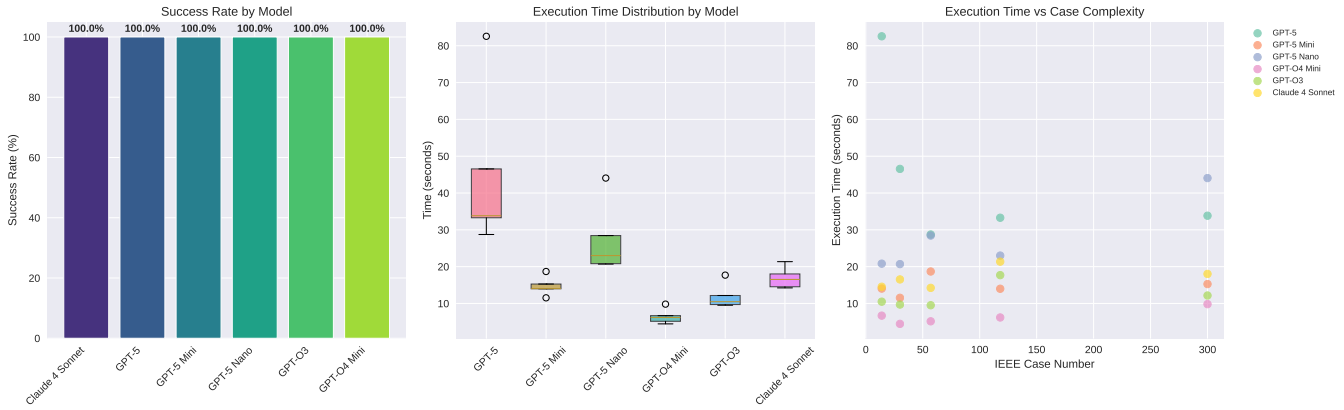

Figure 3: Agent performance in terms of accuracy and execution time

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 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.

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

The results show significant variation in computational efficiency, 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. **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