MAPS: ADVANCING MULTI-MODAL REASONING IN EXPERT-LEVEL PHYSICAL SCIENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Pre-trained on extensive text and image corpora, current Multi-Modal Large Language Models (MLLM) have shown strong capabilities in general visual reasoning tasks. However, their performance is still lacking in physical domains that require understanding diagrams with complex physical structures and quantitative analysis based on multi-modal information. To address this, we develop a new framework, named **M**ulti-Modal Scientific Re**A**soning with **P**hysics Perception and **S**imulation (**MAPS**) based on an MLLM. MAPS decomposes expert-level multi-modal reasoning task into physical diagram understanding via a Physical Perception Model (PPM) and reasoning with physical knowledge via a simulator. The PPM module is obtained by fine-tuning a visual language model using carefully designed synthetic data with paired physical diagrams and corresponding simulation language descriptions. At the inference stage, MAPS integrates the simulation language description of the input diagram provided by PPM and results obtained through a Chain-of-Simulation process with MLLM to derive the underlying rationale and the final answer. Validated using our collected college-level circuit analysis problems, MAPS significantly improves reasoning accuracy of MLLM and outperforms all existing models. The results confirm MAPS offers a promising direction for enhancing multi-modal scientific reasoning ability of MLLMs. We will release our code, model and dataset used for our experiments upon publishing of this paper.

1 INTRODUCTION

Pre-trained on large-scale text and image corpora, Multi-Modal Large Language Models (MLLM) exhibit strong capabilities in general visual reasoning tasks, including image captioning and visual question-answering (Li et al., 2022; Team et al., 2023; AI; Liu et al., 2024). Through elaborated pre-training and post-training, the proficiency of LLMs in text-only mathematical reasoning and programming has significantly improved (Hendrycks et al., 2021; Lu et al., 2022; Lightman et al., 2023), broadening their applications to more scientific and professional tasks. However, for scientific disciplines that require understanding complex physical structures in images and mathematical reasoning based on scientific knowledge from multi-modal information, the capabilities of MLLMs remain weak (Yue et al., 2023). This limitation hinders their further application in educational, academic, and industrial scenarios. Thus, enhancing the multi-modal reasoning abilities of MLLMs in expert-level physical sciences while extending their application scenarios is a valuable yet challenging research direction.

The current methods in multi-modal reasoning (Zhang et al., 2023b; Zheng et al., 2023; Mitra et al., 2024) primarily concentrate on generating a rationale that integrates multi-modal information, allowing the model to derive the final answer from this intermediate result. This process is commonly referred to as Chain-of-Thought (**CoT**) (Wei et al., 2022) reasoning. Another commonly adopted pathway is to integrate LLMs with external tools, including small-sized specialized multi-modal models, as well as software such as code interpreter (Gao et al., 2023; Wang et al., 2024a). However, these methods mainly focus on general images or diagrams containing simple physical information, making it difficult to directly transfer to scientific scenarios that involve complex physical diagrams and require precise numerical analysis.

To address the aforementioned limitations, we proposed **M**ulti-Modal Scientific Re**A**soning with **P**hysics Perception and **S**imulation (**MAPS**), a novel framework for solving complex multi-modal

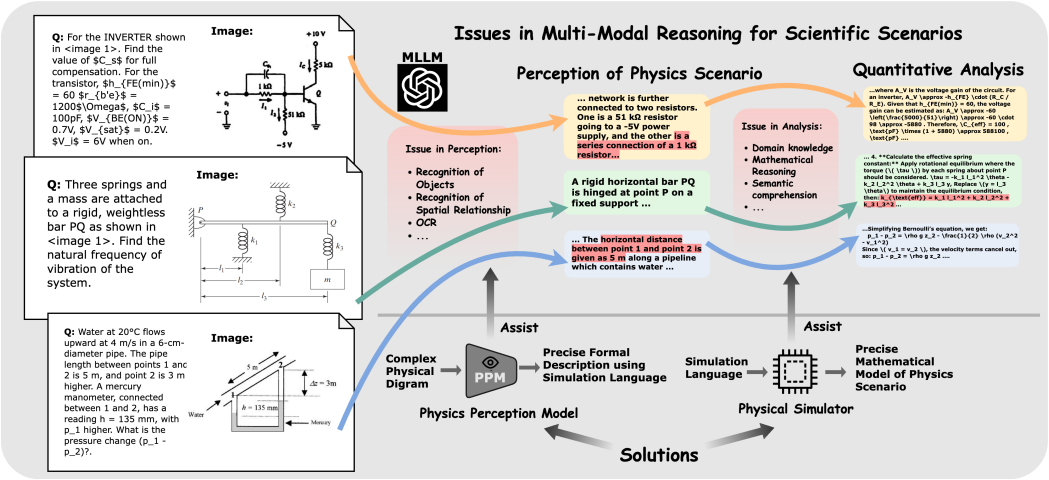


Figure 1: Two Issues in Multi-Modal Reasoning for Scientific Scenarios and Our Solutions. The location of the model error is highlighted in red. The scientific questions are sampled from MMMU (Yue et al., 2023).

reasoning problems in physical disciplines. And in this paper we verified the effectiveness of MAPS in electrical discipline, which typically involves multiple circuit diagrams and is representative of expert-level physical science requiring reasoning on complex physical diagrams. The core idea of MAPS is to decompose expert-level reasoning problems into two sub-tasks: understanding the physical diagram and reasoning based on this comprehension and related physical knowledge. MAPS realizes physical diagram understanding by fine-tuning a visual language model using carefully designed synthetic data, resulting in what we term the Physics Perception Model (PPM). The role of PPM is to translate physical diagrams into simulation language descriptions that can be executed by a simulator. At the inference stage, MAPS integrates the converted simulation language description and their respective simulation results, obtained through a Chain-of-Simulation process, to derive intermediate rationales and ultimately the final answer to the question. Experiment results on college-level circuit analysis problems demonstrate that our framework can successfully address the challenges in complex multi-modal reasoning tasks in physical science. Most importantly, it significantly reduces the occurrence of hallucination when using and solving physical equations. This advancement creates new avenues for precise multi-modal scientific reasoning using MLLMs.

To summarize, our main contributions in this work are as follows:

- We introduce a novel multi-modal reasoning framework MAPS to address the current limitations of MLLMs in solving expert-level scientific problems involving complex physical diagrams. MAPS incorporates MLLMs with a finetuned perception model and physical simulator to improve the precision of its reasoning steps and results.
- Through our experiments on college-level circuit analysis problems, we demonstrate that MAPS significantly outperforms existing methods, offering a viable pathway to build multi-modal solutions for expert-level scientific problems.
- We devise an automated pipeline to synthesize diverse paired training data for finetuning an MLLM. By leveraging intrinsic generalization ability of pre-trained models, the pipeline helps MLLMs effectively adapts to complex real-world problems, alleviating the issue of data scarcity in scientific domains.

2 MOTIVATION

Following the human approach to solving science problems with diagrams, we break down the problem into two steps: understanding the physical context in the multi-modal input (**Perception**) and using scientific knowledge and mathematical deduction to derive the answer (**Analysis**). Based on these two steps, we summarize the limitations of current MLLM-based solutions for solving such problems into two main categories:

Issues in Perception. Based on observations reported in the MMMU benchmark (Yue et al., 2023) and our empirical studies, we found that current general purpose MLLMs, including the most powerful ones such as GPT-4V and Claude-3.5, exhibit poor perception abilities in understanding diagrams related to physical sciences (e.g., circuit diagrams). This corresponds to the perceptual error identified in the error analysis in (Yue et al., 2023). This significantly limits their application in the field of scientific reasoning with multi-modal input.

Issues in Analysis. Although MLLMs can sometimes correctly understand diagrams, their domain knowledge and mathematical reasoning abilities can still be lacking. This often leads to hallucinations during further reasoning steps, resulting in misleading answers.

We offer some specific cases in Figure 1, illustrating how an off-the-shelf MLLM makes mistakes in perception and analysis steps. Based on these observations, we decide to decompose this complex multi-modal reasoning task into sub-tasks and leverage expert models and domain-specific tools to solve the sub-tasks that are infeasible for current MLLMs. Concretely, as shown in Figure 1, our proposed two solutions to the two issues mentioned above are:

Solution to *Issue in Perception*: Translate physical diagrams into simulation language descriptions. We adopt simulation language for two reasons. First, it describes the physical scene of the diagram using a formal language, enabling the language model to directly access the fundamental structure behind the question. Second, with parameters of physical objects provided, we can directly use a simulator to obtain all states and observation values of the physical scene. In the context of circuit analysis, we use SPICE (Nagel, 1975) as our simulation language. For other scenarios, there are corresponding choices, such as ANSYS APDL (Kohnke, 1982) in mechanical disciplines and ZPL (Laikin, 2018) in optics domains. Specifically, we develop an expert visual language model to complete this conversion. Since there is no large-scale available dataset or existing model for this task, we devise a data synthesis pipeline to generate abundant physical diagrams and their corresponding simulation languages for our visual language model training.

Solution to *Issue in Analysis*: Reasoning under the assistance of simulation. Although current MLLMs can perform mathematical reasoning using external tools (Chen et al., 2022; Zhou et al., 2023), recent research found it still challenging to prompt LLMs to write programs for solving scientific problems (Tian et al., 2024). In the benchmark evaluating real-world scientific programming tasks, even the best model achieves an accuracy of less than 10% in completing a main problem. To address the issue of hallucination when MLLMs perform mathematical derivations and synthesize scientific programs, we delegate the main quantitative reasoning tasks to a domain-specific tool, namely a physical simulator. The simulator comprises domain-specific knowledge and thus is guaranteed to be precise in its output with respect to the given input.

Combining the two solutions above, we design a **Chain-of-Simulation** process that obtains simulation language description and simulation results utilizing the fine-tuned perception model and simulator, and prompt an MLLM to compute the answer under the assistance of simulation language description and simulation results at the inference stage.

3 METHODOLOGY

Our proposed MAPS framework, illustrated in Figure 2, consists of two phases: the **Physics Perception Model (PPM) construction phase** and the **Inference phase**.

The core components of our framework are as follows:

- **Physics Perception Model (PPM).** It serves as an expert perception model that translates a given physical diagram into a simulation language (SL) description. This model is fine-tuned from a pre-trained Visual Language Model (VLM) using a synthetic dataset designed for the diagram-to-SL conversion.
- **Physical Simulator.** The simulator is used to perform numerical simulations and obtain the state and observations about the physical scene carried in the diagram.
- **Multi-modal Large Language Model (MLLM)**. The MLLM primarily handles semantic understanding and basic mathematical reasoning, based on the results provided by PPM and the simulator. When solving physical problems with diagrams, the MLLM parses the target problem from the textual question, refines the simulation language description generated

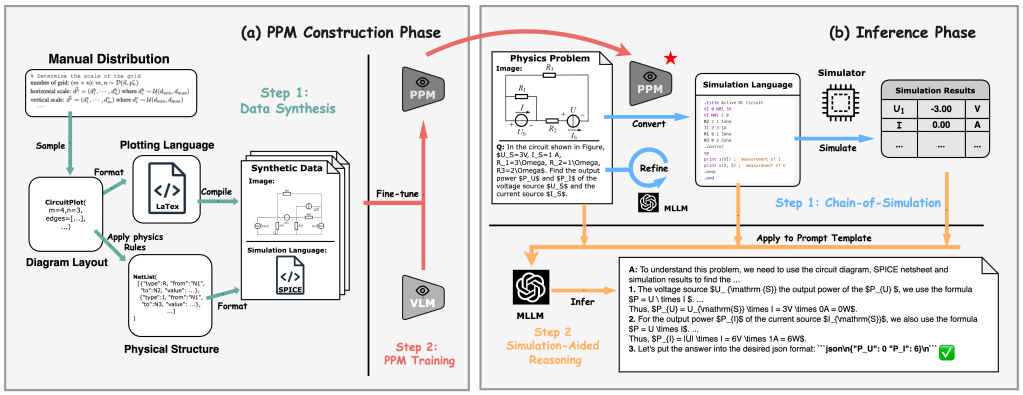


Figure 2: Our proposed **MAPS** framework is integrated with a Multi-modal Large Language Model (MLLM), a Physics Perception Model (PPM) and a Physical Simulator. **(a)** At **PPM Construction Phase**, we fine-tune a pre-trained VLM with carefully designed synthetic data to obtain PPM which can convert physical diagram into simulation language descriptions. **(b)** At **Inference Phase**, we apply **Chain-of-Simulation** to acquire simulation language description and simulation results which assist MLLM with the further reasoning to obtain final answer of original problem.

by the PPM, extracts useful simulation results, and performs final reasoning based on the original question and the added simulation information.

3.1 Inference Phase

We first introduce the inference phase because it conveys the main philosophy of our solution. As shown in Figure 2(b), this stage includes two steps: Chain-of-Simulation and Simulation-Aided Reasoning. Suppose we have a scientific problem with a physical diagram X_V in a pixel format and textual description X_L, our model is required to infer the answer Y_L.

3.1.1 Chain-of-Simulation

The first step in the Chain-of-Simulation (CoS) process is to use the PPM to convert the pixel schematic diagram X_V into an initial SL description Z. Since the problem involves multi-modal information, the initial SL description produced by the PPM may lack completeness in depicting the full physical scene. For example, in a circuit diagram, a resistor might be labeled as R_1, but its value might be provided in the accompanying textual description in the question X_L. To address this, we employ the MLLM to refine initial SL description based on textual input X_L. The MLLM incorporates additional information from the accompanying text, resulting in a comprehensive and accurate SL text that fully describes the physical scene.

Algorithm 1 MAPS: Inference Phase

Input: X_V, X_L, PPM, Simulator, MLLM
Output: Y_L
`% Chain-of-Simulation`
Obtain SL description $Z \leftarrow \text{PPM}(X_V)$
Refine SL description
  $Z \leftarrow \text{MLLM}(X_L, Z, \texttt{prompt_refine})$
Obtain simulation result $R \leftarrow \text{Simulator}(Z)$
`% Simulation-Aided Reasoning`
if check_valid(R) **then**
  $Y_L \leftarrow \text{MLLM}(X_L, X_V, Z, R, \texttt{prompt_sar})$
else
  $Y_L \leftarrow \text{MLLM}(X_L, X_V, Z, \texttt{prompt_sl})$
end if
return Y_L

Once the comprehensive SL description Z is generated, it is fed into the physical simulator to perform physical simulations. This process produces simulation result R, including state values and observation values of the physical scene. This approach effectively mitigates mathematical reasoning errors that may arise from the model's hallucinations in scientific computation, ensuring accurate and reliable results.

3.1.2 Simulation-Aided Reasoning

After the CoS process, MAPS will apply the question information (X_L, X_V), SL description Z, and simulation result R to a well-designed prompt template. This template prompts the MLLM

to generate further rationale and infer the final answer Y_L. We consider the simulation language and simulation results as intermediate rationale in the model's reasoning process, similar to various Chain-of-Thought (CoT) mechanisms in existing prompting methods (Mitra et al., 2024; Zhang et al., 2023b; Zheng et al., 2023). The entire process is illustrated using pseudo-code in Algorithm 1. Experiments show that under the assistance of CoS, the MLLM can be prompted to accurately answer questions, effectively narrowing the gap between current MLLM capability and expert-level performance on scientific problems.

3.2 PPM Construction Phase

Accurate conversion from pixel schematics to simulation language descriptions is crucial for the CoS to function effectively. We highlight this importance with a red star in Figure 2, emphasizing the significance of PPM in our framework. Due to the scarcity of real-world paired data that maps physical diagrams to simulation language descriptions, which is crucial for training Vision-Language Models (VLMs) to recognize the physical diagrams of interest, we choose to synthesize the paired data. To achieve this, we craft rules to generate a large dataset comprising diverse circuit diagrams and their corresponding simulation language descriptions. These synthetic data are then used to fine-tune the pre-trained VLM, ultimately producing the PPM.

3.2.1 Data Synthesis

The data synthesis process is depicted in Figure 2(a). And the detailed steps of a generation process are described as follows.

The diagram layout is our data structure designed to correspond to the plotting language, encompassing all the physical objects, their displayed positions and annotations in the diagram. Subsequently, the pipeline synthesizes the diagram and the corresponding SL description through two paths: the **diagram synthesis path** and the **simulation language (SL) synthesis path**.

Diagram synthesis path. As shown in the upper branch of Figure 2(a), the diagram layout is first converted to a plotting language. There are various plotting languages available, such as LaTeX (TikZ) and Graphviz, which use formal syntax to describe diagrams and can be compiled into pixel images. The design of diagram layout allows for a straightforward transformation from diagram layout to plotting language. Finally, we compile the generated plotting language using its designated compiler to generate the diagram in pixel format.

SL synthesis path. This path focuses on distilling the physical structure from the diagram layout using physical knowledge. Operationally, we apply physical rules to the diagram layouts to derive the intrinsic physical model, which contains only abstract physical objects and the functional relationships between them. For example, in circuit diagrams the physical structure can be formulated using a netlist model (Nagel, 1975), which includes all components along with their types, parameters, and topological connections. In mechanical scenarios, the physical structure can be described using a FEM (Rao, 2010) model to represent the mechanical system. Eventually, the physical structure is formatted into simulation language.

This process produces both a physical diagram and its corresponding simulation language description. Since each step of the generation procedure involves random sampling, a large number of diagram with different objects, spatial relationships and annotations can be generated through sufficient sampling.

3.2.2 PPM Training

The training goal of the Physics Perception Model (PPM) is to generate the corresponding SL description from a given diagram. We use a decoder-only pre-trained visual language model as the base model for the PPM. In practice, the training loss during PPM fine-tuning is the average negative Maximum Likelihood Estimation (MLE) loss (Bishop & Nasrabadi, 2006) over the synthetic data.

4 Experiments

We evaluate our MAPS framework through extensive experimentations on real-world scientific problems. Given the substantial workload involved in constructing and validating the entire pipeline, we have limited our initial verification to the circuit analysis scenario, which is generally believed very difficult for state-of-the-art MLLMs (Yue et al., 2023).

4.1 IMPLEMENTATION

In this section, we describe our implementation of MAPS framework in the circuit analysis scenario.

Synthesis of Training Data for PPM. In the context of circuit diagrams, the diagram layout is defined as the planar grid and components connected between the grid nodes, with the values or labels annotated alongside the component symbols. The grid structures of synthetic data are randomly sampled from a predefined hierarchical distribution, ensuring the diversity of shapes, components, and annotations in the generated circuits. We use CircuitTikz as our plotting language to draw the circuit diagram using a LaTeX compiler. Since the component annotations in the real-world diagrams can be in a numerical format (e.g. 10Ω) or a label format (e.g. R_1), we generate two types of circuit diagrams to cope with this variation accordingly: (1) Numerical-type circuit, where the value is annotated on the diagram. The PPM is required to infer both the type and value of the components; (2) Label-type circuit, where only the labels of the components are provided in the diagram. The PPM predicts the type and label of the components, with an `<Empty>` token in the value position.

The physical structure of a circuit diagram can be represented using by a netlist model (Nagel, 1975; Tao et al., 2024), which is a directed graph where each node represents an equipotential point, and each edge represents a circuit component. The SL synthesis step involves writing rules to automatically identify equivalent circuit nodes using basic physical properties and converting grid information into a netlist. We utilize SPICE (Nagel, 1975) as our simulation language for circuit analysis problems. The syntax of SPICE is based on a netlist model, allowing us directly translated the netlist model into SPICE (Nagel, 1975) program at the end of each generation process. Please refer to Appendix B.1 for our design of the hierarchical distribution and a detailed illustration of the synthesis process.

We name our synthetic data `ppm-syn-lprc`, as our current data synthesis process only supports the generation of Linear Pure Resistive Circuits (LPRC) (Svoboda & Dorf, 2013). `ppm-syn-lprc` contains 20k pairs of synthetic circuit diagrams and their simulation descriptions, divided into training, validation, and test sets in a ratio of 8:1:1.

PPM Training. For the training of PPM, we adopt CogVLM-17B (Wang et al., 2023a) as the base model of PPM. The PPM is fine-tuned to generate the SPICE description for given the circuit diagram. For our detailed settings, please refer to Appendix B. Based on our preliminary experiments, the base model is largely unable to accurately perform the conversion task for most circuit diagrams when using prompting methods. Therefore, the training stage is essential for the development of the MAPS pipeline.

Inference. In our main experiments, we use GPT-4V as our MLLM and NgSPICE (Nenzi & Vogt, 2011) as our physical simulator to execute circuit simulation. Given a circuit analysis problem with a diagram and textual description, our framework infers the answer to the problem following the process described in Algorithm 1. For more implementation details, please refer to Appendix A.

Evaluation Dataset. To evaluate the entire MAPS framework on real-world physical problems, we collected 79 high-quality circuit analysis problems from related textbooks and name it `SimpleCircuitEval`. `SimpleCircuitEval` is constrcuted based on exercise problems primarily collected Chinese circuit analysis text books, but since current MLLMs are primarily multilingual and the linguistic type is not an influencing factor in our framework, this should not affect the evaluation of different MLLMs on this dataset. As each question in `SimpleCircuitEval` has an exact golden answer, we can directly compare the answer produced by the candidate model with the golden answer to compute the accuracy. For a fair evaluation of our proposed solution framework, we only retained questions that involve LPRC type questions, which are covered in the first four chapters of the textbook.

4.2 EVALUATION OF PPM

We first assess the quality of PPM in translating circuit diagram into SPICE language. We adopt 3 metrics to measure its quality:

Component Quantity Accuracy (ACC_{CQ}). This metric measures the accuracy of PMM's prediction in terms of the number of circuit components. The prediction is marked as correct only when the number of different types of components are all correct. This measures the object recognition

quality of PPM and is a necessary condition for correct conversion from a circuit diagram to its simulation language description.

Component Value Accuracy (ACC_{CV}). Based on ACC_{CQ}, ACC_{CV} requires the model to predict the correct value of each component. This is also a necessary condition and is only applicable for Numerical-Type Circuits. ACC_{CV} reflects both the object recognition quality for circuit components and the PPM's ability to recognize numerical values in the diagram.

Simulation Accuracy (ACC_{sim}). This metric measures correctness of PPM's conversion results by comparing the consistency of simulation results between the generated SPICE code and the label code. Although ACC_{CQ} is a necessary condition for PPM to be useful in MAPS, in practice, achieving the same simulation results indicates the same physical circuit with high probability. For the specific examples of these metrics, please refer to Appendix B.2.

We first evaluate PPM on the test split of `ppm-syn-lprc`. Then, we integrate PPM into the inference framework for further evaluation on real-world diagrams. The evaluation result of PPM is shown on Table 1. Through training, our PPM can successfully convert most of the synthetic diagrams. For the conversion of real-world schematics, our PPM only has around 50% simulation accuracy, which leaves a big room for further improvement. We provide more in-depth discussions about PPM in Section 5.2.

Table 1: Conversion efficacy of PPM on synthetic dataset `ppm-syn-lprc-test` and 20 diagrams on real-world dataset `SimpleCircuitEval`. "Num." indicates Numerical-Type diagram while "Lab." indicates Label-Type.

Metrics	`ppm-syn-lprc-test`		`SimpleCircuitEval`	
	Num.	Lab.	Num.	Lab.
$ACC_{CQ}\uparrow$	99.2	98.5	87.0	80.0
$ACC_{CV}\uparrow$	95.5	-	87.0	-
$ACC_{sim}\uparrow$	85.4	-	53.3	-

4.3 Evaluation of MAPS Framework

To verify the effectiveness of the MAPS framework, we implemented it using existing advanced MLLMs, including GPT-4V (Achiam et al., 2023), Claude-3.5 (Anthropic, 2024) and GLM-4V (GLM et al., 2024). We compared our method with directly prompting these MLLMs to generate the results. Additionally, we implemented the Multimodal-CoT (Zhang et al., 2023b), which prompts the model to generate detailed descriptions and analyses of the given circuit diagram and then infer the result based on the generated multi-modal thought for comparison.

Our main results are reported in Table 2, which demonstrates that MAPS significantly improves the MLLM's multi-modal reasoning capability on circuit-analysis problems and help it outperform existing models and methods. For example, the state-of-the-art GPT-4V only achieved less than 7.6% accuracy on the real-world circuit analysis problems, while our solution raised bar more than 3 times to 32.9%. Through our case studies, we found MAPS effectively alleviates the issues on physical diagrams understanding and complex mathematical reasoning of current MLLM mentioned in Section 2.

We found that our framework and baseline methods all fail at solving problems collected from the Chapter 2 of the textbook, which mainly focuses on the Equivalent Transformation of Resistance and mostly cover the circuits that could not be directly executed in a simulator. When the problem is not simulatable, the MLLM can leverage the additional information from simulation language to reason the final answer. Please refer to our Appendix C.2 for more specific case studies. However, how to improve MLLM's general scientific reasoning ability through interaction with a physical simulator is still a challenging problem and remains for our future work.

5 Analysis

5.1 Analysis on Inference Phase Design of MAPS

We perform in-depth analysis of our framework and investigate the contribution of its different components. Our ablation study was performed using a sample of 20 randomly selected problems from `SimpleCircuitEval`. We analyze our MAPS framework by answering a series of questions.

Q: Can we directly prompt MLLM to generate simulation language descriptions of given circuit diagrams, instead of training the expert model PPM?

Table 2: Evaluation results of MAPS and other baselines on `SimpleCircuitEval`. MAPS significantly surpass existing models and method on complex circuit analysis problems.

	↑ **Acc.(%)**				
Chapter	Chap1	Chap2	Chap3	Chap4	All
#Problem	25	24	19	11	79
GPT-4V	16.0	4.17	5.26	9.09	7.6
GPT-4V + MMCoT	8.0	4.17	5.26	0.0	5.1
GPT-4V + MAPS (**Ours**)	52.0	7.14	36.8	45.5	32.9(×4.3)
Claude-3.5	16.0	4.17	0.0	0.0	6.33
Claude-3.5 + MAPS (**Ours**)	44.0	12.5	21.1	36.4	25.3(×4.0)
GLM-4V	8.0	4.17	5.26	0.0	6.33
GLM-4V + MAPS (**Ours**)	32.0	0.0	15.8	36.4	19.0(×3.0)
Gemini-1.5	8.0	12.5	0.0	0.0	6.33
GPT-4o	20.0	4.17	5.26	9.09	10.1

A: Despite being pre-trained on large-scale corpora, we found that even the most advanced MLLMs, such as GPT-4V, often struggle to generate accurate simulation language descriptions for relatively complex circuit diagrams. Specifically, GPT-4V refused to generate SPICE code in 8 out of 10 instances in our evaluation set.

Q: Is the simulator necessary for MAPS framework?

A: We use ablation analysis to answer this question and report the results in Table 3. We found that MAPS does not work without the assistance of the simulator when we remove the simulation results from the final query (i.e., **MAPS w.o. Simulator**). We also verified the necessity of simulator by prompting MLLM to write Python programs to infer the answer (Chen et al., 2022) given the simulation language and problem description (i.e., **MAPS w.o. Simulator + PoT**). Notably, **MAPS w.o. Simulator** and **MAPS w.o. Simulator + PoT** both achieved only 15% accuracy on the evaluation set. This underscores the importance of incorporating a professional simulator when addressing problems with complex physical backgrounds.

Table 3: Ablation study of MAPS framework on Problems sampled from `SimpleCircuitEval`. The results show a high reliance of the MAPS framework on the physical simulator.

Method	↑ **Acc.(%)**
MAPS (**Ours**)	**55.0**
MAPS w.o. SL	45.0
MAPS w.o. Simulator	15.0
MAPS w.o. Simulator + PoT	15.0

Q: Is the simulation language description helpful for the final reasoning?

A: We found that when the problem is not simulatable, the simulation language can still be helpful to the final reasoning of framework. The structural information provided by the simulation language significant reduces hallucination of MLLM when understanding the diagram, akin to the role of scene graph in general multi-modal reasoning (Mitra et al., 2024). Appendix C.2 presents a detailed example showing how simulation description in MAPS alleviate the MLLM's hallucination problem when the physical scene is not simulatable.

We also investigate whether the simulation language description is necessary to simulation-aided reasoning step when simulation results are given, denoted as **MAPS w.o. SL** in Table 3. The result shows that the SL plays a vital role in final reasoning even when the simulation results are given, bridging the gap between the diagram information and numerical simulation results.

5.2 Analysis of PPM Construction

Philosophy of PPM Construction. Although we focus solely on circuit disciplines in our evaluation, the philosophy of constructing a PPM is universal across all physical disciplines. The target of PPM is to convert a physical diagram to its **formal** and **simulatable** language description, which requires paired training data in the form of physical diagrams and corresponding simulation language descriptions.

Since there is no available open-source data in such a format and human annotations on a large corpus is quite costly, we devised an automated data synthesis solution to enhance the VLM's per-

ception ability on real-world diagrams. The assumption behind our data synthesis pipeline is that *the potential space of physical diagrams can be effectively covered by a human-designed distribution*. Physical diagrams are often composed of dots, lines, and symbols with specific physical meanings, and are primarily designed to abstract real-world scenarios. By distilling the core patterns of these diagrams, we can establish a distribution to generate representative training data for PPM. For example, in circuit diagrams, we have observed that most inputs are formed with planar grids, with components placed at the edges of these grids. In mechanical diagrams, the pattern could be composition and positional relationship of mechanical objects (pole, ball, box etc.). Since our work is exploratory, designing a universal generator for physical diagrams or obtaining a comprehensive physical perception model remains an open problem.

Using VLM to implement PPM. Converting a physical diagram into its simulation language description can be viewed as a comprehensive vision task which involves the recognition of physical objects and the OCR of its detached labels, along with the complex topology about the components' connection. In terms of circuit schematics, some previous works (Bayer et al., 2023; Bailey et al., 1995; Tao et al., 2024) investigate multi-step process to convert a pixel-level circuit into digital structure, but their methods are expensive to implement and not scalable to diagrams of new styles.

By using VLM as our perception model, we obtain an end-to-end physical diagram recognition solution whose capability can be extended through expanding the data distribution during training. Besides, we also observe that pre-trained VLMs exhibit promising generalization ability after training on our synthetic data, e.g., its OCR ability on float number although our synthetic data only contains integer values.

Scaling the conversion task. Through a development set based on our synthetic Numerical-Type Circuits, we also found that the conversion accuracy (ACC_{sim}) decreases as circuit's complexity increases. Figure 3 illustrates that as the number of nodes and components increases in our synthetic data, the simulation accuracy of PPM's predictions shows a downward trend. This result is intuitive since the smaller circuits with simpler physical structures show higher accuracy during test.

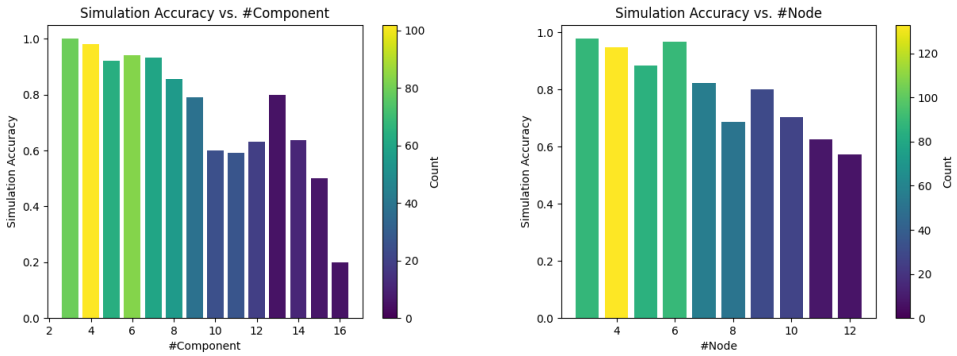


Figure 3: With the increase in circuit scale—specifically the number of components and electrical nodes—the accuracy of PPM decreases. The colorbar display the sample amount in each scale.

6 Related Work

Improving Multi-modal Reasoning Ability of MLLM. Reasoning ability is foundational for building agent that assist human to solve complex real-world tasks. There are many studies focusing on improving the reasoning ablility of MLLMs. There have been three main directions to boost the reasoning capability of MLLMs, including instruction-tuning, prompt engineering and tool use (Wang et al., 2024c). As instruction tuning requires high-quality multi-modal training corpus which is scare in scientific domains, most studies focus on how to improve the scientific reasoning ability of MLLMs via prompting methods and tool utilization.

The multi-modal prompting methods involve designing effective prompts to fully activate the model's image understanding and language reasoning abilities based on the carefully crafted text instructions. Specifically, existing methods (Zhang et al., 2023b; Mitra et al., 2024; Zheng et al., 2023; Zhou et al., 2024; Zhong et al., 2024; Yang et al., 2023) focus on how to enable the model to generate intermediate rationales, or chain-of-thought (CoT) (Wei et al., 2022), for parsing image

information and then further reason based on these intermediate results (Zhang et al., 2023b). Some variants of Multimodal-CoT focus on the form of CoT, for example, Zheng et al. (2023) format the CoT as a decomposition of original problems and use the answers of sub-problems to generate the result, while Mitra et al. (2024) adopt Scene Graph (SG) as the rationale to assist the inference of final answer.

On the other hand, integrating LLMs with external tools exhibits significant improvement in scientific reasoning (Chen et al., 2022; Gou et al., 2023). For the reasoning problems containing data from different modalities, the category of available tools extends beyond traditional external software (e.g., calculators, calendars, search engines, code execution environments) to specialized vision models (e.g., object detection models, OCR models, image semantic segmentation models, etc.). Research in this direction seeks to build useful tool sets and design mechanisms for MLLMs to interact with these tools, thereby completing designated reasoning tasks more effectively (Liu et al., 2023; Wang et al., 2024a; Gao et al., 2023).

Multi-Modal Agent in Scientific Scenarios. A multi-modal AI agent is a system designed to realize users' general-purpose requirements by perceiving its environment with multi-modal information and making decisions based on its observations (Xie et al., 2024). Multi-modal agents have been developed for many important domains, including GUI automation (Gur et al., 2024; Wen et al., 2024; Wang et al., 2024b), embodied AI (Qin et al., 2024; Wang et al., 2023b) and general understanding, generation and editing of images, videos and audios (Wu et al., 2023; Gao et al., 2023; Liu et al., 2023; Yang et al., 2023). For example, Gur et al. (2024) devise an LLM-based agent that learns from interactive experiences to follow human instructions and complete tasks on real websites, such as click, type or making selection.

To construct multi-modal agents in scientific domains, an important research direction involves how to perceive multi-modal information encompassing diverse scientific concepts. With the development of LLMs, a lot of efforts have been made to build multi-modal foundation models tailored for specific scientific scenarios. These models are capable of perceiving or generating chemical formulas, protein sequences, geographical information, graphs, and more (Luo et al., 2022; Li et al., 2023; Frey et al., 2023; Jiang et al., 2023; Zhang et al., 2023a). However, most previous works focus solely on multi-modal information in text format, neglecting the pixel-format information of physical diagrams that are prevalent in human knowledge bases.

7 Discussion & Conclusion

In this work, we introduce the MAPS framework to address the inability of existing MLLMs in understanding complex physical diagrams and to solve such problems analytically. Our framework, which trains a Physics Perception Model (PPM) to interpret physical diagrams and applies Chain-of-Simulation and Simulation-Aided Reasoning during inference, successfully solves the circuits analysis problem, a typical and important type of real-world physical problems.

MAPS excels in deriving final answers when the physical scenario is directly simulatable. However, a key limitation is its static workflow, which lacks feedback interaction with the physical simulator. To address this, a dynamic workflow where the simulator acts as an external environment for feedback is necessary. In this setting, PPM still serves an important role in connecting multi-modal information with the physical simulator. This improvement would significantly enhance the versatility of our physical agent and is an important focus for future work.

As the first attempt of this kind, this work only tested MAPS on LPRC circuit analysis problems. Extending MAPS to other scientific disciplines with complex illustrative schematics is an important next step. It requires developing a universal and accurate PPM for the Chain-of-Simulation. This is a challenging task in computer vision that remains for future work. Additionally, simulators are currently domain-specific, making effective organization across simulators of different domains or the development of a universal simulator crucial for MAPS's broader application.

As shown in our experiment results, our work presents a solid path towards building multi-modal agents capable of solving expert-level scientific problems, contributing to the progress towards achieving AGI in scientific domains.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Anthropic AI. Introducing the next generation of Claude — anthropic.com. https://www.anthropic.com/news/claude-3-family. [Accessed 04-06-2024].

Anthropic. Introducing Claude 3.5 Sonnet, 2024. URL https://www.anthropic.com/news/claude-3-5-sonnet. [Accessed 01-10-2024].

Donald Bailey, Andrew Norman, Giovanni Moretti, and P North. Electronic schematic recognition. *Massey University, Wellington, New Zealand*, 1995.

Johannes Bayer, Shabi Haider Turabi, and Andreas Dengel. Text extraction for handwritten circuit diagram images. In Mickael Coustaty and Alicia Fornés (eds.), *Document Analysis and Recognition – ICDAR 2023 Workshops*, pp. 192–198, Cham, 2023. Springer Nature Switzerland. ISBN 978-3-031-41498-5.

Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*, 2022.

Nathan C Frey, Ryan Soklaski, Simon Axelrod, Siddharth Samsi, Rafael Gomez-Bombarelli, Connor W Coley, and Vijay Gadepally. Neural scaling of deep chemical models. *Nature Machine Intelligence*, 5(11):1297–1305, 2023.

Difei Gao, Lei Ji, Luowei Zhou, Kevin Qinghong Lin, Joya Chen, Zihan Fan, and Mike Zheng Shou. Assistgpt: A general multi-modal assistant that can plan, execute, inspect, and learn. *arXiv preprint arXiv:2306.08640*, 2023.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, et al. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*, 2024.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yujiu Yang, Minlie Huang, Nan Duan, Weizhu Chen, et al. Tora: A tool-integrated reasoning agent for mathematical problem solving. *arXiv preprint arXiv:2309.17452*, 2023.

Izzeddin Gur, Hiroki Furuta, Austin V Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. A real-world webagent with planning, long context understanding, and program synthesis. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=9JQtrumvg8.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. Structgpt: A general framework for large language model to reason over structured data. *arXiv preprint arXiv:2305.09645*, 2023.

PC Kohnke. Ansys. In *Finite Element Systems: A Handbook*, pp. 19–25. Springer, 1982.

Milton Laikin. *Lens design*. Crc Press, 2018.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pp. 12888–12900. PMLR, 2022.

Yuesen Li, Chengyi Gao, Xin Song, Xiangyu Wang, Yungang Xu, and Suxia Han. Druggpt: A gpt-based strategy for designing potential ligands targeting specific proteins. *bioRxiv*, pp. 2023–06, 2023.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36, 2024.

Shilong Liu, Hao Cheng, Haotian Liu, Hao Zhang, Feng Li, Tianhe Ren, Xueyan Zou, Jianwei Yang, Hang Su, Jun Zhu, et al. Llava-plus: Learning to use tools for creating multimodal agents. *arXiv preprint arXiv:2311.05437*, 2023.

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in Neural Information Processing Systems*, 35:2507–2521, 2022.

Renqian Luo, Liai Sun, Yingce Xia, Tao Qin, Sheng Zhang, Hoifung Poon, and Tie-Yan Liu. Biogpt: generative pre-trained transformer for biomedical text generation and mining. *Briefings in bioinformatics*, 23(6):bbac409, 2022.

Chancharik Mitra, Brandon Huang, Trevor Darrell, and Roei Herzig. Compositional chain-of-thought prompting for large multimodal models, 2024.

Laurence W Nagel. Spice2: A computer program to simulate semiconductor circuits. *College of Engineering, University of California, Berkeley*, 1975.

Paolo Nenzi and Holger Vogt. Ngspice users manual version 23. *Experiments/ngspice23-manual.pdf*, 2011.

Yiran Qin, Enshen Zhou, Qichang Liu, Zhenfei Yin, Lu Sheng, Ruimao Zhang, Yu Qiao, and Jing Shao. Mp5: A multi-modal open-ended embodied system in minecraft via active perception, 2024. URL https://arxiv.org/abs/2312.07472.

Singiresu S Rao. *The finite element method in engineering*. Elsevier, 2010.

James A Svoboda and Richard C Dorf. *Introduction to electric circuits*. John Wiley & Sons, 2013.

Zhuofu Tao, Yichen Shi, Yiru Huo, Rui Ye, Zonghang Li, Li Huang, Chen Wu, Na Bai, Zhiping Yu, Ting-Jung Lin, and Lei He. Amsnet: Netlist dataset for ams circuits, 2024.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Minyang Tian, Luyu Gao, Shizhuo Dylan Zhang, Xinan Chen, Cunwei Fan, Xuefei Guo, Roland Haas, Pan Ji, Kittithat Krongchon, Yao Li, et al. Scicode: A research coding benchmark curated by scientists. *arXiv preprint arXiv:2407.13168*, 2024.

Chenyu Wang, Weixin Luo, Qianyu Chen, Haonan Mai, Jindi Guo, Sixun Dong, Zhengxin Li, Lin Ma, Shenghua Gao, et al. Tool-lmm: A large multi-modal model for tool agent learning. *arXiv preprint arXiv:2401.10727*, 2024a.

Junyang Wang, Haiyang Xu, Jiabo Ye, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. Mobile-agent: Autonomous multi-modal mobile device agent with visual perception, 2024b. URL https://arxiv.org/abs/2401.16158.

Weihan Wang, Qingsong Lv, Wenmeng Yu, Wenyi Hong, Ji Qi, Yan Wang, Junhui Ji, Zhuoyi Yang, Lei Zhao, Xixuan Song, et al. Cogvlm: Visual expert for pretrained language models. *arXiv preprint arXiv:2311.03079*, 2023a.

Yiqi Wang, Wentao Chen, Xiaotian Han, Xudong Lin, Haiteng Zhao, Yongfei Liu, Bohan Zhai, Jianbo Yuan, Quanzeng You, and Hongxia Yang. Exploring the reasoning abilities of multimodal large language models (mllms): A comprehensive survey on emerging trends in multimodal reasoning, 2024c. URL https://arxiv.org/abs/2401.06805.

Zihao Wang, Shaofei Cai, Anji Liu, Yonggang Jin, Jinbing Hou, Bowei Zhang, Haowei Lin, Zhaofeng He, Zilong Zheng, Yaodong Yang, Xiaojian Ma, and Yitao Liang. Jarvis-1: Open-world multi-task agents with memory-augmented multimodal language models, 2023b. URL https://arxiv.org/abs/2311.05997.

Irene Weber. Large language models as software components: A taxonomy for llm-integrated applications. *arXiv preprint arXiv:2406.10300*, 2024.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Hao Wen, Hongming Wang, Jiaxuan Liu, and Yuanchun Li. Droidbot-gpt: Gpt-powered ui automation for android, 2024. URL https://arxiv.org/abs/2304.07061.

Douglas Brent West et al. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River, 2001.

Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. Visual chatgpt: Talking, drawing and editing with visual foundation models, 2023. URL https://arxiv.org/abs/2303.04671.

Junlin Xie, Zhihong Chen, Ruifei Zhang, Xiang Wan, and Guanbin Li. Large multimodal agents: A survey, 2024. URL https://arxiv.org/abs/2402.15116.

Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. Mm-react: Prompting chatgpt for multimodal reasoning and action. *arXiv preprint arXiv:2303.11381*, 2023.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. *arXiv preprint arXiv:2311.16502*, 2023.

Yifan Zhang, Cheng Wei, Shangyou Wu, Zhengting He, and Wenhao Yu. Geogpt: Understanding and processing geospatial tasks through an autonomous gpt, 2023a. URL https://arxiv.org/abs/2307.07930.

Zhuosheng Zhang, Aston Zhang, Mu Li, Hai Zhao, George Karypis, and Alex Smola. Multimodal chain-of-thought reasoning in language models. *arXiv preprint arXiv:2302.00923*, 2023b.

Ge Zheng, Bin Yang, Jiajin Tang, Hong-Yu Zhou, and Sibei Yang. Ddcot: Duty-distinct chain-of-thought prompting for multimodal reasoning in language models. *Advances in Neural Information Processing Systems*, 36:5168–5191, 2023.

Yiwu Zhong, Zi-Yuan Hu, Michael R Lyu, and Liwei Wang. Beyond embeddings: The promise of visual table in multi-modal models. *arXiv preprint arXiv:2403.18252*, 2024.

Aojun Zhou, Ke Wang, Zimu Lu, Weikang Shi, Sichun Luo, Zipeng Qin, Shaoqing Lu, Anya Jia, Linqi Song, Mingjie Zhan, and Hongsheng Li. Solving challenging math word problems using gpt-4 code interpreter with code-based self-verification, 2023. URL https://arxiv.org/abs/2308.07921.

Qiji Zhou, Ruochen Zhou, Zike Hu, Panzhong Lu, Siyang Gao, and Yue Zhang. Image-of-thought prompting for visual reasoning refinement in multimodal large language models, 2024.

A Additional Implementation Details of MAPS Inference Phase

A.1 Simulation

For the simulation of circuit problems, we use NgSPICE (Nenzi & Vogt, 2011) developed by the UC Berkeley CAD Group as our simulator. The core arguments we set for the simulation are listing on Table 4.

Table 4: Parameters of NgSPICE simulator

Param.	Setting
Temperature	27°
Nominal Temperature	27°

We store the simulation results in `dictionary` format, which is a commonly used data structure in programming as well as the conversation with MLLM (Mitra et al., 2024; Weber, 2024). Figure 4 shows an example of the post-processing of simulation result.

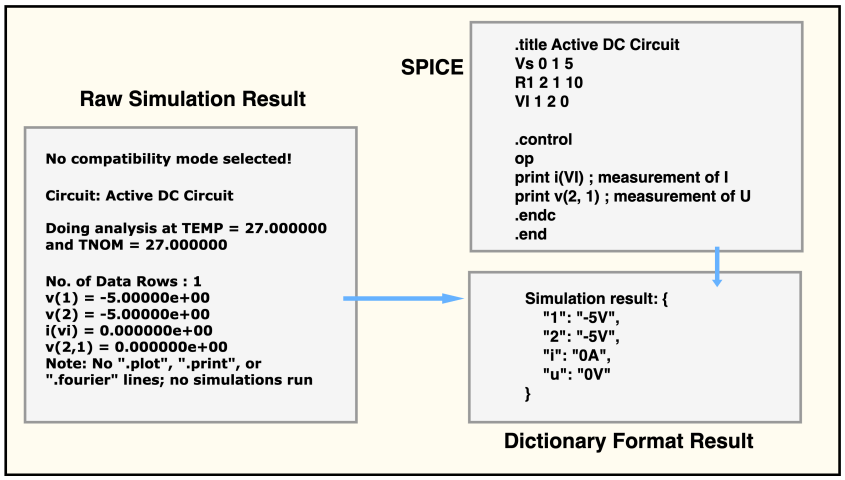


Figure 4: Post-processing of simulation results.

A.2 Prompt Templates

In this section, we will showcase the prompt templates that we used at Inference Stage.

At the Chain-of-Simulation step of MAPS inference, since our training PPM is merely an image-to-text model, the component values of circuit in textual description is merged to the simulation language by MLLM in Refine process.

The prompt we used for this process is shown at Figure 5 . This prompt will only be applied when we detect `<Empty>` token in generated simulation language of PPM, which is a special token design for the component with missing value in the diagram.

In the Simulation-Aided Reasoning(SAR) step, the MLLM infers the answer based on the information provided by Chain-of-Simulation. Figure 6 shows the prompt template used for SAR in circuit disciplines.

If the simulation results are not obtained (due to incorrect simulation language) in SAR step, we use a special prompt that allows the MLLM to infer the final result based on the information provided in the problem and the simulation language. Figure 7 shows the special prompt.

Figure 8 shows the prompt template that we prompt MLLM to directly infer the answer. Since current MLLMs have been trained on CoT data, they will apply an automated CoT to infer the answer.

Prompt Template - MAPS Inference - CoS - Refine

You are an expert in circuit principles and need to solve a circuit problem, consisting of a circuit diagram and a question. We have converted the circuit diagram into a SPICE netlist language using a circuit recognition model. However, some values might be missing (marked as <Empty>) in the netlist of the circuit diagram. You need to complete the SPICE code based on the information provided in the problem, such as filling in the resistance values from the question into the netlist. Your output should be the completed SPICE code, and please place your answer between ```SPICE and ```.

Question
{question}{subq}

SPICE Netlist Converted from Circuit Diagram
{spice}

Figure 5: Prompt template of refine process at Chain-of-Simulation step

Prompt Template - MAPS Inference - SAR

You are an expert in circuit principles and need to solve a circuit problem, consisting of a circuit diagram and a question. We have converted the circuit diagram into a SPICE netlist language using a circuit recognition model. Meanwhile, we have imported the SPICE code into circuit simulation software and obtained simulation results. Based on the circuit diagram, the question, the SPICE netlist, and the simulation results, please infer the answer to the problem.

Please represent the final numerical value or expression in JSON format as follows:
```json
{
    "Value1": "Result1",
    "Value2": "Result2"
}
```
If the simulation results directly contain the answer, you should output the answer obtained from the simulation results. If the simulation results show anomalies or do not contain the answer to the question, you need to further infer the final result based on the netlist and simulation results.

Question
{question}{subq}

SPICE Netlist Converted from Circuit Diagram
{spice}

Simulation Results
{sim_ret}

Figure 6: Prompt template of Simulation-Aided Reasoning step

Prompt Template - MAPS Inference - SAR w.o. simulation result

You are an expert in circuit principles and need to solve a circuit problem, consisting of a circuit diagram and a question. We have converted the circuit diagram into SPICE netlist language using a circuit recognition model. Based on the question, the circuit diagram, and the SPICE netlist, please infer the answer to the problem.

Please represent the final numerical value or expression in JSON format as follows:
```json
{
    "Value1": "Result1",
    "Value2": "Result2"
}
```

Question
{question}{subq}

SPICE Netlist Converted from Circuit Diagram
{spice}

Figure 7: Prompt template of Simulation-Aided Reasoning step (No Simulation Result)

Prompt Template - MLLM Direct

```
Please solve the following circuit problem, and represent the final numerical values or expressions
in JSON format, formatted as:
```json
{
    "Value1": "Result1",
    "Value2": "Result2"
}
```

Question
{question}{subq}

Accompanying Diagram
{diagram}
```

Figure 8: Prompt template of MLLM directly inference

MMCoT (Zhang et al., 2023b) decomposes the multi-modal reasoning process into a two step paradigm: Rationale Genetraion and Answer Inference. Following the similar idea, we define the rationale in our setting as the language description of the physical diagrams. Figure 9 and 10 display our prompt templates for the two step generation.

Prompt Template - MLLM + MMCoT Step 1

```
You will receive a circuit problem with a circuit diagram. You need to generate a descriptive language
for the circuit diagram to facilitate further inference. {extra_prompt}

# Question
{question}{subq}

# Accompanying Diagram
{diagram}
```

Figure 9: Prompt Template for Step 1 of MMCoT

Prompt Template - MLLM + MMCoT Step 2

```
You need to solve a circuit problem with a circuit diagram, and you can refer to the descriptive
language of the circuit diagram. Please represent the final numerical values or expressions in JSON
format, formatted as:
```json
{
    "Value1": "Result1",
    "Value2": "Result2"
}
```

Question
{question}{subq}

Circuit Diagram Descriptive Language
{desc}

Accompanying Diagram
{diagram}
```

Figure 10: Prompt Template for Step 2 of MMCoT

B Additional Implementation Details of PPM Construction Phase

In this section, we will delve into the implementation details involved in the construction phase of the Physics Perception Model (PPM).

B.1 Data Synthesis

The data synthetic pipeline has been shown in the left side of Figure 2(a). We have introduced the general process of data generation in Section 3.2.1. In this section, we will introduce our synthesis pipeline in circuit discipline with a specific example.

In the first step, we sample an diagram layout from the manual distribution. As discussed in Section 5.2, the key property of our designed generation distribution is to cover the distribution of real-world diagrams as comprehensively as possible.

Our implementation of diagram layout sampling in synthesizing PPM's training data for Linear Pure Resistive Circuit (LPRC) (Svoboda & Dorf, 2013) diagrams is shown in Algorithm 2, where $\mathcal{D}, \mathcal{U}$ in the pseudo code represent Discrete Probability Distribution and Uniform Distribution respectively. We only show our main idea in the pseudo code due to its tedium. Since we use a hierarchical sampling process, we can sample diverse circuits with different shapes, components and annotations. The hyperparameters of the sampling process are set by human experiences.

Algorithm 2 Diagram Layout Sampling for LPRC

Input: $d_{\text{max}}, d_{\text{min}}, \vec{n}, \vec{p_n}, \vec{t}, \vec{p_t}, \cdots$
Output: Diagram Layout $\mathcal{I}$
`% Determine the scale of the grid`
number of grid: $(m \times n)$: $m, n \sim \mathcal{D}(\vec{n}, \vec{p_n})$
horizontal scale: $\vec{d^h} = (d_1^h, \cdots, d_n^h)$ where $d_i^h \sim \mathcal{U}(d_{\min}, d_{\max})$
vertical scale: $\vec{d^v} = (d_1^v, \cdots, d_m^v)$ where $d_i^v \sim \mathcal{U}(d_{\min}, d_{\max})$
`% Determine the component's type & direction in each edge`
horizontal component: $T^h = [T_{i,j}^h]_{m\times(n-1)}$ where $T_{i,j}^h \sim \mathcal{D}(\vec{t}, \vec{p_t})$
vertical component: $T^v = [T_{i,j}^v]_{(m-1)\times(n)}$ where $T_{i,j}^v \sim \mathcal{D}(\vec{t}, \vec{p_t})$
direction of horizontal component: $D^h = [D_{i,j}^h]_{m\times(n-1)}$ where $D_{i,j}^h \sim \mathcal{D}((0,1),(0.5,0.5))$
direction of vertical component: $D^v = [D_{i,j}^v]_{(m-1)\times n}$ where $D_{i,j}^v \sim \mathcal{D}((0,1),(0.5,0.5))$
`% Determine the component's value & unit in each edge`
$\cdots$
`% Determine the component's label in each edge`
$\cdots$
`% Determine the observation's label & direction in each edge`
$\cdots$
`% Assign the observation label to controlled source`
$\cdots$
$\mathcal{I} = \text{CircuitDiagram}(m, n, \vec{d^h}, \vec{d^v}, ...)$
return $\mathcal{I}$

In our illustrative case, we sampled a 4×4 grid and assign each edge with specific components, as shown in Figure 11.

After the sampling of diagram layout, there are two synthesis paths that respectively generate pixel-format diagram and simulation language(SL) description.

The diagram synthesis path involves converting the grid into LaTeX language that can describe a circuit diagram. We use the LaTeX package `circuitikz` to plot the circuit. After compiling the LaTeX code, a pixel-level circuit diagram can be generated. Each edge in the grid is converted into a line of LaTeX drawing language. The drawing statement includes the start and end positions of

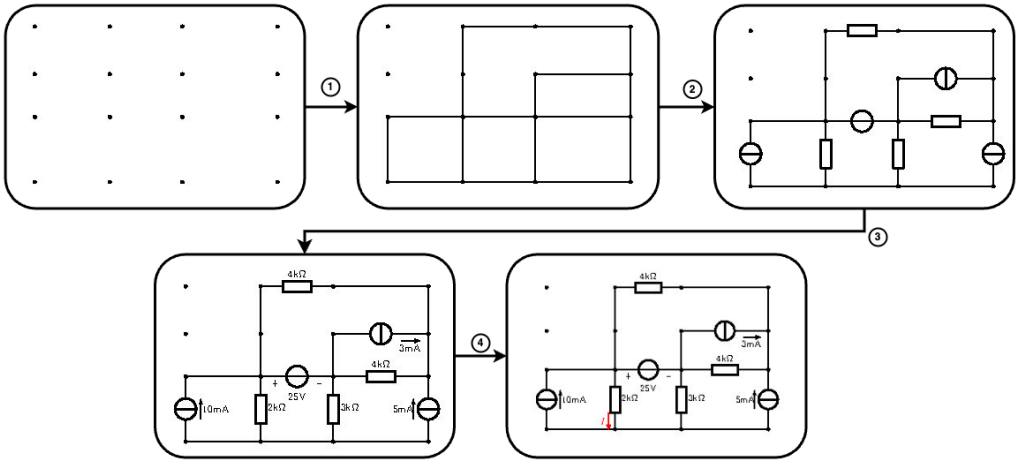


Figure 11: Data Synthesis: Diagram Layout Sampling

the element/wire, the shape of the element, the label of the element (number or string), the type of measurement, and its label.

The SL synthesis path primarily focuses on distilling the physical structure from the diagram layout using human prior knowledge. The physical structure of a circuit can be represented by a netlist (Nagel, 1975; Tao et al., 2024) model, which is a directed graph (West et al., 2001) where each node represents an equipotential point and each edge represents a component. The netlist model includes all components along with their types, parameters, and topological connections of a circuit, while filtering out position and scale noise when plotting the diagram. We write rules to automatically identify equivalent electrical nodes using basic physical properties and convert grid information into a netlist. Figure 12 illustrates the physical structure extraction process of our example case.

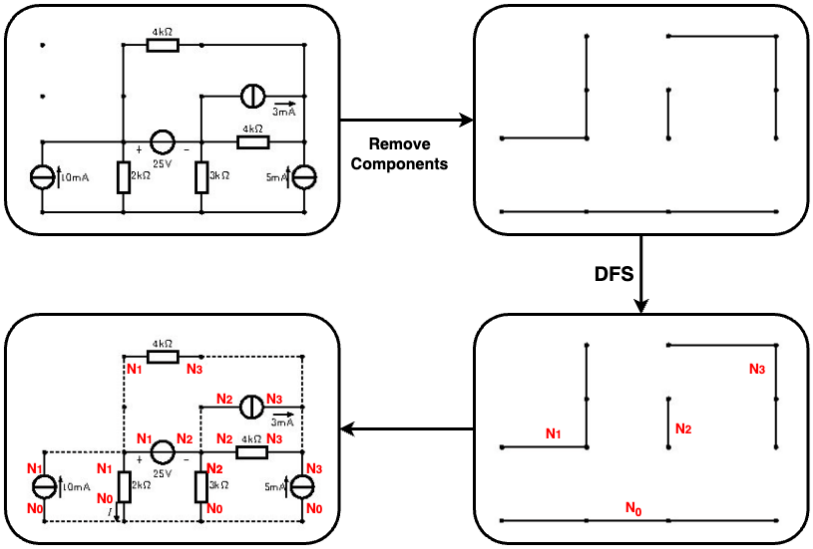


Figure 12: Data Synthesis: Extracting physical structure from diagram layout using physical rules

Finally, the netlist is translated into SPICE code, i.e. our simulation language as the training label of PPM. The SPICE language can be mainly divided into two parts: the first part is the description of circuit elements (Element Card), and the second part consists of control commands that determine the simulation type and output results (Control Card). Since each edge in the netlist corresponds

directly to a circuit element description line in SPICE, the conversion of the first part is merely a formatting process. For the second part, which involves control commands, we set up the simulation for a steady-state analysis of Linear Purely Resistive Circuits(LRPC). This is done by using the .OP (Operating Point) command to define the simulation type as a DC operating point analysis and the .PRINT command to specify the circuit state quantities to be measured.

We also counted the amount of electrical nodes and components in our synthetic dataset `ppm-syn-lprc-test`. The statistic results are shown on Table 5, where "#X" represents the number of objects of type X.

Table 5: Statistics of `ppm-syn-lprc-test`

Parameter	Mean	Std	Max	Min
#Nodes	7.876	3.137	26.0	1.0
#Branches	11.088	5.364	45.0	0.0
#Resistors	6.566	3.452	25.0	0.0
#Voltage Sources	1.340	1.268	9.0	0.0
#Current Sources	1.517	1.340	12.0	0.0
#Controlled Sources	1.411	1.457	11.0	0.0
#Shorts	0.255	0.508	5.0	0.0
#Voltage Measurements	1.192	0.834	7.0	0.0
#Current Measurements	0.503	0.713	6.0	0.0

Figure 13 shows some cases of our synthetic circuit diagrams.

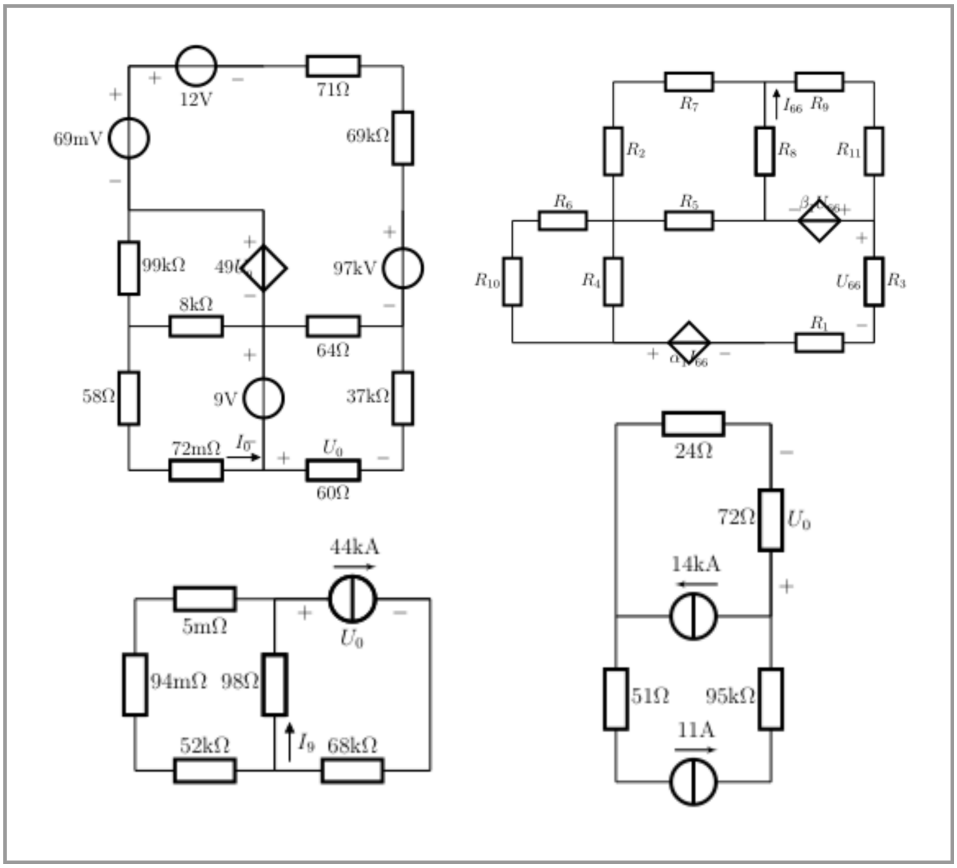


Figure 13: Examples of synthetic diagrams

The synthetic paired data are used for the training of physics perception model.

B.2 PPM TRAINING

We will introduce our PPM training process for our main experiments in detail in this section.

The training objective of PPM is to predict the simulation language given the visual diagram input. Let the diagram input be X_V, and the output text sequence be $Y_L = (y_{L,1}, ..., y_{L,T})^T$, with a length of T. The model parameters are denoted as θ. The probability distribution for predicting the next token under the model can be represented by $p_\theta(y|X_V, Y_{L,1:t})$. The MLE fine-tuning loss can therefore be written in the form $\mathcal{L}_\theta^{MLE}(X_V; Y_L) = -\sum_{t=0}^{T-1} \log p_\theta(y = y_{L,t+1}|X_V, Y_{L,1:t})$, where $Y_{L,1:t} = (y_{L,1}, ..., y_{L,t})^T$.

Let the training dataset be $\mathcal{D} = \{X_V^{(i)}; Y_L^{(i)}\}_{i=1:N}$, containing N samples. The training process involves minimizing the negative MLE loss for all training samples:

$$\theta^* = \min_\theta \sum_{X_V^{(i)}, Y_L^{(i)} \sim \mathcal{D}} \mathcal{L}_\theta^{MLE}(X_V^{(i)}, Y_L^{(i)}) \tag{1}$$

In our experiments, we adopt CogVLM-17B as our base model to train the PPM. The model version for main experiment is `cogagent-vqa-17B`.

We primarily train the visual modules and the image-text cross-attention part, while the parameters of the text generation part remain mostly unchanged. This is because the main challenge of this task lies in image understanding, and the text generation aspect has already been adequately learned through pre-training in the language model part of CogVLM. We control the trainable parameters of the model as follows: the visual encoder, the ViT, the visual multi-layer perceptron and rotary encoding module, the BOI token and the EOI token, resulting in a total of 11.6B parameters that need updating. The remaining parameters are kept freezed.

We employ Low-Rank Adaptation (LoRA) (Hu et al., 2021) as the fine-tuning strategy to train the VLM. Using the LoRA algorithm to train the VLM significantly reduces the number of parameters for which gradients need to be computed, thereby greatly decreasing the memory overhead of training.

We list our main hyperparameters used for PPM training at Table 6.

Table 6: Main Hyper-parameters of PPM Training

Param.	**Setting**
lora-rank	50
max-length	2000
batch-size	32
train-iters	2000
optimizer	Adam
learning-rate	1e-5
lr-decay-style	cosine
warmup	0.2

After the training process, we evaluated the PPM using the metrics introduced in Section 4.2. To illustrate how these metrics work, we present three cases in Figure 14.

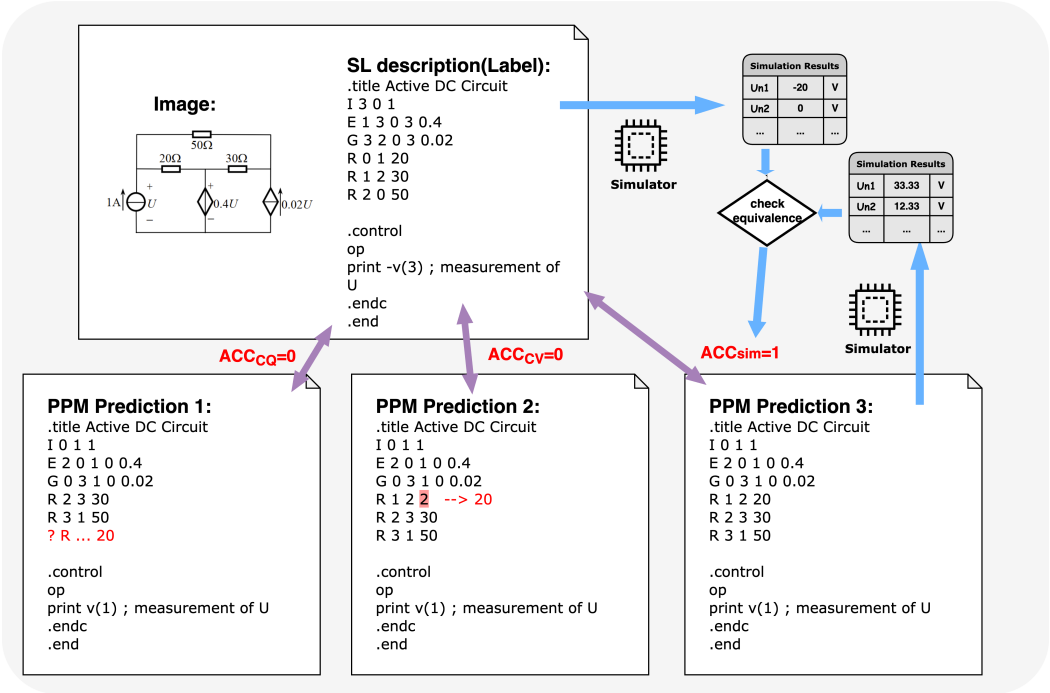


Figure 14: Cases of evaluating PPM.

C Additional Details of Evaluation on SimpleCircuitEval

C.1 Details of SimpleCircuitEval

To evaluate the performance of MAPS on Linear Pure Resistive Circuits (LPRC), we selected questions involving LPRC from the first four chapters of a circuit course textbook. To ensure question diversity and coverage, we consulted domain experts to remove redundant questions, resulting in a final set of 79 questions. The characteristics of the problems from these four chapters and their details are summarized in the table below:

Table 7: Summarization of SimpleCircuitEval.

Chapter	Content	#Components(Avg.)	Characteristics
1	Circuit elements and circuit laws	4.7	Basics circuits. No controlled sources.
2	Analysis method of simple resistance circuit	6.8	Resistance circuits. Not directly simulable, require calculation of equivalent resistance.
3	General method of analysis of linear resistance circuits	7.6	LPRC circuits. Normal topologies.
4	Some theorems of electric circuits	6.2	LPRC circuits. Complex topologies.

In Figure 15 we offer 4 example question for each chapter in SimpleCircuitEval.

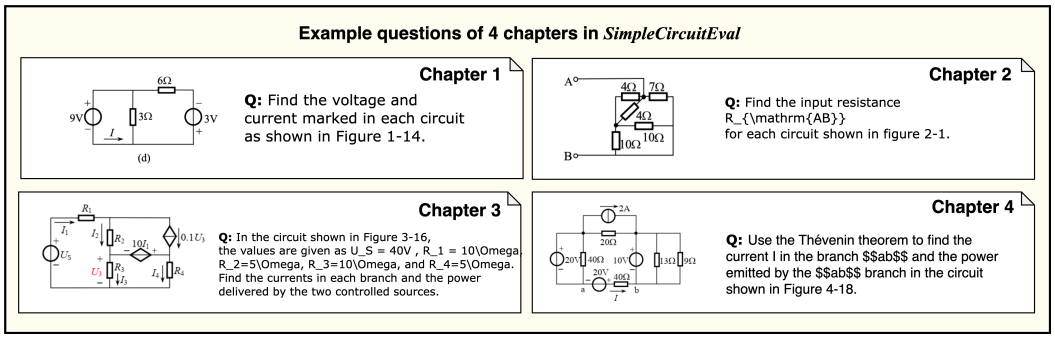


Figure 15: Example questions in `SimpleCircuitEval`.

C.2 Successful Examples

We found simulation language can reduce the hallucination of MLLM on understanding the physical diagram, as shown on Figure 16. These results are consistent with those in a concurrent work (Tao et al., 2024), which uses SPICE descriptions as auxiliary information to guide MLLM's decisions in the context of IC design.

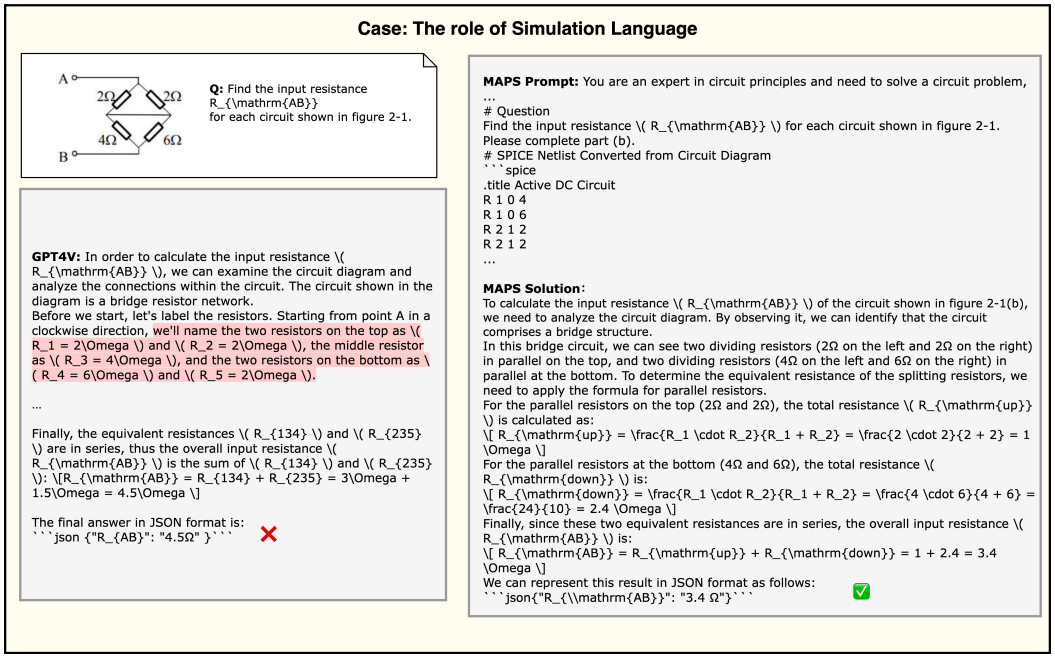
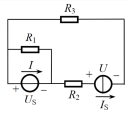


Figure 16: Case Study: The vital role of simulation The positions marked in red are the hallucination positions of the MLLM.

Figure 17 illustrates how MAPS effectively overcomes the challenge of MLLM's inability to comprehend complex physical diagrams by employing formal simulation descriptions and executing precise simulations. In another case, as shown in Figure 18, we found that MAPS successfully addresses the issue of MLLM's inability to perform derivations of complex equations.

Case: MAPS solve the Diagram Understanding Issue in MLLM



Q: In the circuit shown in Figure, $U_S=3V, I_S=1 A, R_1=3\Omega, R_2=1\Omega, R3=2\Omega$. Find the output power P_U and P_I of the voltage source U_S and the current source I_S. .

GPT4V: First, we need to determine the electric power provided by \(U_{\mathrm{S}} \) and \(I_{\mathrm{S}} \). ...

From the given circuit diagram, we know that the current source and voltage source are in parallel, which means that the current flowing through the voltage source \(U_{\mathrm{S}} \) is the same as the current provided by the current source \(I_{\mathrm{S}} \), which is 1A. Therefore, the power output of \(U_{\mathrm{S}} \) is:

\[P_U = U_{\mathrm{S}} \times I_{\mathrm{S}} = 3V \times 1A = 3W \]

...
Because the current source will ensure the current flows through it, the voltage across the current source is determined solely by the resistors \(R_2 \) and \(R_3 \), with the total resistance given by \(R_{\mathrm{total}} = R_2 + R_3 = 1\Omega + 2\Omega = 3\Omega \).

Thus, ...

Therefore, the power output of the current source \(I_{\mathrm{S}} \) is:

\[P_I = I_{\mathrm{S}} \times U = 1A \times 3V = 3W \]

We represent the answers in JSON format as follows:

```json
{
  "P_U": 3,
  "P_I": 3
}
```
❌

MAPS Prompt: You are an expert in circuit principles and need to solve a circuit problem,...

Question
Known pictured above 1-25 circuit, $U_ {\ mathrm {S}} = 3 \ mathrm {n} $, $I_ {\ mathrm {S}} = 1 \ mathrm {A} $, $R_ {1} = \ Omega 3 $, $R_ {2} = 1 \ Omega $, $R_ {3} = 2 \Omega$. For voltage source $U_ {\ mathrm {S}} $and current source $I_ {\ mathrm {S}} the output power of the $$P_ {U} $and $P_ {I} $.

Circuit diagram turned into SPICE nettable

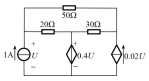
```
.title Active DC Circuit
V1 0 N01 3V
VI N01 1 0
R2 2 1 1ohm
I1 2 3 1A
R1 0 1 3ohm
R3 0 3 2ohm
.control
op
print i(VI) ;  measurement of I
print v(2, 3) ;  measurement of U
.endc
.end
```

Simulation results
{'1': '-3V', '2': '-4V', '3': '2V', 'n01': '-3V', 'i': '0A', 'u': '-6V'}

MAPS Solution:
To understand this problem, we need to use the circuit diagram, SPICE netsheet and simulation results to find the ...
1. The voltage source $U_ {\mathrm {S}} the output power of the $P_{U} $, we use the formula $P = U \ times I $. ...
Thus, $P_{U} = U_{\mathrm{S}} \times I = 3V \times 0A = 0W$.
2. For the output power P_{I} of the current source I_{S}, we also use the formula $P = U \times I$. ...
Thus, $P_{I} = |U| \times I = 6V \times 1A = 6W$.
3. Let's put the answer into the desired json format:
```json
{
 "P_U": 0,
 "P_I": 6}
```
✅

Figure 17: Case Study: MAPS addresses the diagram understanding issue of MLLM.

Case: Simulation Tackles the Derivation Issue in MLLM



Q: Use the loop current method to find the power produced by the 1A current source in the circuit shown in Figure 3-15.
.

GPT4V: To solve this circuit problem and find the power produced by the 1A current source, we need to first determine the unknown voltage U in the circuit. Then we can use the power formula, \(P = IV \), where \(I \) is the current and \(V \) is the voltage, to calculate the power.

We can see that the circuit has ...

Assume that the current through the 50-ohm resistor is \(I1 \), the current through the loop consisting of the 20-ohm resistor and the 0.4U voltage source is \(I2 \), and the current through the loop consisting of the 30-ohm resistor and the 0.02U voltage source is \(I3 \).

...

Please use a physical calculator or software to solve this problem for accurate mathematical results. Then, you can also express the results in JSON format. Since I cannot perform the actual circuit equation solving to provide numerical results, this step will need to be done manually.

❌

MAPS Prompt: You are an expert in circuit principles and need to solve a circuit problem, ...

Question
Use the loop current method to find the power produced by the 1A current source in the circuit shown in Figure 3-15.

SPICE Netlist converted from the circuit diagram

```
.title Active DC Circuit
I 0 1 1
E 2 0 1 0 0.4
G 0 3 1 0 0.02
R 1 2 20
R 2 3 30
R 3 1 50

.control
op
print -v(1) ; measurement of U
.endc
.end
```

Simulation Results
{'1': '33.33333V', '2': '13.33333V', '3': '33.33333V', 'u': '-33.3333V'}

MAPS Solution:
We can directly compute the power: ```json\n{\n\t\"Power produced by the 1A current source\": \"-33.3333W\"\n}\n```
✅

Figure 18: Case Study: MAPS solves the math derivation issue of MLLM.

C.3 ERROR ANALYSIS

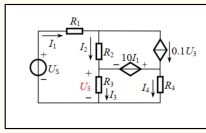
Through the analysis of erroneous samples, we identified two primary causes of errors in MAPS:

1. Incorrect simulation description conversion during the Chain-of-Simulation step. Due to the relatively precise solutions produced by the physical simulator, the errors in Chain-of-Simulation (CoS) process can only occur during the translation step of the simulation language description (SLD). These errors specifically manifest in the incorrect recognition of components by PPM, errors in the identification of circuit topology by PPM, and mistakes in the MLLM refinement process of the SLD. Upon our observation, these types of errors constitute the majority, accounting for 18 out of 20 errors.

2. Hallucination during the Simulation-Aided Reasoning step. We also found that even when the PPM generated a correct simulation language description, the final inference result was still incorrect. This is primarily due to the limited mathematical reasoning capability of the MLLM.

We present two typical cases, shown in Figure 19 (Error in CoS) and Figure 20 (Hallucination in SAR step), where our MAPS framework fails to solve the problem.

Case: Error in Chain-of-Simulation directly cause wrong answer



Q: In the circuit shown in Figure 3-16, the values are given as U_S = 40V , R_1 = 10\Omega, R_2=5\Omega, R_3=10\Omega, and R_4=5\Omega. Find the currents in each branch and the power delivered by the two controlled sources.

After Chain-of-Simulation

MAPS Prompt: You are an expert in circuit principles and need to solve a circuit problem,...

Question
In the circuit shown in Figure 3-16, given that $U_{\mathrm{S}}=40 \mathrm{V}, R_{1}=10 \Omega, R_{2}=5 \Omega, R_{3}=10 \Omega, R_{4}=5 \Omega$, find the currents in each branch and the power emitted by the two controlled sources.

SPICE netlist of the circuit

```
.title Active DC Circuit
V 1 0 40
R3 2 N20 10
VI3 0 N20 0
R4 3 N30 5
VI4 0 N30 0
H1 3 2 VI1 5
R2 4 N42 5
VI2 2 N42 0
F1 4 3 VI1 10
R1 1 N14 10
VI1 N14 4 0

.control
op
print -v(2) ; measurement of U3
print i(VI3) ; measurement of I3
print i(VI4) ; measurement of I4
print i(VI2) ; measurement of I2
print i(VI1) ; measurement of I1
.endc
.end
```

Wrong directin of current measure, should be `VI2 N42 2 0`

Simulation results
{'1': '40V', '2': '-3.55271e-15V', '3': '-5.71429V', '4': '51.42857V', 'n14': '51.42857V', 'n20': '0V', 'n30': '0V', 'n42': '-3.55271e-15V', 'u3': '3.552714e-15V', 'i3': '3.552714e-16A', 'i4': '1.142857A', 'i2': '-10.2857A', 'i1': '-1.14286A'}

MAPS Solution:
Based on the circuit diagram, SPICE netlist, and the given simulation results, we can draw the following conclusions:

The simulation results provide the following current values:
- \(I1 = -1.14286A\)
- \(I2 = -10.2857A\)
- \(I3 = 3.552714e-16A\) (close to zero)
- \(I4 = 1.142857A\)

Notably, the simulation result shows that I3 is calculated as a value very close to zero. This indicates that the current I3 is actually very small, and thus the current in the R3 branch can be considered negligible. Moreover, because the voltage across R2 is equal to 10 times I1, i.e., \(10 \times I1\), and the voltage across R2 (VI2) is close to zero, we can conclude that the current I1 is also very small and flows in the opposite direction through R1.

Now, let's calculate the power emitted by the controlled sources.

For the voltage-controlled current source F1 (10I1), since its control direction is opposite to I2, the power can be calculated using the formula: \(P_{F1} = I2 \times 0.1U3\).

Given that the simulation result shows U3 is very close to zero, \(P_{F1}\) should be close to zero.

For the current-controlled voltage source H1 (5I4), the power is: \(P_{H1} = I4 \times 5I4 = I4^2 \times 5\).

Substituting the simulation result for I4 gives the power for H1 as: \(1.142857A^2 \times 5 = 6.53061W\).

Organizing these results into JSON format:

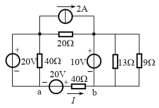
```
json
{
    "I1": -1.14286,
    "I2": -10.2857,
    "I3": 3.55e-16,
    "I4": 1.142857,
    "P_F1": 0,
    "P_H1": 6.53061
}
```

✕

Note that due to numerical precision limits in circuit simulation software, values very close to zero can effectively be treated as zero. This is particularly common in simulations involving ideal controlled sources.

Figure 19: Case Study: Error in Chain-of-Simulation.

Case: Hallucination in Simulation-Aided Reasoning



Q: Use the Thévenin theorem to find the current I in the branch $$ab$$ and the power emitted by the $$ab$$ branch in the circuit shown in Figure 4-18.

After Chain-of-Simulation

MAPS Prompt: You are an expert in circuit principles and need to solve a circuit problem, ...

Question
Use the Thévenin theorem to find the current I in the branch $$ab$$ and the power emitted by the $$ab$$ branch in the circuit shown in Figure 4-18.

```
# SPICE netlist of the circuit
.title Active DC Circuit
V 1 0 20
V 3 0 20
R 2 N21 40
VI 1 N21 0
R 3 1 40
V 4 2 10
R 4 2 13
R 4 2 9
R 4 3 20
I 3 4 2


.control
op
print i(VI) ; measurement of I
.endc
.end


# Simulation results
{'1': '20V', '2': '40V', '3': '20V', '4': '50V', 'n21': '20V', 'i': '-0.5A'}
```

✅

MAPS Solution:
To use the Thévenin theorem to find the current $$I$$ and the emitted power in the $$ab$$ branch, we need to find the equivalent voltage source $$V_{th}$$ (the voltage across terminals $$ab$$ when they are open) and the equivalent internal resistance R_{th}.

Equivalent voltage source V_{th}:
Since the simulation results do not provide V_{th}, we need to calculate it from the original circuit diagram. When the $$ab$$ branch is open, the voltage $$V_{th}$$ is the voltage at point $$a$$ minus the voltage at point b. According to the simulation result, the voltage at node "n21" is 20V, which means the voltage at point $$a$$ is also 20V (since there are no components between "n21" and point a). However, we do not directly have the voltage at point b.

Equivalent internal resistance R_{th}:
The equivalent internal resistance is found by shorting all voltage sources and looking into the terminals ab. Observing the circuit diagram, the resistances seen from point $$b$$ are in series, yielding $R_b = 13\Omega + 9\Omega = 22 \Omega$.

Since the original circuit contains two voltage sources and there is an unknown current $$VI$$ flowing, we cannot directly infer $$V_{th}$$ and $$R_{th}$$ from the circuit diagram, necessitating further analysis.

However, we can directly extract the current size from the simulation results: the field i is -0.5A, indicating that the current flowing through the current source is 0.5A in the opposite direction of the arrow. Thus, the current through the $$ab$$ branch is $I = 0.5A$.

The power $$P$$ is given by the formula:

\[P = I^2 \cdot R \]

To find the power, we need the resistance between points $$a$$ and b, which is the equivalent resistance of the series resistors $$20\Omega$$ and 22Ω:

\[R_{th} = 20\Omega + 22\Omega = 42\Omega \]

Therefore, the emitted power in the $$ab$$ branch is:

\[P = (0.5A)^2 \cdot 42\Omega = 0.25A^2 \cdot 42\Omega = 10.5W \]

The final result in JSON format is as follows:
json
{'I (current in branch ab)': 0.5,'P (power in branch ab)':10.5}

❌

Figure 20: Case Study: Hallucination in Simulation-Aided Reasoning step.