M., Ringleb, M., Roesner, N., Schreiber, J., Schubert, U.S., Stafast, L.M., Wonanke, D., Pieler, M., Schwaller, P., Jablonka, K.M.: Are large language models superhuman chemists? (2024). <https://arxiv.org/abs/2404.01475>
- [27] Yanguas-Gil, A., Dearing, M.T., Elam, J.W., Jones, J.C., Kim, S., Mohammad, A., Thang Nguyen, C., Sengupta, B.: Benchmarking large language models for materials synthesis: The case of atomic layer deposition. Journal of Vacuum Science & Technology A **43**(3), 032406 (2025) <https://doi.org/10.1116/6.0004319>
- [28] Alampara, N., Schilling-Wilhelmi, M., Ríos-García, M., Mandal, I., Khetarpal, P., Grover, H., Krishnan, N., Jablonka, K.: Probing the limitations of multimodal language models for chemistry and materials research. Nature Computational Science **5**(10), 952–961 (2025) <https://doi.org/10.1038/s43588-025-00836-3>
- [29] Circi, D., Khalighinejad, G., Chen, A., Dhingra, B., Brinson, L.: How well do large language models understand tables in materials science? Integrating Materials and Manufacturing Innovation **13**(3), 669–687 (2024) <https://doi.org/10.1007/s40192-024-00362-6>

- [30] Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., Le Noac’h, A., Li, H., McDonell, K., Muennighoff, N., Ociepa, C., Phang, J., Reynolds, L., Schoelkopf, H., Skowron, A., Sutawika, L., Tang, E., Thite, A., Wang, B., Wang, K., Zou, A.: The Language Model Evaluation Harness. Zenodo (2024). <https://doi.org/10.5281/zenodo.12608602>
- [31] Xu, Q., Hong, F., Li, B., Hu, C., Chen, Z., Zhang, J.: On the Tool Manipulation Capability of Open-source Large Language Models (2023). <https://arxiv.org/abs/2305.16504>
- [32] Liu, X., Yu, H., Zhang, H., Xu, Y., Lei, X., Lai, H., Gu, Y., Ding, H., Men, K., Yang, K., Zhang, S., Deng, X., Zeng, A., Du, Z., Zhang, C., Shen, S., Zhang, T., Su, Y., Sun, H., Huang, M., Dong, Y., Tang, J.: AgentBench: Evaluating LLMs as Agents (2025). <https://arxiv.org/abs/2308.03688>
- [33] Miret, S., Krishnan, N.M.A.: Enabling large language models for real-world materials discovery. *Nature Machine Intelligence* **7**(7), 991–998 (2025) <https://doi.org/10.1038/s42256-025-01058-y>
- [34] Yanguas-Gil, A.: Performance of AI agents based on reasoning language models on ALD process optimization tasks. *Journal of Vacuum Science & Technology A* **44**(4), 043410 (2026) <https://doi.org/10.1116/6.0005313>
- [35] Cho, Z., Osvaldo, M., Doloi, S., Das, M., Goh, J., Tan, B., Wang, J., Li, Y., Xiao, X., Joshi, A., Ng, L.: Large language models in materials science: assessing RAG evaluation frameworks through graphene synthesis. *RSC Advances* **16**(13), 11306–11313 (2026) <https://doi.org/10.1039/d5ra09726f>
- [36] Kim, S., Jung, Y., Schrier, J.: Large language models for inorganic synthesis predictions. *Journal of the American Chemical Society* **146**(29), 19654–19659 (2024) <https://doi.org/10.1021/jacs.4c05840>
- [37] Kononova, O., Huo, H., He, T., Rong, Z., Botari, T., Sun, W., Tshitoyan, V., Ceder, G.: Text-mined dataset of inorganic materials synthesis recipes. *Scientific Data* **6**(1), 203 (2019) <https://doi.org/10.1038/s41597-019-0224-1>
- [38] Kessels, E., Devi, A., Park, J.-S., Ritala, M., Yanguas-Gil, A., Wiemer, C.: Atomic layer deposition. *Nature Reviews Methods Primers* **5**(1), 66 (2025) <https://doi.org/10.1038/s43586-025-00435-6>