2026-Nvidia-iQuHack-Challenge
AI Report

AutoQurelation

Kevin Joven[1], Abhishek Karna[2], Lang Ji[2], and Matthew Chen[2]

[1]North Carolina State University, Department of Electrical and Computer Engineering, Raleigh, NC, USA
[2]Duke University, Department of Physics, Durham, NC, USA

February 1, 2026

---

1 Team Roles and Responsibilities

Role	Name	GitHub Handle	Discord Handle
Project Lead	Kevin Joven	`KevinJoven11`	`kevinjoven`
GPU Acceleration PIC	Lang Ji	`lang-ji`	`_langji`
Quality Assurance PIC	Abhishek Karna	`abhishek-karna-duke`	`kronos_3`
Technical Marketing PIC	Matthew Chen	`mattchen10923`	`_thatasian`

2 Workflow

The project workflow was designed to combine targeted use of large language models with domain expertise and manual validation. Literature review and contextual grounding were conducted using *Perplexity Pro Deep Research*, which was used exclusively as a search and summarization engine. Its role was to identify relevant prior work, extract key results, and surface primary sources. All scientific claims and technical conclusions derived from this process were validated by direct inspection of the cited literature.

Code development was carried out using a combination of ChatGPT, Perplexity, and CODA as interactive coding assistants. These tools were used to accelerate drafting, refactoring, and documentation of classical and quantum algorithms. They were not used as autonomous agents, and no external API calls or self-directed agent loops were deployed. All generated code was reviewed line by line, edited, and integrated manually.

Algorithmic design decisions, parameter selections, and methodological trade-offs were guided by the team's prior experience with variational and adiabatic quantum algorithms, classical metaheuristics, and hybrid quantum–classical workflows. When ambiguity arose, preference was given to physically motivated constructions and empirically verifiable behavior rather than purely heuristic or black-box solutions.

Overall iteration followed a tight loop consisting of literature-informed design, AI-assisted code generation, manual review and correction, automated testing, and empirical evaluation. This ensured that AI tools functioned as accelerators of productivity rather than substitutes for scientific judgment. Reproducibility and avoidance of black-box behavior were treated as first-order requirements throughout.

3 Verification Strategy

All AI-assisted code was subjected to explicit verification before acceptance. Verification combined manual inspection with automated unit testing designed specifically to catch hallucinations, silent logic errors, and incorrect assumptions commonly produced by large language models.

Unit tests were written to validate boundary conditions, conservation properties, and invariants implied by the underlying algorithms. For example, in variational quantum algorithm implementations, tests verified that cost functions returned known analytical values for trivial or exactly solvable instances. Parameterized tests were used to ensure consistent behavior under small perturbations of inputs, which helped detect incorrect indexing, sign errors, or misuse of library primitives introduced during AI-assisted refactoring.

In classical optimization components, unit tests checked monotonicity properties, feasibility constraints, and convergence behavior on toy problems with known optima. Mock inputs were also used to confirm that AI-generated code did not rely on undefined variables, undocumented side effects, or implicit global state. Any test failure triggered manual inspection and revision, and no AI-generated code was merged without passing the full test suite.

4 The Vibe Log

4.1 Win

AI assistance saved multiple hours during the initial literature review phase. Perplexity Pro Deep Research rapidly surfaced relevant papers spanning quantum optimization, hybrid algos, and benchmark methodologies, along with concise summaries and citations. This reduced what would normally be a multi-day manual search process to a few focused hours, allowing the team to allocate more time to algorithmic design and experimentation.

4.2 Learn

Early interactions with coding assistants produced verbose but poorly structured implementations. This was improved by altering the prompting strategy to include explicit constraints, expected input–output behavior, and references to existing internal abstractions. Providing short context blocks describing the intended algorithmic role of each function significantly improved correctness and reduced downstream refactoring. *They also helped us a lot to convert our plots in NVIDIA theme colors for aesthetics!*

4.3 Fail

In one instance, an AI assistant produced a quantum circuit construction that appeared syntactically correct but violated a key physical constraint implied by the problem Hamiltonian. This failure was detected through unit tests that checked known symmetry properties. The issue was resolved by reverting to a manually derived formulation and updating prompts to explicitly state the required physical invariants, preventing recurrence of the error.

Another failure was quite amusing. We were using ChatGPT to generate images for presentation, and it kept putting R_X circuit boxes everywhere in the circuit image, even though we kept insisting not to do so. We eventually gave up...

5 Context Dump

The following excerpts illustrate the type of structured prompting used throughout the project.

1. `Find reported literature that obtains the ground state energy for the LABS problem for different values of N. Provide citations and clearly indicate whether results are exact or heuristic.`

2. ```
   Read the attached paper and identify the circuit validation
   mechanisms implemented by the authors. Focus on explicit checks,
   benchmarks, and assumptions used to verify correctness.
   ```

3. ```
   Help convert the plots into an NVIDIA-style color theme using
   Neon Green, White, and Charcoal Black. Preserve readability and
   scientific clarity.
   ```

4. ```
   Summarize peer-reviewed work on variational quantum algorithms
   for combinatorial optimization. Provide citations and clearly
   distinguish empirical results from theoretical guarantees.
   ```

5. ```
   - Prefer explicit parameterization over implicit defaults.
   - No black-box optimizers without documented assumptions.
   - All AI-generated code must be unit tested against
     analytically solvable cases.
   ```

These artifacts show our strategy of deliberate context setting and constraint-driven prompting that ensures AI tools were operated within well-defined technical and scientific boundaries.

6 Conclusion

Our project used a disciplined approach to AI-assisted development in a research-intensive setting. By constraining AI tools to well-defined roles, enforcing rigorous verification, and maintaining human control over all scientific decisions, the workflow achieved significant productivity gains without sacrificing interpretability, correctness, or reproducibility.