AI Post-Mortem Report

Born Rule Breakers:
Steve Cao, Haadi Khan, Rhik Mazumder, Alice Wang

January 2026

1 Workflow

Our agentic workflow closely followed the Project Requirements Document. We organized our agents as follows:

- **MCP Server Development:** Agent Alan was tasked with creating a custom Model Context Protocol (MCP) server. Alan web crawled the `CUDA-Q` documentation, then semantically chunked and embedded the text using OpenAI's `embedding-3-small` model, storing these embeddings in a ChromaDB vector database. We then built MCP tools to query the database effectively and accurately.
- **Multi-Agent Research Team:** Once we had a working MCP server, lead agent Kole orchestrated a team of specialized subagents. Kole had access to all files involved in our research plan, including provided Jupyter Notebooks and referenced papers. The subagents were:
- **Isabella:** Created the initial implementation of the seed PCE using scientific literature as context
- **Tanish:** Responsible for verification (described below)
- **Ethan:** Performed hardware optimization, including GPU acceleration tasks

Additionally, we utilized a multi-agent **"consortium of experts"** as a problem-solving tool when exploring different approaches to optimize PCE parameters. All agents utilized Claude Sonnet 4.5.

Verification Strategy. Verification was primarily performed by subagent Tanish serving as **LLM as a judge**, who had access to our verification sources consisting of known solutions to the LABS problem up to $N \leq 82$. To guard against AI-introduced logic errors or hallucinated behavior, we implemented two explicit unit tests in `tests.py`, each anchored to externally verified ground-truth data:

1. **Energy function validation (deterministic unit test):** We validated the correctness of our energy evaluation function by applying it to the known optimal Low Autocorrelation Binary Sequences (LABS) for lengths $N = 1, \ldots, 82$, as reported in the reference table of known solutions. For each sequence, we computed its energy using our implementation and asserted exact agreement with the tabulated energies. This test catches implementation errors such as incorrect autocorrelation computation, indexing mistakes, or sign/convention mismatches.

2. **MTS algorithm validation (stochastic but bounded unit test):** To validate our MTS heuristic implementation, we tested whether it could recover known optimal energies for LABS instances of size $N = 1, \ldots, 20$. For each N, we ran MTS for at most 1000 generations and asserted that the best sequence found achieved the optimal energy documented in the literature. This test detects logical flaws in the MTS update rules, acceptance criteria, or stopping conditions that could otherwise produce superficially plausible but incorrect behavior.

2 "Vibe" Log

Win. The most essential use of AI in our project was the creation of the MCP server. Having an MCP server meant that writing code and debugging could be achieved purely through prompt-writing rather than personally understanding and interfacing between various tools and environments. Furthermore, using an agent to create the MCP server—a task that would have taken at least several hours of reading documentation without AI—became a relatively simple process that saved us significant time.

Fail. The "consortium of experts" failed when parsing papers on VQA optimization. When given an article on the implementation of EGT-CG, the agent incorrectly claimed that such an approach would be totally unfeasible for our application to the LABS problem and, moreover, failed to improve parameter optimization in general. However, based on our personal understanding of the paper, we believed this to be a hallucination.

Learn. We determined that this failure was caused by the agent believing it lacked sufficient context and therefore searching beyond our source files for information. We prevented this from recurring by explicitly instructing the agent to only consider our provided source files. This modification to our prompting strategy significantly improved the accuracy of subsequent responses.

Context Dump. To achieve the best results, we employed precise prompt engineering. For each agent we created a `skills.md` file with the following structure:

```
# Agent Skills and Context

You are an expert in quantum computing and combinatorial
optimization with specific expertise in [domain].

## Your Role
[Specific responsibilities]

## Available Resources
- CUDA-Q documentation (via MCP server)
- Research papers on [specific topics]
- Known LABS solutions for N <= 82

## Constraints
- Only use information from provided source files
- Validate all code against unit tests in tests.py
- [Additional domain-specific constraints]
```

This structured approach ensured agents remained focused on verified sources and adhered to our validation requirements.