VeraGrid-Agent: Tool-Augmented LLMs for Distribution Optimal Power Flow at the Grid Edge

Shivanshu Tripathi, Hamed Mohsenian-Rad, and Maziar Raissi

***Abstract*— Language models have demonstrated remarkable success in solving a wide range of tasks. However, answering complex scientific questions about the power flow often requires solving the distribution optimal power flow (D-OPF) problem. These questions call for numerical solvers and simulators, as linguistic reasoning from parametric knowledge often gives incorrect answers. In this work, we present VeraGrid-Agent, a tool-augmented LLM that autonomously writes the simulator input, executes the open-source VeraGrid solver, and reads the solver output before answering. To evaluate performance, we introduce VeraGrid-MCQ-150, a set of deterministic, expert template driven, 150 multiple-choice questions. We evaluate the performance under two regimes: (i) no-tool reasoning and (ii) agent (LLM with simulator access). Without tools, every model performs with an accuracy of 41.3%–49.3%. However, with VeraGrid-Agent, accuracy increases to 97.3%–100.0%. We also do a failure-mode analysis to show that the few remaining errors arise from wrong interpretations during multi-step reasoning, rather than any failure in the simulators execution.**

Keywords: Large language models, grid edge, distributed energy resources, tool-augmented LLM, VeraGrid simulator

I. Introduction

Large language models (LLMs) have advanced rapidly in code generation and multi-step reasoning. These capabilities have encouraged their adoption in scientific and engineering domains, including power-systems. The electric grid is changing, with distribution feeders now hosting large amounts of distributed energy resources (DERs) [1], [2]. We built an agent that can solve complex questions that are not primarily linguistic, requiring the agents to call the solver tool. The questions include the rise of voltage at the PV node, the capacity of the bus, the curtailment of DERs, the power flow at the substation, or the feeder losses. When an LLM is asked such questions without any access to tools, it is forced to guess, producing outputs that are numerically incorrect, as illustrated in Fig 1.

We consider a DER dispatch problem at the grid-edge and use AC-OPF to solve the problem. For a given network of buses, branches, loads, and generators, AC-OPF finds the generator dispatch and bus voltages that minimize the total generation cost while satisfying non-linear AC power-flow equations and the operating limits. This problem is non-convex and NP-hard in general [3] and is solved by specialized packages such as the open-source VeraGrid simulator.

This gap has motivated the use of tool-augmented LLMs that connect language reasoning with calls to external software. Therefore, rather than answering from parametric memory, the model writes the input, executes the solver, reads back the quantities it needs from the solution, and only then commits to an answer. The question we address is whether this pattern is sufficient to measure progress in a way that separates genuine computation from memorization and guessing.

S. Tripathi and H. Mohsenian-Rad are with the Department of Electrical and Computer Engineering, and M. Raissi is with the Department of Mathematics at the University of California, Riverside, {strip008, hamedrad, maziarr}@ucr.edu.

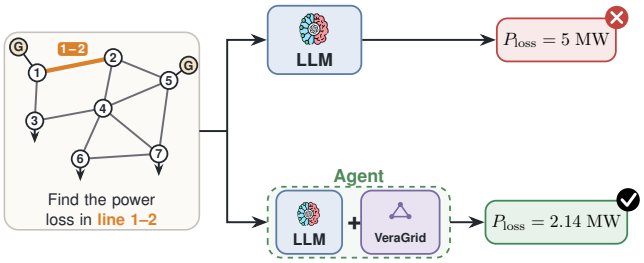


Fig. 1. The figure compares a standalone LLM with an agent given the same grid-edge query for a given circuit configuration. The standalone LLM answers from memory and returns an incorrect loss value, whereas the agent invokes the VeraGrid power-flow solver and computes the correct answer.

A. Related Work

Several works extend the capabilities of LLMs by enabling them with external tools. Toolformer showed that language models can self-supervise to learn when to call tools [4]. ReAct showed that connecting LLMs with tool actions improves multi-step decision making and factual grounding [5]. Other agentic frameworks, such as MRKL systems [6], HuggingGPT [7] and Gorilla [8] showed that LLMs can orchestrate diverse tools, models, and APIs to solve tasks beyond their parametric knowledge. Our work follows this paradigm by equipping an LLM with a solver to get reliable answers for complex power-flow questions that cannot be recovered from parametric knowledge alone.

The use of LLMs is rapidly expanding in power system applications [9]. Recent works like ChatGrid [10] and ElecBench [11] have used retrieval-augmented LLMs to answer questions about power-system. However, these works focus on the textual correctness and usefulness of the generated response, rather than numerical understanding in simulator derived ground truth. More recently, GridMind [12] moved toward executable power-system analysis by coupling LLMs with deterministic solvers. Our work complements these efforts by introducing an MCQ-based evaluation scheme to score different LLMs. PowerAgentBench [13] evaluates tool-using agents in steady-state workflows, such

as contingency analysis and mitigation using hidden physical validation. The benchmark in [14] evaluates a broad set of tasks, including power flow, protection, and stability, with deterministic evaluators and held-out test cases [14]. The work [15] shows that current models struggle in complicated mathematical analysis. Our work introduces a VeraGrid-MCQ-150, a benchmark that evaluates the performance of models, and their ability to correctly retrieve quantitative information from a solved network through 150 MCQ questions.

General scientific and mathematical benchmarks, such as MATH and SciBench [16]–[18] have driven progress in quantitative reasoning but rely on static problem sets that are susceptible to memorization. Our generator addresses this by making every answer a deterministic function of a specific solved record, i.e., the same template yields a different ground truth under different network configurations. Recent work [19] trains a machine learning model to solve OPF, rather than approximating the solver. Instead, VeraGrid-Agent directly calls a deterministic solver, giving exact answers over guessed answers.

B. Contributions

This paper makes three main contributions. First, we develop a tool-augmented architecture for D-OPF analysis in which the LLM writes the feeder and DER simulator input, executes VeraGrid, and reads only the result sections relevant to the question, keeping the context compact and every answer traceable to the solver output. Second, we provide a deterministic, template-driven benchmark of 150 MCQs at easy, medium, and hard difficulty levels, with verified ground truth of the simulator that is reproducible and dependent on the feeders and its DERs. Third, we conduct an empirical study consisting of seven LLMs under the no-tool and agent regimes, showing that having access to the tools substantially raises accuracy. We also discuss where the tool helps the most and analyze the failure modes of the agent.

II. VeraGrid-Agent

A. Task overview

The D-OPF with DER dispatch for a feeder graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is expressed as:

$$\min_{V_i, p_i, q_i} \sum_{\mathcal{E}} |I_{ij}|^2 + \lambda_c \sum_{\mathcal{D}} (\bar{p}_i - p_i) + \lambda_v \sum_{\mathcal{N}} |V_i|^2 \tag{1a}$$

$$\text{s.t. } S_i^{\mathrm{G}} + s_i - S_i^{\mathrm{L}} = V_i I_i^*, \ \ \mathbf{I} = \mathbf{Y}\mathbf{V} \tag{1b}$$

$$0 \le p_i \le \bar{p}_i, \quad |s_i| \le \bar{s}_i \tag{1c}$$

$$\underline{V}_i \le |V_i| \le \overline{V}_i, \quad |S_{ij}| \le \bar{S}_{ij} \tag{1d}$$

$\mathcal{N}$, $\mathcal{E}$, $\mathcal{D} \subseteq \mathcal{N}$ are the sets of buses, branches, and DER buses; $V_i = |V_i| e^{\jmath\theta_i}$, I_i, and I_{ij} the bus voltage, injected current, and branch current; $\mathbf{Y}$ the admittance matrix; S_i^{G}, S_i^{L} the substation and load power; $s_i = p_i + \jmath q_i$ the DER injection with limits $\bar{p}_i$ (available power) and $\bar{s}_i$ (inverter rating); and $\lambda_c, \lambda_v > 0$ the objective weights. The Problem (1) is non-convex in general [3], so practical instances are solved numerically. Let Φ denote the solution operator implemented by VeraGrid simulator and let $R = \Phi(\mathcal{G})$ denote the output of the simulator containing records of voltages, flows, dispatches, and cost. Our task is to answer queries in natural-language about R without revealing R directly to the model. To solve the problem, we design a VeraGrid-Agent that autonomously writes, executes, and interprets AC-OPF simulations.

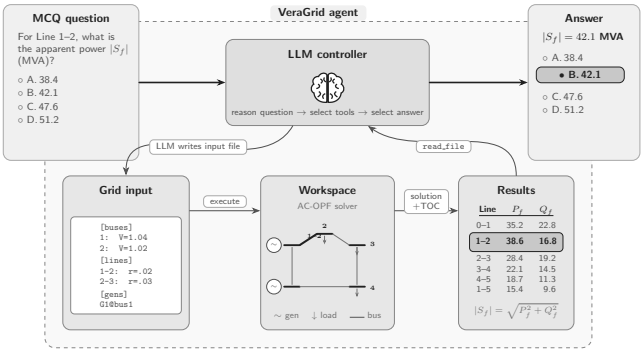


Fig. 2. The figure shows the VeraGrid-Agent workflow. For each question, the LLM receives the feeder and DER description and MCQ query, then iteratively calls the tools before emitting the final answer.

We evaluate the agent by introducing the VeraGrid-MCQ-150 dataset, a rule-based benchmark of 150 four-option multiple-choice questions (MCQs) with single correct answer generated autonomously from generation rules specified by domain experts.

B. Performance metric

Consider each MCQ with correct label $\ell^\star$ and the predicted label by LLM or the agent ($\mathcal{A}$) as $\hat{\ell}$. Missing or malformed responses count as incorrect. Performance is evaluated as:

$$\mathrm{Acc}(\mathcal{A}) = \frac{1}{|\mathcal{Q}|} \sum_{i=1}^{|\mathcal{Q}|} \mathbb{1}\big[\hat{\ell}_i = \ell_i^\star\big]. \tag{2}$$

where $|\mathcal{Q}|$ denotes the number of questions.

C. Tool Interface and Control Loop

Fig. 2 summarizes the workflow of VeraGrid-Agent. Each question is solved in an isolated workspace. The LLM may call these tools: `read_file` for workspace inspection; `write_file` to author the JSON input describing the feeder and DER configuration; and `execute_veragrid` to run the simulator. A deliberate design choice is that the solver tool does *not* return the full solution. It returns execution metadata and a table of contents for the results file. This helps the agent to have targeted reads, keep the context window compact, and make every reported value traceable to a specific output line. The agent follows the ReAct framework [5], at each step, it either calls a tool or submits a final answer. To prevent runaway loops in cases where the model keeps issuing tool calls without converging on an answer, we cap the number of iterations at $T_{\max}$, the agent returns failure if this budget is exhausted. In practice, the great majority of queries are resolved in a few iterations.

D. System prompt

The prompt consists of: system prompt, and query prompt. The system prompt defines the agent's role, available tools, the VeraGrid input schema, the feeder and DER details, and answer format. It is fixed across interactions and encodes the operational rules of the agent. This defines the feeder and DER configuration and changes whenever the configuration changes. The query prompt contains the user query as MCQ. The system prompt provided to the LLM controller is:

```
You are a VeraGrid agent.

=== TOOLS ===
- read_file(path, start, end): read a file
- write_file(name, content): overwrite
file
- run_veragrid(input): run VeraGridEngine

=== WORKFLOW RULES ===
- use tools for every quantitative query
if available
- on failure, inspect error, fix and retry

=== INPUT SCHEMA (JSON) ===
{ "name", "baseMVA", "baseKV",
  "buses": [{id, type, Vm, Pd, Qd}],
  "branches": [{from, to, r, x, b,
rateA}],
  "ders": [{bus, kind, Pavl, Sinv, Qmin,
Qmax}],
  "substation": {bus, Vset} }

=== FEEDER DETAILS ===
Substation @ Bus 0: V = 1.00 pu, 12.47 kV
Bus 17 (PQ+PV): Pd = 0.10 MW, Qd = 0.06
MVAr, Pavl = 0.60 MW, Sinv = 0.66 MVA
Bus 30 (PQ+storage): Pd = 0.20 MW, Sinv =
0.50 MVA
Line 0-1: r = 0.05, x = 0.11, rateA = 6.0
MVA
Voltage band: 0.95-1.05 pu
...

=== ANSWER FORMAT ===
Respond with ONLY one option (A, B, C, D).
```

III. Rule based Dataset generation

The VeraGrid-MCQ-150 is produced by a deterministic, template-driven generator whose ground truth is computed directly by the VeraGrid simulator. Given the ground truth answers from the simulator $R = \Phi(\mathcal{G})$, each template in the library specifies (i) a question, (ii) a formula that computes the correct value from R, and (iii) rules for constructing the three distractors. The benchmark contains three difficulty levels: easy, medium, and hard, with 50 questions per livel. Each question is constructed so that it cannot be answered from the LLM's parametric memory alone. The generated questions are dependent on the feeder and DER configuration, i.e., different configuration yields different sets of questions.

The three expert-defined difficulty tiers span direct retrieval, single-step derivation, and multi-step quantitative reasoning. The benchmark comprises 33 templates in total: 8 Easy, 10 Medium, and 15 Hard.

- Easy: The questions include direct lookup, counting, and identification on the solved feeder. Examples include node voltage magnitudes, DER inventory, feeder element counts, extrema (e.g., the worst voltage node or the most heavily loaded transformer), solver convergence status, and feeder base voltage and base power. In this, we do not require any arithmetic beyond retrieval.
- Medium: These questions require single-step derivations to retrieve answers, such as per-unit conversion, nodal power balance at DER node, line impedance magnitude $|Z| = \sqrt{r^2 + x^2}$, DER apparent power and power factor, generator utilization and reserve, curtailment faction, the R/X ratio, the feeder loss ratio, and the voltage rise at DER node.
- Hard: These questions require multi-step computation, cross-table inference, and system-level statistics. These include branch loss aggregated from both terminals, $P_{\text{loss}} = P_f + P_t$; net reserve power at sub-station, per-unit current magnitude, $|I_f| = |S_f|/V_m$; bus voltage-angle differences; DER real/reactive dispatch shares, and and thermal margins.

The benchmark is designed to evaluate the computational ability of an agent rather than its parametric knowledge. Since every answer is derived from a specific solved network, any change in loads, impedances, or generator limits alters the ground truth, so answers cannot be recalled from memorized data. The answer choices are randomly permuted to mitigate position and selection biases [18]. The benchmark and results are fully reproducible and the results can be evaluated on any topology, or operating conditions[1].

A. Evaluation Protocol

We evaluate the framework under two regimes that differ only in tool access, with system prompt and MCQs fixed.

- No-tool: one chain-of-thought response from the feeder and DER description and the question.
- Agent: the full VeraGrid-Agent loop with iteration and self-correction [5].

The two regimes differ in both solver access and interaction protocol. The no-tool baseline produces a single chain-of-thought response without using VeraGrid, whereas the agent calls the VeraGrid tool iteratively and can correct its output after a failed execution.

IV. Results

We consider a DER integrated radial distribution feeder with 8-buses as shown in Fig. 3, with 185 MW total load with 100 MVA base. We consider two regimes: a no-tool baseline in which the model answers from the feeder and DER description alone, and a tool-augmented agent in which the model may invoke the VeraGrid distribution AC-OPF

[1]The dataset and code are open source at https://github.com/Shivanshu-11/veragrid-llm-benchmark, and a preview is available at https://shivanshu-11.github.io/-mcqbuilder.veragrid/.

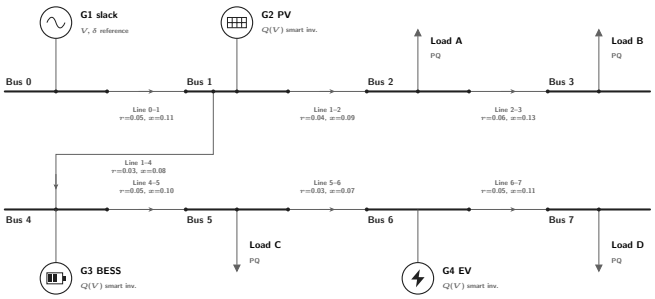


Fig. 3. The figure shows a DER-integrated radial distribution feeder with 8-bus, and 4-generator used in VeraGrid-MCQ-150. Bus 0 is the slack, buses 1, 4, and 6 host rooftop PV, battery storage, and EV charging with smart inverter reactive control. The remaining buses are PQ loads.

TABLE I

ACCURACY ON VERAGRID-MCQ-150 IN THE NO-TOOL AND TOOL-AUGMENTED AGENT REGIMES. THE Δ IS THE IMPROVEMENT FROM THE NO-TOOL TO AGENT CASE.

Model	Mode	Easy	Medium	Hard	Overall
GPT-5.2	No tool	18/50	28/50	18/50	42.7%
	Agent	50/50	50/50	49/50	99.3%
	Δ	+32	+22	+31	+56.7%
Claude Sonnet 4.5	No tool	18/50	34/50	19/50	47.3%
	Agent	50/50	50/50	49/50	99.3%
	Δ	+32	+16	+30	+52.0%
Gemini 3 Flash	No tool	20/50	33/50	21/50	49.3%
	Agent	50/50	50/50	50/50	100.0%
	Δ	+30	+17	+29	+50.7%
Grok 4.3	No tool	24/50	27/50	18/50	46.0%
	Agent	50/50	50/50	50/50	100.0%
	Δ	+26	+23	+32	+54.0%
Grok 4.5	No tool	20/50	28/50	19/50	44.7%
	Agent	50/50	50/50	50/50	100.0%
	Δ	+30	+22	+31	+55.3%
Composer 2.5	No tool	19/50	30/50	18/50	44.7%
	Agent	50/50	50/50	46/50	97.3%
	Δ	+31	+20	+28	+52.7%
Opus 4.8	No tool	19/50	29/50	14/50	41.3%
	Agent	50/50	50/50	48/50	98.7%
	Δ	+31	+21	+34	+57.3%

solver and read back the solved state before committing to an answer.

We compare the performance of several LLM models such as GPT-5.2, Claude Sonnet 4.5, Gemini 3 Flash, Grok 4.3, Grok 4.5, Composer 2.5, and Opus 4.8.

A. Parametric Baseline vs. Tool-Augmented Accuracy

Table I reports the overall accuracy of the two regimes: without tool access and with VeraGrid-Agent access.

Without tools, every model scores only 41.3–49.3%. However, with the model having access to the VeraGrid-agent, every model scores 97.3–100.0%. Gemini 3 Flash, Grok 4.3, and Grok 4.5 reach perfect 100% accuracy; GPT-5.2 and Claude Sonnet 4.5 each miss a single hard question, and Opus 4.8 and Composer 2.5 misses the correct answer for two and four questions, respectively.

TABLE II

ERROR DECOMPOSITION FOR EACH MODEL IN AGENT MODE, REPORTED AS COUNTS OVER 150 QUESTIONS PER MODEL. NO TOOL USE REFERS TO CASES IN WHICH THE MODEL ANSWERED FROM MEMORY WITHOUT INVOKING ANY EXTERNAL TOOLS. WRONG INTERPRETATION DENOTES CASES IN WHICH THE MODEL SUBMITTED AN INCORRECT ANSWER.

Model	No tool use	Wrong interpretation	Total
GPT-5.2	0	1	1
Claude Sonnet 4.5	0	1	1
Gemini 3 Flash	0	0	0
Grok 4.3	0	0	0
Grok 4.5	0	0	0
Composer 2.5	1	3	4
Opus 4.8	0	2	2

B. Where Tools Help and Failure mode analysis

The benefit of the tool is strongly uneven across different categories, as shown in Fig. 4. We observe that the questions that are computable from the circuit description alone gain little because the no-tool baseline is already high. For example, questions related to line parameters, feeder topology, and load and DER nameplate data can be answered fairly well without the solver, whereas questions about reserve power flow, feeder losses, and node voltages require it.

Table II decomposes the error modes for different models. There are no execution errors or timeouts. Examples of errors are semantic representations, e.g., computing branch loss from one terminal instead of using total power loss, or confusing DER curtailment with dispatch, and so on. Across different models, the median solved question needs only 1-4 tool calls, which takes an average of 17-27 s. Opus 4.8 is the most economical backbone with median call to the 1 tool in 17 s, yet not the most accurate, whereas the model with the full accuracy scorers spends 3-4 calls with 20-24 s.

C. Discussion

The results provide a comparison of the performances of the agent (LLM+tools) and LLM alone. VeraGrid performs the DER dispatch computations, while the model translates a natural-language query into the appropriate sequence of tool calls and maps the returned quantities to the correct answer.

For grounded grid-edge question answering, the choice of LLM backbone matters far less than access to computation. In fact, the best models like Opus-4.8 and GPT-5.2 have lower performance than many older models. Once numerical reasoning is provided to the solver, large differences in baseline performance disappear largely. Two observations prove this conclusion. First, the benefits of tool use are highly concentrated. Categories that can already be answered from the prompt, including feeder topology, line parameters, and load data, exhibit little improvement. Second, the failures are primarily semantic rather than the solver's error. Models can execute the solver correctly, yet still produce an incorrect

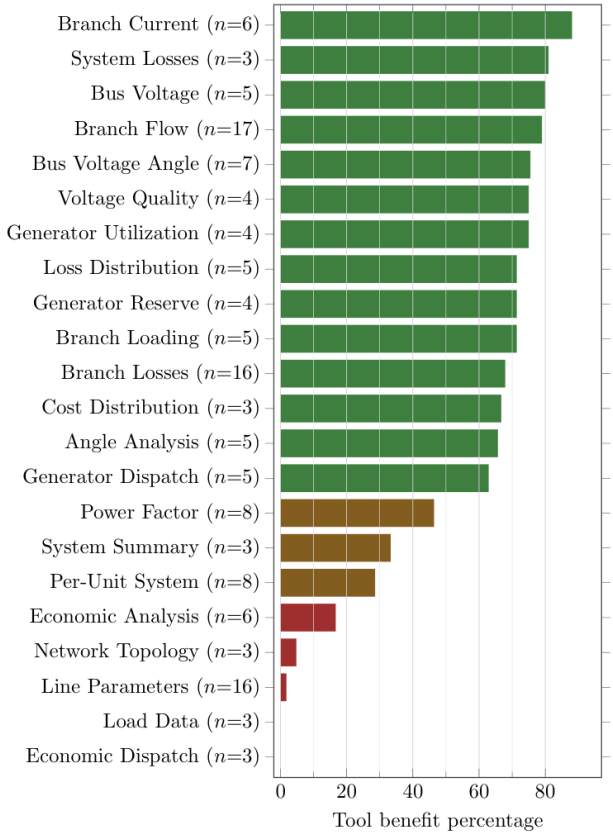


Fig. 4. The figure illustrates the improvement from tool augmentation relative to the no-tool baseline, averaged across models on different question category with $n \geq 3$. The tool benefit is defined as the difference in accuracy between the agent and no-tool settings, averaged across all models. The green bars indicate substantial gains of 60–90%, whereas the red bars show only marginal improvements for categories where the no-tool baseline already achieves high accuracy.

answer by formulating the wrong procedure, such as aggregating branch loss at a single terminal, or confusing DER curtailment with DER dispatch.

The results indicate that the solver access is the main source of the gain. Categories answerable from the feeder description, such as topology and line parameters, gain little, since a single no-tool response already answers them well. This suggests that the no-tool baseline is limited by missing information rather than by the number of reasoning steps.

V. Conclusion

We presented VeraGrid-Agent, a tool augmented LLMs, and the VeraGrid-MCQ-150 benchmark, a reproducible testbed to evaluate the performance of the LLMs in the no-tool and tool augmented settings. Our experiments show that access to external tools improves performance and can solve tasks that cannot be reliably solved using parametric knowledge alone. These results highlight the importance of solver-based agents for numerically intensive engineering. Still several limitations remain as our experiments use only a single radial feeder modeled in balanced form. Future work will extend this framework to larger unbalanced three-phase feeders. Another issue is that the VeraGrid solver takes a long time to give the final solution. Future work will reduce the time-taken by the agent to reach the final answer. We will use the reasoning and the response generated by VeraGrid-Agent to fine-tune a small language model to have improved power-system reasoning capabilities without access to tools.

References

[1] M. H. J. Bollen and F. Hassan, *Integration of Distributed Generation in the Power System*. Wiley-IEEE Press, 2011.

[2] S. Tripathi, A. Jain, and A. K. Behera, "A partially distributed fixed-time economic dispatch algorithm with Kron's modeled power transmission losses," *IEEE Transactions on Control of Network Systems*, vol. 11, no. 1, pp. 498–509, 2023.

[3] D. Bienstock and A. Verma, "Strong NP-hardness of AC power flows feasibility," *Operations Research Letters*, vol. 47, no. 6, pp. 494–501, 2019.

[4] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, E. Hambro, L. Zettlemoyer, N. Cancedda, and T. Scialom, "Toolformer: Language models can teach themselves to use tools," in *Advances in Neural Information Processing Systems*, vol. 36, 2023.

[5] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing reasoning and acting in language models," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2023.

[6] E. Karpas, O. Abend, Y. Belinkov, O. Lieber, N. Ratner, Y. Shoham, H. Bata, Y. Levine, K. Leyton-Brown, D. Muhlgay, N. Rozen, E. Schwartz, G. Shachaf, S. Shalev-Shwartz, A. Shashua, and M. Tenenholtz, "MRKL systems: A modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning," *arXiv preprint arXiv:2205.00445*, 2022.

[7] Y. Shen, K. Song, X. Tan, D. Li, W. Lu, and Y. Zhuang, "HuggingGPT: Solving AI tasks with ChatGPT and its friends in Hugging Face," in *Advances in Neural Information Processing Systems*, vol. 36, 2023.

[8] S. G. Patil, T. Zhang, X. Wang, and J. E. Gonzalez, "Gorilla: Large language model connected with massive APIs," in *Advances in Neural Information Processing Systems*, vol. 37, 2024, pp. 126 544–126 565.

[9] M. Sarwar, M. Rizwan, M. Aziz, and A. R. Sudais, "Large language models for power system applications: A comprehensive literature survey," *arXiv preprint arXiv:2512.13004*, 2025.

[10] M. Ni, J. Zhang, C. Fu, J. Wang, X. Ning, and S. Li, "ChatGrid: Intelligent knowledge Q&A for power dispatching control based on large language models and retrieval-augmented generation," in *2024 IEEE 7th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, vol. 7, 2024, pp. 921–925.

[11] X. Zhou, H. Zhao, Y. Cheng, G. Liang, G. Liu, W. Liu, Y. Xu, and J. Zhao, "ElecBench: A power dispatch evaluation benchmark for large language models," *arXiv preprint arXiv:2407.05365*, 2024.

[12] H. Jin, K. Kim, and J. Kwon, "GridMind: LLMs-powered agents for power system analysis and operations," *arXiv preprint arXiv:2509.02494*, 2025.

[13] C. Mylonas, M. Foti, A. Pomarico, M. Duarte, Q. Zhang, and E. Varvarigos, "PowerAgentBench-SS: A benchmark for agentic AI in power system steady-state studies," *arXiv preprint arXiv:2606.18789*, 2026.

[14] S. Trashchenkov, "Power systems agent benchmark: Executable evaluation of AI agents in electric power engineering," *arXiv preprint arXiv:2606.20950*, 2026.

[15] C. Shen, Z. Guo, X. Wan, Z. Yang, Y. Zhang, W. Huang, J. Song, Z. Zhang, and M. Sun, "ProOPF: Benchmarking and improving LLMs for professional-grade power systems optimization modeling," *arXiv preprint arXiv:2602.03070*, 2026.

[16] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Zhu, D. Song, and J. Steinhardt, "Measuring mathematical problem solving with the MATH dataset," *arXiv preprint arXiv:2103.03874*, 2021.

[17] X. Wang, Z. Hu, P. Lu, Y. Zhu, J. Zhang, S. Subramaniam, A. R. Loomba, S. Zhang, Y. Sun, and W. Wang, "SciBench: Evaluating college-level scientific problem-solving abilities of large language models," in *Proc. Int. Conf. Machine Learning (ICML)*, vol. 235, 2024, pp. 50 622–50 649.

[18] C. Zheng, H. Zhou, F. Meng, J. Zhou, and M. Huang, "Large language models are not robust multiple choice selectors," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2024.

[19] D. Owerko, F. Gama, and A. Ribeiro, "Optimal power flow using graph neural networks," *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 5930–5934, 2020.