RESEARCH-ARTICLE

GridMind: LLMs-Powered Agents for Power System Analysis and Operations

HONGWEI JIN, Argonne National Laboratory, Lemont, IL, United States

KIBAEK KIM, Argonne National Laboratory, Lemont, IL, United States

JONGHWAN KWON, Argonne National Laboratory, Lemont, IL, United States

PDF Download
3731599.3767409.pdf
29 December 2025
Total Citations: 0
Total Downloads: 1311

Published: 16 November 2025

Citation in BibTeX format

SC Workshops '25: Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis
November 16 - 21, 2025
MO, St Louis, USA

Conference Sponsors:
SIGHPC

GridMind: LLMs-Powered Agents for Power System Analysis and Operations

Hongwei Jin
Argonne National Laboratory (ANL)
Lemont, USA
jinh@anl.gov

Kibaek Kim
Argonne National Laboratory (ANL)
Lemont, USA
kimk@anl.gov

Jonghwan Kwon
Argonne National Laboratory (ANL)
Lemont, USA
kwonj@anl.gov

Abstract

The complexity of traditional power system analysis workflows presents significant barriers to efficient decision-making in modern electric grids. This paper presents GridMind, a multi-agent AI system that integrates Large Language Models (LLMs) with deterministic engineering solvers to enable conversational scientific computing for power system analysis. The system employs specialized agents coordinating AC Optimal Power Flow and N-1 contingency analysis through natural language interfaces while maintaining numerical precision via function calls. GridMind addresses workflow integration, knowledge accessibility, context preservation, and expert decision-support augmentation. Experimental evaluation on IEEE test cases demonstrates that the proposed agentic framework consistently delivers correct solutions across all tested language models, with smaller LLMs achieving comparable analytical accuracy with reduced computational latency. This work establishes agentic AI as a viable paradigm for scientific computing, demonstrating how conversational interfaces can enhance accessibility while preserving numerical rigor essential for critical engineering applications.

CCS Concepts

• **Computing methodologies** → **Artificial intelligence**; *Planning and scheduling*; Multi-agent systems.

Keywords

Agentic AI, Power System Analysis, LLM Agents, AC Optimal Power Flow, Contingency Analysis, Multi-Agent Systems

ACM Reference Format:
Hongwei Jin, Kibaek Kim, and Jonghwan Kwon. 2025. GridMind: LLMs-Powered Agents for Power System Analysis and Operations. In *Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC Workshops '25), November 16–21, 2025, St Louis, MO, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3731599.3767409

1 Introduction

The emergence of Large Language Models (LLMs) and agentic AI systems is fundamentally transforming how we approach complex scientific and engineering problems. Recent advances in agentic AI—where autonomous agents can plan, execute, and reason about complex tasks—present unprecedented opportunities to revolutionize scientific and engineering computing workflows [2, 17, 22]. Unlike simple conversational interfaces, agentic systems can orchestrate multiple computational processes, maintain context across complex analyses, and provide intelligent decision support through natural language interactions. A key capability of these agents is their ability to invoke existing computational functions and tools, ensuring that analytical requests are executed using reliable, validated methods and producing trustworthy results.

This paper examines the application of an agentic AI system in electric power systems, which requires complex planning and operation due to a wide range of technical and operational constraints. Modern power system analysis involves a diverse set of simulations across multiple time scales and levels of fidelity, including state-estimation, load and resource availability forecasting, alternative current optimal power flow (ACOPF), contingency analysis (CA), security-constrained unit commitment and economic dispatch, and long-term capacity planning with system reliability and resource adequacy assessments [11, 18, 29]. Each of these domains traditionally requires separate tools, programming expertise, and deep technical knowledge, leading to inefficient decision-making and fragmented workflows and analysis.

This paper introduces **GridMind**, an early prototype of a **multi-agent** AI system that explores the potential of LLM-powered agents in power system analysis. Rather than relying on tool-driven workflows, our system demonstrates how conversation-based interactions can augment scientific computing, enabling domain experts to perform analyses more intuitively and efficiently.

GridMind addresses four challenges in power system simulation and analysis: (1) **Workflow integration**—seamlessly connecting disparate analysis tasks through intelligent orchestration, (2) **Knowledge accessibility**—reducing programming barriers while maintaining technical rigor, (3) **Context preservation**—maintaining analytical coherence across complex multi-step processes, and (4) **Expert augmentation**—enhancing human decision-making

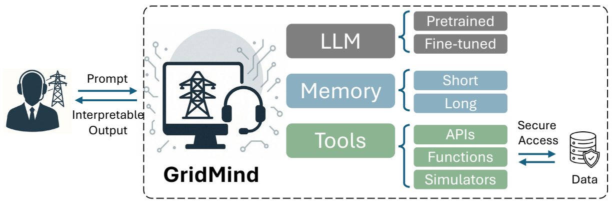


Figure 1: GridMind: LLM Agents for Power System

through AI-powered insights and recommendations. Our key contributions are threefold. First, we present a novel multi-agent architecture demonstrating how LLM agents orchestrate complex scientific engineering workflows through natural conversation. Second, we develop integrated power system analysis capabilities spanning multiple domains, coordinated through agentic workflows with conversational interfaces that interpret multi-step analytical requests. Third, we implement advanced context management preserving analytical state across agent interactions, with comprehensive evaluation confirming that agentic workflows maintain technical accuracy through structured function tools.

More specifically, **GridMind** demonstrates how agentic AI lowers access barriers while supporting the natural iterative "what-if" analysis (e.g., adjust load levels, re-solve, inspect impacts). Automated tool selection and chaining compresses turnaround from question to vetted result. Built-in convergence, constraint, and data integrity checks keep numerical rigor visible in plain language instead of opaque solver logs. The system also serves as an instrumentation bench, logging solver metrics plus LLM backend latency, token usage, and occasional factual slips so reliability trends can be monitored. The prototype GridMind points toward similar gains in other scientific engineering domains that depend on multi-step analytical workflows.

2 Related Work

Agentic AI systems represent a fundamental shift from reactive to proactive AI, where agents can plan, execute, and reason about complex tasks autonomously [22]. Recent work has demonstrated the potential of LLM-powered agents in various domains, including software engineering [7], scientific discovery [26], and system administration [15]. However, limited research exists on applying agentic workflows to scientific computing, particularly in engineering domains requiring rigorous numerical analysis such as power system analysis. Moreover, multi-agent systems (MAS) have shown promise in coordinating complex computational tasks [10, 27], but the integration of natural language processing with scientific computing agents remains largely unexplored. Our work contributes to this emerging field by demonstrating how conversational agents can orchestrate sophisticated engineering analyses.

The application of natural language processing to technical domains has gained significant attention with the advancement of large language models [13, 16]. Code generation and technical reasoning capabilities have enabled new paradigms for human-computer interaction in specialized domains [4]. Frameworks such as LangChain [23], PydanticAI [24], and AutoGen [28] provide abstractions for tool invocation, memory/context handling, and (in AutoGen) multi-agent conversational patterns. PydanticAI explicitly enforces structured input/output validation via Pydantic models, whereas LangChain and AutoGen rely more on conventional Python typing plus runtime checks—so strong type safety and schema validation are not uniformly emphasized across all three. Despite these advances, deploying these general-purpose frameworks for tightly integrated, cross-domain scientific engineering workflows (e.g., coupling economic optimal power flow with large-scale contingency reliability assessment under a shared validated state) remains underexplored: current libraries supply generic orchestration primitives but limited built-in support for domain-specific numerical consistency checks, solver provenance tracking, and cross-analysis result fusion required in power system analytics.

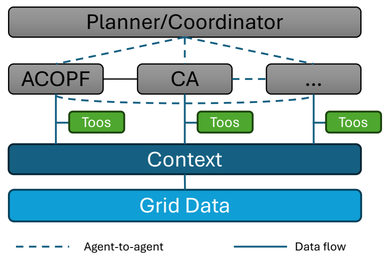


Figure 2: Multi-Agent Collaboration

The GridMind prototype focuses on two power system applications, ACOPF and CA, coordinated through an agentic workflow. ACOPF is a power system scheduling problem. It determines generator dispatch setpoints to minimize total operating costs while enforcing AC power flows and other technical and operational constraints [3, 11]. Traditional optimization solution approaches include interior-point methods [20], semidefinite programming relaxations [12], and metaheuristic algorithms [1].

CA evaluates system reliability under T-1 outages of system assets. It determines post-contingency system operating conditions, including power flow limit violations, voltage violations, and involuntary load shedding [9]. CA also identifies critical system components whose failure could cause widespread violations [8]. Traditional CA tools require extensive setup and expertise [14, 19, 21]. Modern power system solver packages such as MATPOWER [30], PowerModels.jl [5], and PandaPower [25] provide robust implementations but require specialized programming knowledge and lack integrated workflow support across analysis domains. Thus, the GridMind prototype aims to bridge these gaps by providing a user-friendly interface for complex power system analyses, enabling seamless collaboration between human experts and AI agents. It enables intuitive query formulation and result interpretation through natural language interactions and designs to benefit from the reasoning and thinking capabilities of LLMs for enhanced decision-making.

3 GridMind Agentic System Architecture

GridMind employs a multi-agent architecture that demonstrates how agentic AI can orchestrate complex scientific computing workflows. The core innovation of GridMind lies in our multi-agent orchestration approach (Figure 2), where specialized agents handle different aspects of power system analysis while maintaining coherent conversations and shared analytical context. The system comprises specialized agents working in coordination: (1) ACOPF agent, which specializes in economic scheduling of power systems and power flow analysis, (2) CA agent, which focuses on T-1 reliability assessment and critical element identification, (3) agent coordinator that manages inter-agent communication, context sharing, and complex multi-step analyses, and (4) planner agent analyzes user requests to determine appropriate agent assignment and workflow coordination.

3.1 Conversational Interface

The interface is a thin front door: you type intent; the agents quietly parse it, plan a minimal sequence, call deterministic numerical solvers, check the numbers, and reply—no ad-hoc scripts. Shown in Appendix D.

The agent handles intent and entity extraction (e.g., case id, buses, MW changes, outage scope) and sketches a compact plan. It then orchestrates typed tool calls for ACOPF, topology or load edits, and contingency sweeps, then layers validation: convergence flags, power balance tolerance, operating limits, and sanity checks on modified elements. A structured context keeps the latest solved state, applied diffs, and cached contingency fragments so only affected layers are recomputed. Every reported number is pulled from stored structured results, making the reply auditable and reproducible with timestamps and call metadata. New analytical tools can be registered with a schema; the planner notices capabilities without refactoring core logic. The result is a deterministic loop—parse, plan, invoke, validate, narrate, persist—that preserves engineering rigor while trimming the friction of multi-step exploratory studies.

3.2 Domain Agents

We have two domain agents, ACOPF and CA, each with specialized knowledge and toolsets.

3.2.1 *ACOPF Agent.*

The ACOPF agent is implemented as an LLM-driven computational assistant that fuses four pillars: (1) a pre-trained reasoning model (i.e., the LLM), (2) structured conversational memory to maintain context state, (3) a vetted toolbox of deterministic power system solvers, and (4) carefully engineered prompts that bind user intent to verifiable analytical actions. Rather than hallucinating numerical outcomes, the agent plans in language, invokes trusted functions for every quantitative step, then interprets the returned structured results back into domain-aware explanations. And it is designed as a "reason-act-reflect" loop in which the agent can reason about the problem, act on it, and reflect on the results to ensure correctness and reliability.

LLM core and prompting. The language model supplies chain-of-thought style reasoning: it decomposes a user request (e.g., "Evaluate the economic impact of removing transmission line (Line) between buses 37 and 40 in the IEEE 118-bus case") into actionable subgoals: (a) load or confirm the target network, (b) verify whether a base ACOPF solution exists in context, (c) schedule a re-dispatch under the modified topology, (d) compare objective cost, constraint margins, and voltage profile, (e) prepare a concise, auditable report. System prompts constrain behavior ("Never fabricate solver outputs; always call tools for numerical data") while tool-specific prompts provide schemas for input/output validation.

Memory (context). A structured in-session memory object stores: current case metadata (case id, bus/gen counts, last solve timestamp), the latest feasible dispatch, contingency results cache, and any modifications (e.g., load adjustments, line outages) as a chronological diff log. Before acting, the agent replays relevant slices of this state to ground its reasoning ("A solved ACOPF for IEEE 57 exists with generation cost \$41,532.17; two loads were increased previously."). This prevents redundant solves and enables cumulative what-if analysis.

Tool integration. Every numerical claim originates from registered tools, e.g., PandaPower; The agent selects tools by matching inferred subtask types ("needs re-optimization") with tool capability descriptors, then emits a JSON-typed call specification. Post-execution, returned objects are validated (e.g., convergence flag, max power balance mismatch $< 10^{-4}$ p.u.) before interpretation. If validation fails, the agent triggers an automatic recovery path (adjust solver tolerances, fall back to an alternative algorithm, or request clarification).

Trust and auditability. Every numeric in the narrative maps to a field in a stored tool output object (from structured data). This design ensures that the agent does not fabricate numbers or make unverifiable claims. Instead, it provides a transparent audit trail: each request will be traceable to its originating data sources and use proper tools to solve the case in a numerical way, instead of completely relying on LLMs' reasoning.

In sum, the single ACOPF agent exhibits disciplined cognitive cycles—grounded planning, tool-driven action, reflective validation, and evidence-based explanation—yielding trustworthy, inspectable power system analytics through natural language interaction.

3.2.2 *Contingency Analysis Agent.*

The contingency analysis agent systematically evaluates grid reliability by simulating T-1 outages for each transmission element. It executes a structured workflow: modify topology, invoke power flow solvers (e.g., *runpp* in pandapower), validate solutions, and interpret stress patterns. The agent maintains memory of recurring bottlenecks and voltage vulnerabilities, distinguishing between minor single overloads and outages causing multiple simultaneous violations. Rather than fabricating numbers, it anchors every cited metric to structured solver outputs with full provenance. When ranking competing contingencies, it provides auditable narratives comparing thermal congestion, voltage risks, and potential load shedding requirements. This transforms brute-force T-1 enumeration into interactive diagnostics that identify critical elements and suggest targeted mitigation strategies.

3.2.3 *Critical Element Identification.*

The contingency analysis agent uses LLM reasoning to synthesize structured solver outputs (line loading percentages, voltage deviations, load curtailment) into ranked critical transmission elements. Rather than using single metrics, the LLM analyzes evidence from tools: clusters of thermal overloads exceeding 110-115% ratings, voltage excursions below 0.94 p.u., and redispatch triggering load shedding. For instance, removing Line 54-59 in IEEE 118 forcing two lines to 118% and 121% loading while dropping a bus to 0.928 p.u. elevates criticality due to combined thermal and voltage stress. The agent transparently articulates comparisons: "*Outage A causes three overloads requiring 12 MW curtailment, while Outage B causes one marginal overload—therefore A ranks higher.*" This generates actionable recommendations for capacity reinforcement, reactive support, or remedial switching. By grounding ranking steps in validated numerical context, the LLM avoids hallucination and produces auditable justifications supporting operational decision-making.

Abridged dialogue showcase with steps in agents. Please check Appendix D for complete examples.

```
User : Solve IEEE 118.
Agent: 1. (understand the case to be solved) -> reasoning
```

```
      2. (extract relevant parameters)     -> reasoning
      3. (plan solution strategy)          -> reasoning
      4. (invoke ACOPF solver)        -> function tools
      5. (validate results)           -> function tools
      6. (narrate findings)                 -> response
User: Increase the load for bus 10 to 50MW
Agent: 1. (understand the task to solve)    -> reasoning
      2. (retrieve current net status)       -> context
      3. (prepare data for tools)          -> reasoning
      4. (invoke ACOPF solver again)  -> function tools
      5. (validate results)           -> function tools
      6. (summary)                          -> response
User: what's the most critical contingencies in this
      network
Agent: 1. (understand task)                -> reasoning
       2. (shift from ACOPF agent to AC agent)
                                    -> shared context
      3. (run contingency analysis)   -> function tools
       4. ...
```

3.3 Agent Context Management

Data Models and Type Safety. **GridMind** relies on a set of strongly typed, structured data models to keep every agent grounded in the same verifiable representation of the power system and its analytical state. Instead of passing loosely defined dictionaries, the system serializes and validates each object—network snapshots, optimization solutions, contingency outcomes, and shared conversational context—against explicit schemas. A unified `PowerSystem` model captures buses, generators, branches, transformers, and metadata; an `ACOPFSolution` object stores total cost, dispatch, losses, constraint margins, and convergence flags; a `ContingencyAnalysisResult` aggregates per-outage thermal and voltage impacts plus criticality narratives; an `AgentContext` records the active case, cached solutions, applied modifications, and provenance; and a `WorkflowState` traces multi-step analytical plans and their completion status. These structured types do more than enforce shape—they provide semantic anchors the LLM can reference when planning actions, summarizing results, or deciding whether prior computations can be reused. By surfacing field names (e.g., total_cost, max_thermal_loading, min_voltage_pu) directly in tool outputs, the model maps natural language intents ("compare post-contingency voltage minima") to exact data fields, reducing ambiguity and suppressing hallucinated attributes. Type validation also acts as a safety net: malformed or incomplete tool returns trigger automatic recovery paths instead of silently corrupting downstream reasoning. In practice, this schema layer converts free-form dialogue into a controlled loop of intent, structured call, validated result, grounded explanation.

3.4 Cross-Agent Context Management

Agents collaborate through a single structured, versioned session state capturing: (a) the active network plus incremental diffs (load shifts, outages, parameter edits), (b) validated numerical artifacts (latest feasible ACOPF solution, cached power-flow / per-contingency snapshots), and (c) provenance (solver options, timestamps, tool versions, validation flags). After a solve, the ACOPF agent deposits a typed `ACOPFSolution` (dispatch, objective cost, losses, voltage extrema, branch loadings, constraint margins) instead of only prose. The CA agent inspects freshness against the diff log to decide whether it can reuse that base point or must trigger a selective re-solve.

Cross-agent transfer is strictly schema-bound (`ACOPFSolution`, `ContingencyResultSet`) so planning references concrete fields (e.g., base_objective_cost, min_voltage_pu, max_thermal_loading) and avoids hallucination. Each outage evaluation is cached under a composite key (case + outage + diff hash); criticality ranking streams those cached records to detect recurring overload corridors or voltage depressions and writes back ranked justifications as structured data plus a derived human summary. A normalized change log appends every modification; agents replay only relevant diffs to reconstruct required state. Session persistence serializes baseline, diffs, artifacts, contingency cache, and rankings for seamless resumption. This disciplined produce-validate-consume loop lets compound requests ("solve, assess T-1 risk, rank reinforcements") execute efficiently, coherently, and reproducibly.

4 Experimental Evaluation

We conducted comprehensive experiments to evaluate **GridMind** 's agentic capabilities across multiple dimensions: technical accuracy for domain tasks, efficiency of agents by evaluating the response time from LLMs, and robustness under various input perturbations. We implemented the agentic workflow using PydanticAI [24], and systematically evaluate the agentic performance based on a set of LLMs including OpenAI's GPT-5, GPT-5-mini, GPT-5-nano, GPT-o3, GPT-o4-mini, and Anthropic's Claude 4 Sonnet. The backend power system solvers are implemented using PandaPower [25]. We deployed the system on a local machine with 32-core Intel Xeon Gold 5317 CPU, 256GB RAM. The LLMs are accessed through either remote APIs (e.g., OpenAI, Anthropic, proxy server Argo). Our evaluation employed standard IEEE test cases representing different system scales and complexity levels, IEEE 14, 30, 118, and 300-bus systems [6].

4.1 ACOPF Agent

The evaluation of **GridMind** 's ACOPF agent demonstrates remarkable consistency in solution quality across different language models while revealing significant variations in computational efficiency. Figure 3 presents a comprehensive performance overview with three key metrics: success rates, execution time distributions, and execution time versus case complexity.

The left figure shows that all tested models achieve 100% success rates across IEEE test case118, demonstrating that function-calling capabilities enable reliable technical accuracy regardless of the underlying LLM. This consistent performance validates our hypothesis that deterministic solver invocation through structured tool calls maintains numerical rigor even with smaller language models.

The middle figure reveals substantial differences in execution time distributions across models with case118 by running 5 times. GPT-o4-mini exhibits the most variable performance with execution times less than 10 seconds. In contrast, most recent GPT-5, GPT-5-mini, GPT-5 Nano and Claude 4 Sonnet demonstrate more time to reason, potentially due to their large models and complex reasoning processes. This is interesting to note that with function tool capabilities, there is a potential trade-off between model size

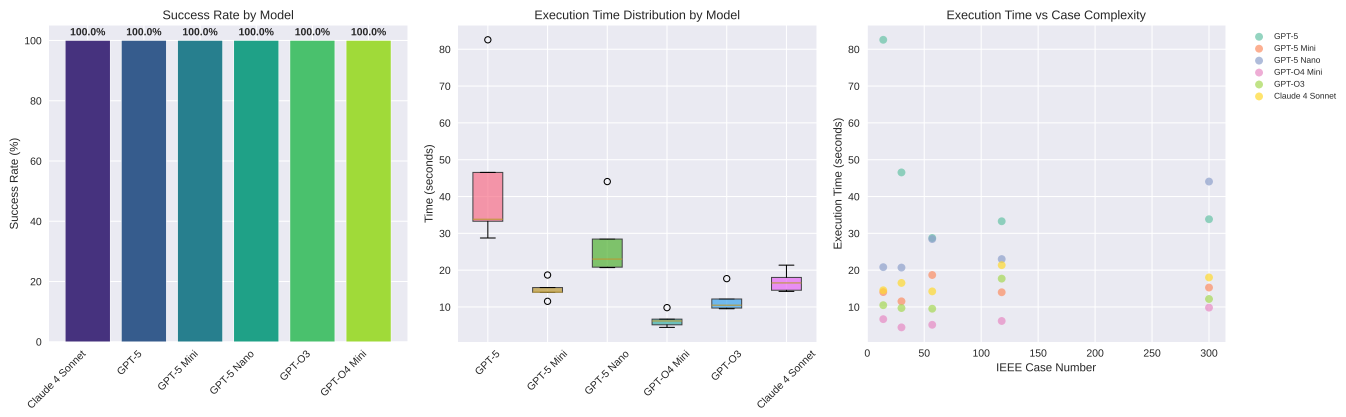


Figure 3: Agent performance in terms of accuracy and execution time

and reasoning speed, as smaller models can achieve comparable accuracy with lower latency.

The right figure illustrates the relationship between system complexity (measured by IEEE case number as a proxy for network size) and execution time. With the increase of case size, there is no significant trends of time to get the solution, indicating that the agentic framework effectively manages complexity without incurring additional computational costs from LLMs. However, calling the tools for specific case is still subject to the case size and complexity of its original problem, e.g., ACOPF.

These findings highlight that agentic scientific computing can achieve both technical rigor and computational economy by leveraging function-calling architectures that combine efficient language models for workflow orchestration with validated domain-specific tools for numerical calculations.

4.2 T-1 Contingency Analysis Agent

Unlike the ACOPF agent that can compare the solution directly from a reference solver, the CA agent cannot retrieve a ground solution for the problem. CA, as a result, relies on the LLM's ability to reason about potential contingencies and their impacts on system reliability. CA agent will first calculate the power flow for the base case, without any contingencies, and then iteratively apply each identified contingency to assess its impact on system reliability. Therefore, we conducted a series of experiments based on different LLMs to identify the top-5 most critical lines, and let agent applies tools (power flow solver) to calculate the maximum overload percentage for each contingency. The results are summarized in Table 1. The results show significant variation in computational efficiency,

Table 1: CA Agent Performance (case118)

Model	Time (s)	Critical Lines (idx)	Max Overload %
GPT-5	92.7	6, 7, 0, 171, 49	137
GPT-5 Mini	24.8	7, 0, 171, 49, 9	165
GPT-5 Nano	26.2	6, 7, 0, 171, 49	137
GPT-o4 Mini	34.2	6, 7, 0, 171, 49	137
GPT-o3	24.6	6, 7, 0, 171, 49	137
Claude 4 Sonnet	63.3	6, 7, 0, 171, 49	137

with GPT-5 Mini and GPT-o3 achieving the fastest execution times at approximately 24-25 seconds, while GPT-5 required the longest processing time at 92.7 seconds. Interestingly, most models identified the same set of critical transmission lines (indices 6, 7, 0, 171, 49) and achieved identical maximum overload percentages of 137%, suggesting consistent analytical accuracy across different AI architectures. The notable exception is GPT-5 Mini, which identified a slightly different set of critical lines (replacing index 6 with index 9) and reported a higher maximum overload of 165%, indicating either a different analytical approach or potentially identifying additional stress conditions in the power system that other models may have missed. This is mainly due to the architectural and training differences from various LLMs and their build-in reasoning nature.

Discussion. GridMind demonstrates how agentic AI can transform scientific computing by converting fragmented workflows into fluent conversational interfaces. Our prototype shows that LLMs, when bound to deterministic tools and structured state management, can orchestrate multi-step analyses while maintaining numerical rigor.

Addressing LLM Hallucination. A critical concern with LLM-based agents is hallucination—generating plausible but incorrect information, e.g., asking of case30 but resulting in case300. **GridMind** mitigates this through: (1) structured function calls that prevent numerical fabrication, (2) rigorous data validation using Pydantic schemas, (3) grounding all quantitative claims in solver outputs, and (4) comprehensive result verification before response generation. This architecture ensures that while agents may hallucinate in reasoning steps, all reported numerical results originate from validated computational tools. Future extensions include expanding to adjacent domains, implementing richer coordination protocols, and developing human-AI interaction policies for critical engineering applications. These results position agentic frameworks as organizing principles for scientific computing ecosystems that harmonize expressiveness, rigor, and accessibility.

5 Conclusion

This paper introduced **GridMind**, a groundbreaking multi-agent AI system that demonstrates the transformative potential of agentic workflows in scientific engineering. Through the seamless integration of specialized agents for ACOPF optimization and T-1 contingency analysis, our system represents a paradigm shift from tool-based to conversational AI-driven scientific computing. Our key contributions demonstrate that agentic AI represents a viable and transformative paradigm for complex scientific workflows. The multi-agent architecture successfully coordinates sophisticated power system analyses through natural language interfaces while preserving the numerical precision essential for engineering applications. Notably, our evaluation reveals that the choice of underlying LLM significantly influences both system performance and reliability. Contrary to expectations, smaller language models can achieve competitive analytical accuracy with substantially reduced reasoning latency, suggesting that the most advanced model is not necessarily optimal for all agentic scientific computing tasks.

Acknowledgments

This material is based upon work supported by Laboratory Directed Research and Development (LDRD) funding from Argonne National Laboratory, provided by the Director, Office of Science, of the U.S. Department of Energy under contract DE-AC02-06CH11357. This research used resources of the Argonne Leadership Computing Facility at Argonne National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02-06CH11357.

Table 2: Test cases

Case	Bus	Gen	Load	AC line	Transformers
IEEE 14	14	5	11	17	3
IEEE 30	30	6	21	41	4
IEEE 57	57	7	42	63	17
IEEE 118	118	54	99	175	11
IEEE 300	300	68	193	283	128

References

[1] MA Abido. 2002. Optimal power flow using particle swarm optimization. *International Journal of Electrical Power & Energy Systems* 24, 7 (2002), 563–571.

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[3] Jacques Carpentier. 1962. Contribution a l'etude du dispatching economique. *Bull. Soc. Fr. Elec. Ser.* 3 (1962), 431.

[4] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374* (2021).

[5] Carleton Coffrin, Russell Bent, Kaarthik Sundar, Yeesian Ng, and Miles Lubin. 2018. Powermodels. jl: An open-source framework for exploring power flow formulations. In *2018 Power Systems Computation Conference (PSCC)*. IEEE, 1–8.

[6] University of Washinton Dept. of Electrical Engineering. 1999. Power systems test case archive. https://labs.ece.uw.edu/pstca/ Accessed: 2025-06-01.

[7] Yangruibo Ding, Zijian Wang, Wasi Ahmad, Hantian Ding, Ming Tan, Nihal Jain, Murali Krishna Ramanathan, Ramesh Nallapati, Parminder Bhatia, Dan Roth, et al. 2023. Crosscodeeval: A diverse and multilingual benchmark for cross-file code completion. *Advances in Neural Information Processing Systems* 36 (2023), 46701–46723.

[8] Ian Dobson, Benjamin A Carreras, Vickie E Lynch, and David E Newman. 2007. Complex systems analysis of series of blackouts: Cascading failure, critical points, and self-organization. *Chaos: An Interdisciplinary Journal of Nonlinear Science* 17, 2 (2007).

[9] GC Ejebe and BF Wollenberg. 2007. Automatic contingency selection. *IEEE transactions on Power Apparatus and Systems* 1 (2007), 97–109.

[10] Shanshan Han, Qifan Zhang, Yuhang Yao, Weizhao Jin, and Zhaozhuo Xu. 2024. LLM multi-agent systems: Challenges and open problems. *arXiv preprint arXiv:2402.03578* (2024).

[11] M Huneault and Francisco D Galiana. 2002. A survey of the optimal power flow literature. *IEEE transactions on Power Systems* 6, 2 (2002), 762–770.

[12] Javad Lavaei and Steven H Low. 2011. Zero duality gap in optimal power flow problem. *IEEE Transactions on Power systems* 27, 1 (2011), 92–107.

[13] Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. 2023. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161* (2023).

[14] Xingpeng Li, Pranavamoorthy Balasubramanian, Mostafa Sahraei-Ardakani, Mojdeh Abdi-Khorsand, Kory W Hedman, and Robin Podmore. 2016. Real-time contingency analysis with corrective transmission switching. *IEEE Transactions on Power Systems* 32, 4 (2016), 2604–2617.

[15] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. 2023. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688* (2023).

[16] Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. 2022. Codegen: An open large language model for code with multi-turn program synthesis. *arXiv preprint arXiv:2203.13474* (2022).

[17] OpenAI. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[18] Giuliano Andrea Pagani and Marco Aiello. 2013. The power grid as a complex network: a survey. *Physica A: Statistical Mechanics and its Applications* 392, 11 (2013), 2688–2700.

[19] Mania Pavella, Damien Ernst, and Daniel Ruiz-Vega. 2012. *Transient stability of power systems: a unified approach to assessment and control.* Springer Science & Business Media.

[20] Victor H Quintana, Geraldo L Torres, and Jose Medina-Palomo. 2000. Interior-point methods and their applications to power systems: a classification of publications and software codes. *IEEE Transactions on power systems* 15, 1 (2000), 170–176.

[21] Mostafa Sahraei-Ardakani, Xingpeng Li, P Balasubramanian, KW Hedman, and Mojdeh Abdi-Khorsand. 2015. Real-time contingency analysis with transmission switching on real power system data. *IEEE Transactions on Power Systems* 31, 3 (2015), 2501–2502.

[22] Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Michael Moor, Zicheng Liu, and Emad Barsoum. 2025. Agent laboratory: Using llm agents as research assistants. *arXiv preprint arXiv:2501.04227* (2025).

[23] LangChain Team. 2024. LangChain: A framework for building applications with LLMs. https://www.langchain.com/ Accessed: 2025-08-01.

[24] PydanticAI Team. 2024. PydanticAI: A framework for building production-grade AI applications. https://ai.pydantic.dev/ Accessed: 2025-08-01.

[25] Leon Thurner, Alexander Scheidler, Florian Schäfer, Jan-Hendrik Menke, Julian Dollichon, Friederike Meier, Steffen Meinecke, and Martin Braun. 2018. pandapower—an open-source python tool for convenient modeling, analysis, and optimization of electric power systems. *IEEE Transactions on Power Systems* 33, 6 (2018), 6510–6521.

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

[27] Michael Wooldridge. 2009. *An introduction to multiagent systems.* John wiley & sons.

[28] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. *arXiv preprint arXiv:2308.08155* 3, 4 (2023).

[29] Xuan Wu and Antonio J Conejo. 2019. Security-constrained ACOPF: Incorporating worst contingencies and discrete controllers. *IEEE Transactions on Power Systems* 35, 3 (2019), 1936–1945.

[30] Ray Daniel Zimmerman, Carlos Edmundo Murillo-Sánchez, and Robert John Thomas. 2010. MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education. *IEEE Transactions on power systems* 26, 1 (2010), 12–19.

A Case Studies

B Implementation Details

B.1 System Requirements

- Python 3.10 or higher
- PydanticAI (version 0.7 or higher) framework for agents, function tools, and context management
- PandaPower for power system simulation and analysis
- NumPy and SciPy for numerical computations
- Rich library for CLI interface formatting

- PandaPower (runpp for powerflow) for line contingencies.

B.2 Supported IEEE Test Cases

Table 2 summarizes the supported IEEE test cases, highlighting their key characteristics and complexities.

B.3 Construct Agent

B.3.1 *ACOPF Agent.*

System prompt. Shown in Figure 4

```
You are an expert ACOPF (AC Optimal Power Flow)
     agent for power system analysis.

Your capabilities include:
1. Solving ACOPF problems for standard IEEE test
     cases (14, 30, 57, 118, 300 bus systems)
2. Modifying system parameters (loads, generation
     limits, etc.) and re-solving
3. Validating solutions by checking power flows,
     voltage limits, and line loadings
4. Assessing solution quality and providing
     recommendations
5. Engaging in conversational interactions about
     power system optimization

You have access to the following tools:
- solve_acopf_case: Load and solve an IEEE test case
- modify_bus_load: Modify load at a specific bus and
      re-solve
- get_network_status: Get current network and
     solution status

When users ask to solve a case, use the
     solve_acopf_case tool with the case name.
When users ask to modify loads, use the
     modify_bus_load tool with the specified
     parameters.
When users ask about current status, use the
     get_network_status tool.

Always provide clear explanations of results,
     including objective values and any constraint
     violations.
Be professional, accurate, and educational in your
     responses.
```

Figure 4: ACOPF Agent System Prompt

Function Tools: available tools for ACOPF agent

```
ACOPF Agent:
    solve_acopf_case
    modify_bus_load
    get_network_status
```

B.3.2 *Contingency Analysis Agent.*

System Prompt. Shown in Figure 5

Function Tools: available tools for CA agent

```
CA Agent:
    solve_base_case
    run_n1_contingency_analysis
    analyze_specific_contingency
    get_contingency_status
```

```
You are an expert Contingency Analysis agent for
     power system reliability assessment.

Your capabilities include:
1. Solving base case ACOPF problems for standard
     IEEE test cases
2. Running comprehensive N-1 contingency analysis
3. Analyzing specific contingencies (line outages,
     transformer outages)
4. Identifying critical contingencies and system
     vulnerabilities
5. Assessing voltage violations and equipment
     overloads
6. Providing recommendations for system
     reinforcement

You have access to the following tools:
- solve_base_case: Load and solve base case before
     contingency analysis
- run_n1_contingency_analysis: Run comprehensive N-1
      analysis
- analyze_specific_contingency: Analyze a specific
     element outage
- get_contingency_status: Get current analysis
     status and results

When users ask to analyze contingencies, first
     ensure a base case is solved, then run the
     appropriate analysis.
Always provide clear explanations of critical
     contingencies, violations, and recommendations.
Be professional, accurate, and focus on system
     reliability and security.
```

Figure 5: Contingency Analysis Agent System Prompt

B.4 Multi-Agent Workflow Capabilities

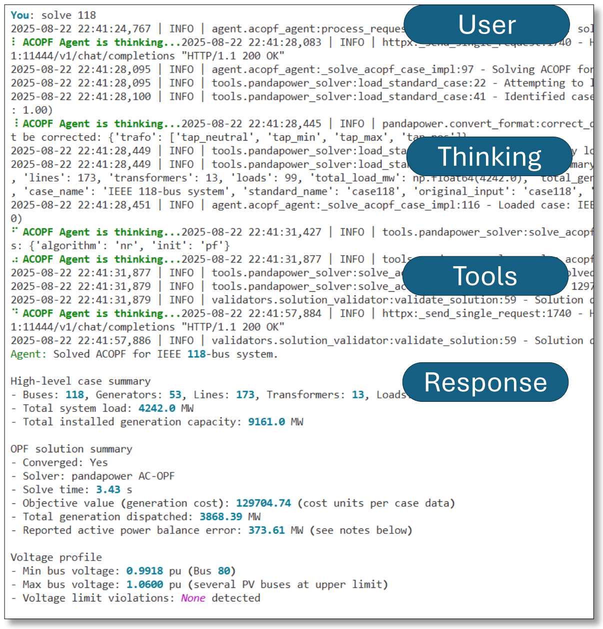
The system supports sophisticated analytical sequences including:

- Sequential analysis workflows (ACOPF followed by contingency analysis)
- Comparative studies (economic vs. security-constrained operation)
- Sensitivity analysis (parameter modifications with impact assessment)
- Cross-domain result correlation and insight generation

C Data schema

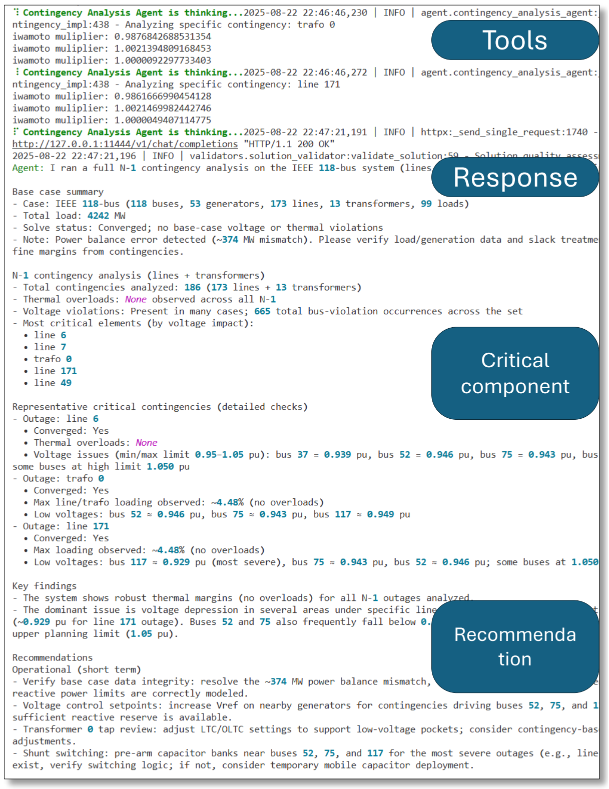
Illustrative schema fragment:

```
class ACOPFSolution(BaseModel):
    case_name: str
    solved: bool
    objective_cost: float
    gen_dispatch_mw: dict[str, float]
    branch_loading: list[BranchLoading]
    min_voltage_pu: float
    max_voltage_pu: float
    convergence_message: str

class SolutionQuality(BaseModel):
  overall_score: float = Field(ge=0.0, le=10.0)
```



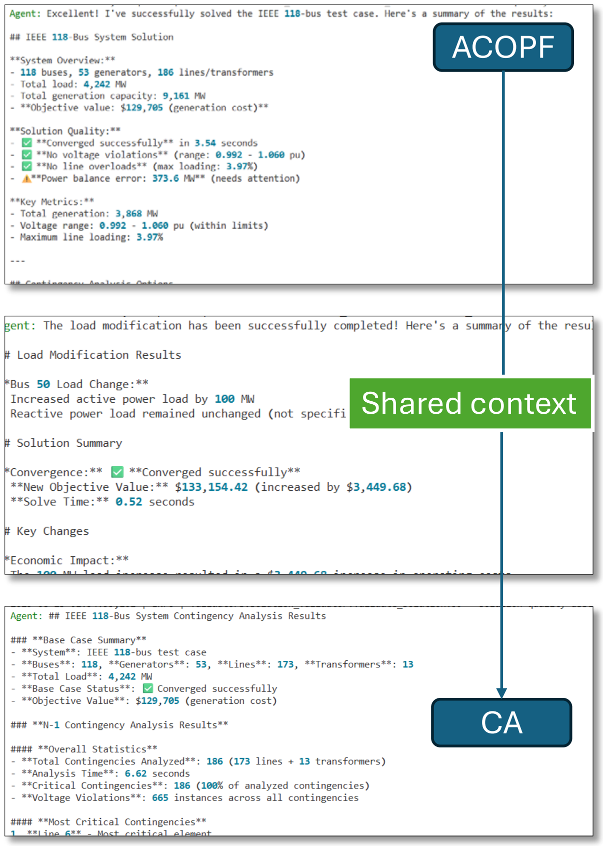
```
You: solve 118
2025-08-22 22:41:24,767 | INFO | agent.acopf_agent:process_reques                    sol
⠋ ACOPF Agent is thinking...2025-08-22 22:41:28,083 | INFO | httpx:
1:11444/v1/chat/completions "HTTP/1.1 200 OK"
2025-08-22 22:41:28,095 | INFO | agent.acopf_agent:_solve_acopf_case_impl:97 - Solving ACOPF for
2025-08-22 22:41:28,095 | INFO | tools.pandapower_solver:load_standard_case:22 - Attempting to l
2025-08-22 22:41:28,100 | INFO | tools.pandapower_solver:load_standard_case:41 - Identified case
: 1.00)
⠇ ACOPF Agent is thinking...2025-08-22 22:41:28,445 | INFO | pandapower.convert_format:correct_d
t be corrected: {'trafo': ['tap_neutral', 'tap_min', 'tap_max', 'ta
2025-08-22 22:41:28,449 | INFO | tools.pandapower_solver:load_sta                          y lo
2025-08-22 22:41:28,449 | INFO | tools.pandapower_solver:load_sta                          mary
, 'lines': 173, 'transformers': 13, 'loads': 99, 'total_load_mw': np.float64(4242.0), 'total_gen
, 'case_name': 'IEEE 118-bus system', 'standard_name': 'case118', 'original_input': 'case118', '
2025-08-22 22:41:28,451 | INFO | agent.acopf_agent:_solve_acopf_case_impl:116 - Loaded case: IEE
0)
⠏ ACOPF Agent is thinking...2025-08-22 22:41:31,427 | INFO | tools.pandapower_solver:solve_acopf
s: {'algorithm': 'nr', 'init': 'pf'}
⠼ ACOPF Agent is thinking...2025-08-22 22:41:31,877 | INFO | tools                         acopf
2025-08-22 22:41:31,877 | INFO | tools.pandapower_solver:solve_a                           olved
2025-08-22 22:41:31,879 | INFO | tools.pandapower_solver:solve_ac                          1297
2025-08-22 22:41:31,879 | INFO | validators.solution_validator:validate_solution:59 - Solution d
⠏ ACOPF Agent is thinking...2025-08-22 22:41:57,884 | INFO | httpx:_send_single_request:1740 - H
1:11444/v1/chat/completions "HTTP/1.1 200 OK"
2025-08-22 22:41:57,886 | INFO | validators.solution_validator:validate_solution:59 - Solution d
Agent: Solved ACOPF for IEEE 118-bus system.

High-level case summary
- Buses: 118, Generators: 53, Lines: 173, Transformers: 13, Loads
- Total system load: 4242.0 MW
- Total installed generation capacity: 9161.0 MW

OPF solution summary
- Converged: Yes
- Solver: pandapower AC-OPF
- Solve time: 3.43 s
- Objective value (generation cost): 129704.74 (cost units per case data)
- Total generation dispatched: 3868.39 MW
- Reported active power balance error: 373.61 MW (see notes below)

Voltage profile
- Min bus voltage: 0.9918 pu (Bus 80)
- Max bus voltage: 1.0600 pu (several PV buses at upper limit)
- Voltage limit violations: None detected
```

Figure 7: ACOPF Agent Workflow



```
⠏ Contingency Analysis Agent is thinking...2025-08-22 22:46:46,230 | INFO | agent.contingency_analysis_agent:
ntingency_impl:438 - Analyzing specific contingency: trafo 0
iwamoto muliplier: 0.9876842688531354
iwamoto muliplier: 1.0021394809168453
iwamoto muliplier: 1.0000092297733403
⠇ Contingency Analysis Agent is thinking...2025-08-22 22:46:46,272 | INFO | agent.contingency_analysis_agent:
ntingency_impl:438 - Analyzing specific contingency: line 171
iwamoto muliplier: 0.9861666990454128
iwamoto muliplier: 1.0021469982442746
iwamoto muliplier: 1.0000049407114775
⠏ Contingency Analysis Agent is thinking...2025-08-22 22:47:21,191 | INFO | httpx:_send_single_request:1740 -
http://127.0.0.1:11444/v1/chat/completions "HTTP/1.1 200 OK"
2025-08-22 22:47:21,196 | INFO | validators.solution_validator:validate_solution:59 - Solution quality assess
Agent: I ran a full N-1 contingency analysis on the IEEE 118-bus system (lines

Base case summary
- Case: IEEE 118-bus (118 buses, 53 generators, 173 lines, 13 transformers, 99 loads)
- Total load: 4242 MW
- Solve status: Converged; no base-case voltage or thermal violations
- Note: Power balance error detected (~374 MW mismatch). Please verify load/generation data and slack treatme
fine margins from contingencies.

N-1 contingency analysis (lines + transformers)
- Total contingencies analyzed: 186 (173 lines + 13 transformers)
- Thermal overloads: None observed across all N-1
- Voltage violations: Present in many cases; 665 total bus-violation occurrences across the set
- Most critical elements (by voltage impact):
  • line 6
  • line 7
  • trafo 0
  • line 171
  • line 49

Representative critical contingencies (detailed checks)
- Outage: line 6
  • Converged: Yes
  • Thermal overloads: None
  • Voltage issues (min/max limit 0.95–1.05 pu): bus 37 = 0.939 pu, bus 52 = 0.946 pu, bus 75 = 0.943 pu, bus
some buses at high limit 1.050 pu
- Outage: trafo 0
  • Converged: Yes
  • Max line/trafo loading observed: ~4.48% (no overloads)
  • Low voltages: bus 52 ≈ 0.946 pu, bus 75 ≈ 0.943 pu, bus 117 ≈ 0.949 pu
- Outage: line 171
  • Converged: Yes
  • Max loading observed: ~4.48% (no overloads)
  • Low voltages: bus 117 ≈ 0.929 pu (most severe), bus 75 ≈ 0.943 pu, bus 52 ≈ 0.946 pu; some buses at 1.050

Key findings
- The system shows robust thermal margins (no overloads) for all N-1 outages analyzed.
- The dominant issue is voltage depression in several areas under specific line
(~0.929 pu for line 171 outage). Buses 52 and 75 also frequently fall below 0.
upper planning limit (1.05 pu).

Recommendations
Operational (short term)
- Verify base case data integrity: resolve the ~374 MW power balance mismatch,
reactive power limits are correctly modeled.
- Voltage control setpoints: increase Vref on nearby generators for contingencies driving buses 52, 75, and 1
sufficient reactive reserve is available.
- Transformer 0 tap review: adjust LTC/OLTC settings to support low-voltage pockets; consider contingency-bas
adjustments.
- Shunt switching: pre-arm capacitor banks near buses 52, 75, and 117 for the most severe outages (e.g., line
exist, verify switching logic; if not, consider temporary mobile capacitor deployment.
```

Figure 8: Contingency Analysis Agent Workflow



```
Agent: Excellent! I've successfully solved the IEEE 118-bus test case. Here's a summary of the results:

## IEEE 118-Bus System Solution

**System Overview:**
- 118 buses, 53 generators, 186 lines/transformers
- Total load: 4,242 MW
- Total generation capacity: 9,161 MW
- **Objective value: $129,705 (generation cost)**

**Solution Quality:**
- ✅ **Converged successfully** in 3.54 seconds
- ✅ **No voltage violations** (range: 0.992 - 1.060 pu)
- ✅ **No line overloads** (max loading: 3.97%)
- ⚠️**Power balance error: 373.6 MW** (needs attention)

**Key Metrics:**
- Total generation: 3,868 MW
- Voltage range: 0.992 - 1.060 pu (within limits)
- Maximum line loading: 3.97%

---
```

```
gent: The load modification has been successfully completed! Here's a summary of the resu

# Load Modification Results

*Bus 50 Load Change:**
 Increased active power load by 100 MW
 Reactive power load remained unchanged (not specifi

# Solution Summary

*Convergence:** ✅ **Converged successfully**
 **New Objective Value:** $133,154.42 (increased by $3,449.68)
 **Solve Time:** 0.52 seconds

# Key Changes

*Economic Impact:**
```

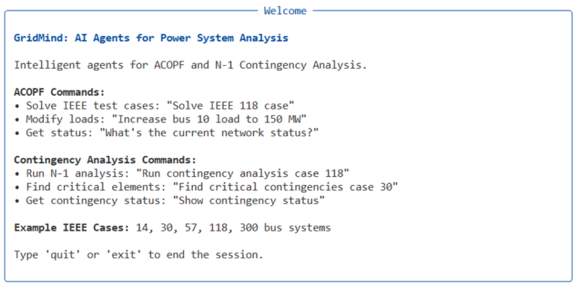
```
Agent: ## IEEE 118-Bus System Contingency Analysis Results

### **Base Case Summary**
- **System**: IEEE 118-bus test case
- **Buses**: 118, **Generators**: 53, **Lines**: 173, **Transformers**: 13
- **Total Load**: 4,242 MW
- **Base Case Status**: ✅ Converged successfully
- **Objective Value**: $129,705 (generation cost)

### **N-1 Contingency Analysis Results**

#### **Overall Statistics**
- **Total Contingencies Analyzed**: 186 (173 lines + 13 transformers)
- **Analysis Time**: 6.62 seconds
- **Critical Contingencies**: 186 (100% of analyzed contingencies)
- **Voltage Violations**: 665 instances across all contingencies

#### **Most Critical Contingencies**
```

Figure 9: Multi-Agent Workflow Example

```
convergence_quality:float = Field(ge=0.0, le=10.0)
constraint_satisfaction: float =
        Field(ge=0.0, le=10.0)
economic_efficiency:float = Field(ge=0.0, le=10.0)
system_security: float = Field(ge=0.0, le=10.0)
detailed_metrics: Dict[str, Any] =
        Field(default_factory=dict)
recommendations: List[str] =
        Field(default_factory=list)
```

D Example Agentic Workflows

D.1 CLI interface



```
Welcome

GridMind: AI Agents for Power System Analysis

Intelligent agents for ACOPF and N-1 Contingency Analysis.

ACOPF Commands:
• Solve IEEE test cases: "Solve IEEE 118 case"
• Modify loads: "Increase bus 10 load to 150 MW"
• Get status: "What's the current network status?"

Contingency Analysis Commands:
• Run N-1 analysis: "Run contingency analysis case 118"
• Find critical elements: "Find critical contingencies case 30"
• Get contingency status: "Show contingency status"

Example IEEE Cases: 14, 30, 57, 118, 300 bus systems

Type 'quit' or 'exit' to end the session.
```

Figure 6: Conversational interface (CLI prototype) driving multi-agent analysis.

D.2 Single-Domain Analysis Examples

D.2.1 ACOPF Agent. Figure 7 shows the ACOPF agent for solving optimal power flow problems.

D.2.2 Contingency Analysis. Figure 8 shows the contingency analysis agent for T-1 reliability assessment.

D.3 Multi-Agent Workflow Examples

Figure 9 shows the agentic workflow, particularly from ACOPF agent to CA agent with shared context.

```
User:
Solve IEEE 118 case, then run contingency analysis
and identify critical elements for reinforcement
```