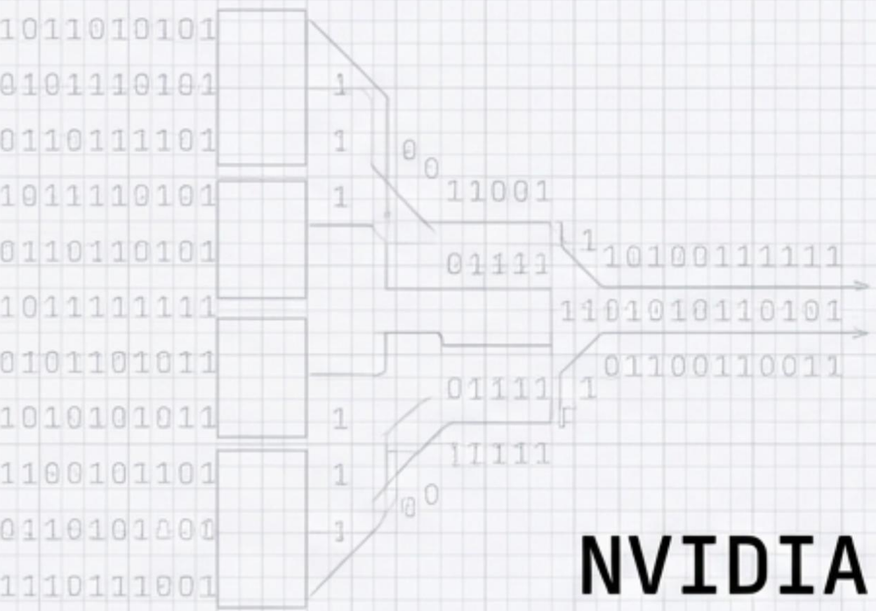
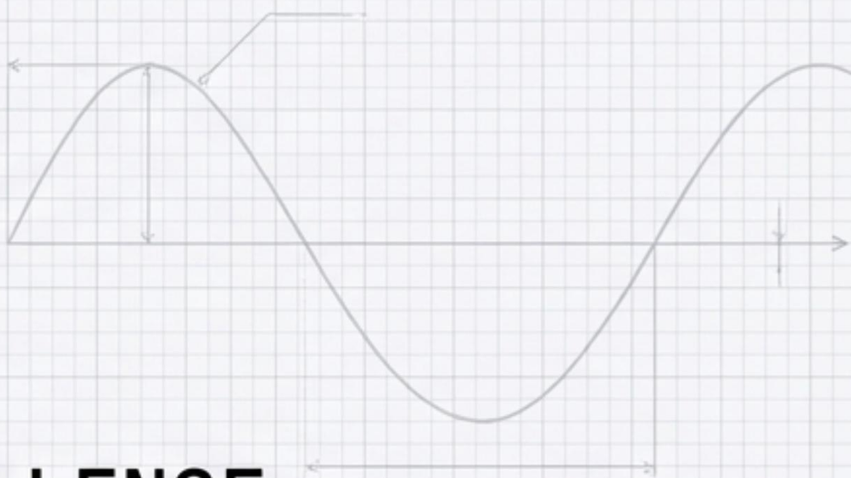


SOLO

NVIDIA iQuHack 2026 CHALLENGE



Solving the LABS Problem via Quantum-Enhanced Memetic Tabu Search

Developer: Musfar Muhamed Kozhikkal

Repository: github.com/musfarmuhamed/NVIDIA-SOLO

Tracks: Phase 1 (Quantum Logic) & Phase 2 (GPU Acceleration)

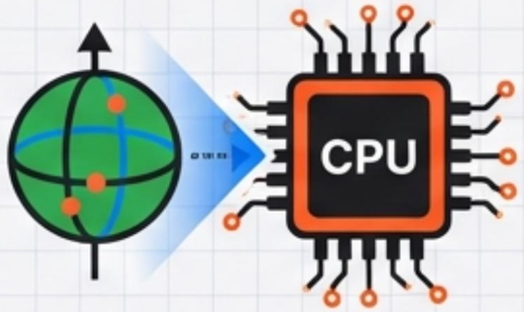
EXECUTIVE SUMMARY

ARCHITECTURE

Hybrid Pipeline Implemented

Counter-Diabatic Quantum Optimization (CD-QO) seeds the population.

Classical Memetic Tabu Search (MTS) refines it.



VERIFICATION

Codebase Verified

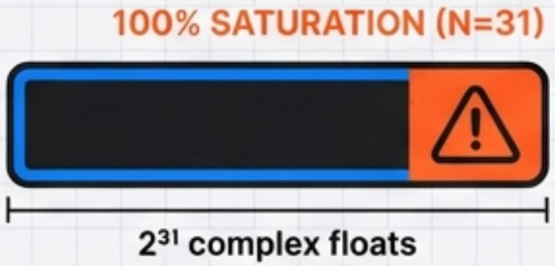
`pytest` suite confirms interaction generation (G_2, G_4) and energy calculations ($E(s)$).

100% Deterministic.

THE BOUNDARY

Hardware Limit Identified

Full state-vector simulation on NVIDIA A100 saturates memory at exactly N=31 (2^{31} complex floats).

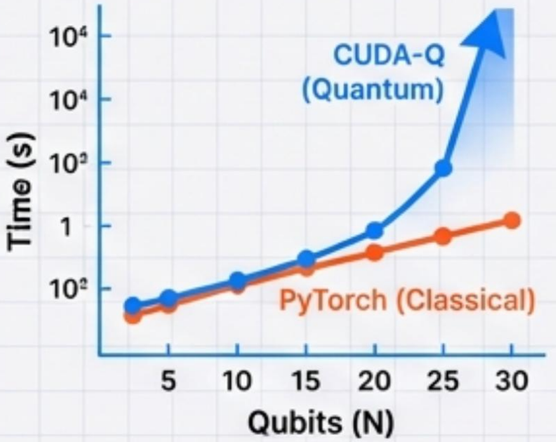


OPTIMIZATION

Performance Benchmarked

PyTorch (Classical) achieves linear scaling.

CUDA-Q (Quantum) hits exponential wall at N=25+.



The Challenge: Low Autocorrelation Binary Sequences (LABS)

A needle in an exponentially growing haystack.

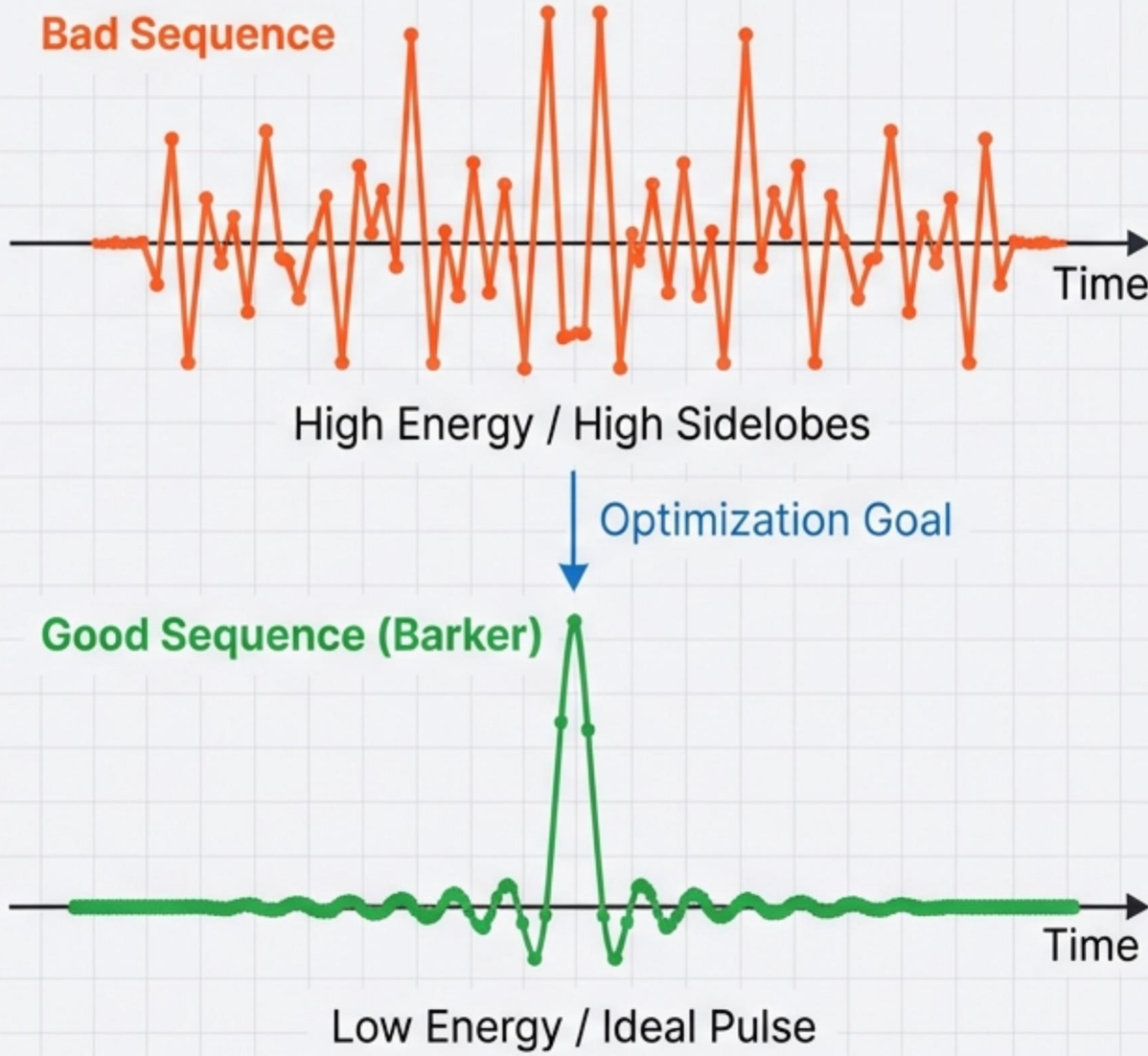
Context: in JetBrains Mono Medium.
Critical optimization for high-performance radar pulse compression and telecommunications.

The Math:

Minimize Energy: $E(s) = \sum_{k=1}^{N-1} C_k^2$

Where Autocorrelation $C_k = \sum_{i=1}^{N-k} s_i s_{i+k}$

Complexity:
Search space is 2^N. For N=60, this is intractable for brute force.



SOLUTION ARCHITECTURE: QUANTUM SEEDING + CLASSICAL REFINEMENT

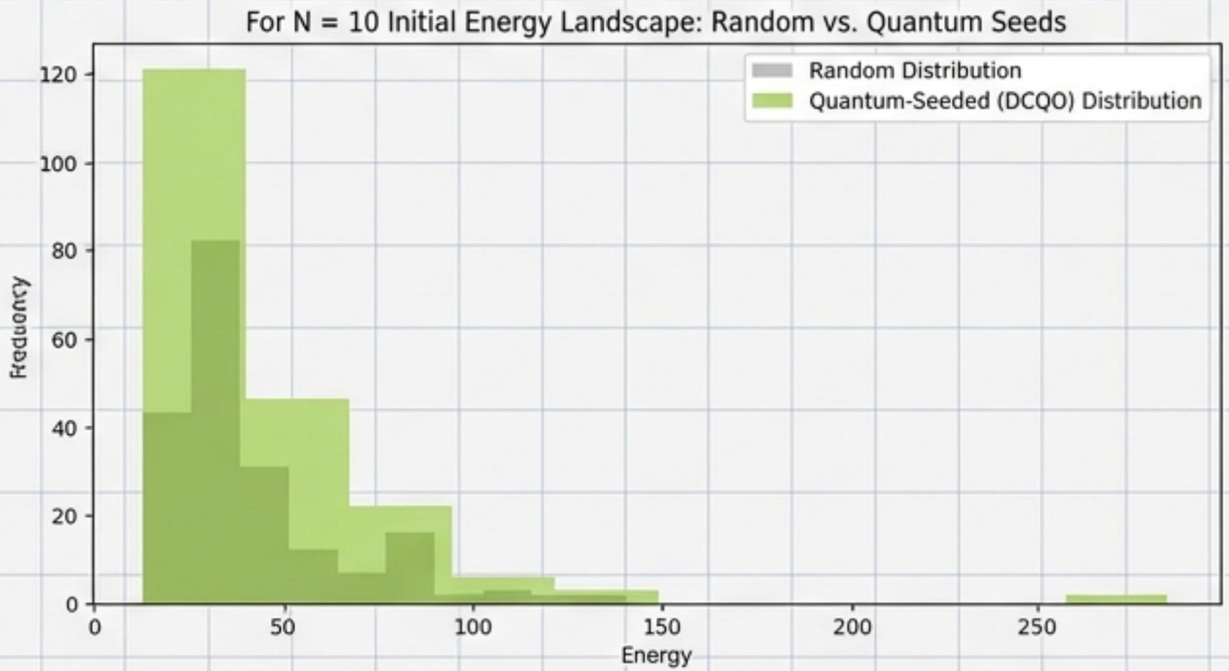
QUANTUM SEEDING (CUDA-Q)

- Counter-Diabatic Optimization
- Suppresses ground state transitions
- Output: Better-than-random distribution

Seeding the Population

CLASSICAL OPTIMIZATION (PYTORCH)

- Memetic Tabu Search (MTS)
- Input: Quantum Seeds
- Process: Greedy local search & refinement



Phase 1 Implementation: Constructing the Hamiltonian

Complex Multi-Qubit Interactions required for LABS.

- 1. Two-body terms (G_2): Mapped to `two_qubit_rotation_block` (R_YZ, R_ZY).
- 2. Four-body terms (G_4): Mapped to `four_qubit_rotation_block`. Requires 10 entangling **Rzz** gates.
- **Trotterization**: Dynamic Theta Calculation $\theta(t) \approx -\Gamma_1(t) / \Gamma_2(t)$.

CUDA-Q Kernel

```
@cudaq.kernel
def four_qubit_rotation_block(theta:float, q0, q1, q2, q3):
    # Layer 1: Basis changes & Rzz Entanglement
    rx(-np.pi/2, q0); ry(np.pi/2, q1); ry(-np.pi/2, q2)
    rzz(-np.pi/2, q0, q1)
    rzz(-np.pi/2, q2, q3)
    # ... sequence continues for 10 entangling gates
    rzz(theta, q1, q2)
```

Verification Strategy

Speed is nothing without accuracy.

Tooling

- Comprehensive `pytest` suite developed for Classical and Quantum modules.

Classical Verification (PyTorch)

✅ Verified `calculate_energy_batch` against known Barker sequences (N=3, E=1).

✅ Verified `tabu_search` logic ensures energy monotonicity (solutions never get worse).

Quantum Verification (CUDA-Q)

✅ Validated topology overlaps (I_{22}, I_{44}) for correct Gamma term calculation.

✅ Verified interaction generation (G_2, G_4) bounds for arbitrary N.

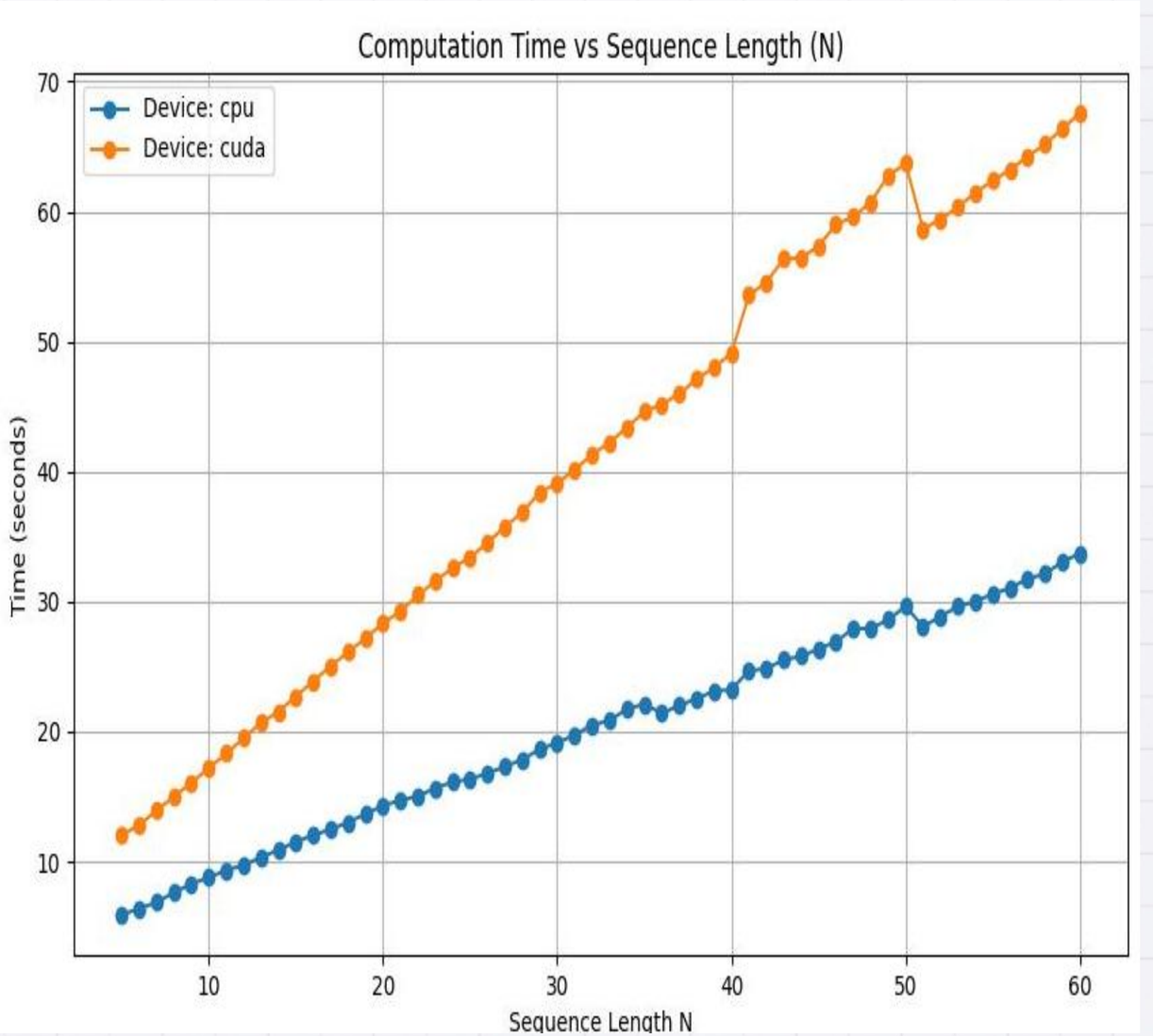
```
$ pytest test_labs_mts.py test_labs_quantum.py
test_labs_mts.py::test_energy_calculation_barker_n3  PASSED [ 50%]
test_labs_quantum.py::test_interaction_generation_n5 PASSED [100%]

12 passed in 0.42s
```

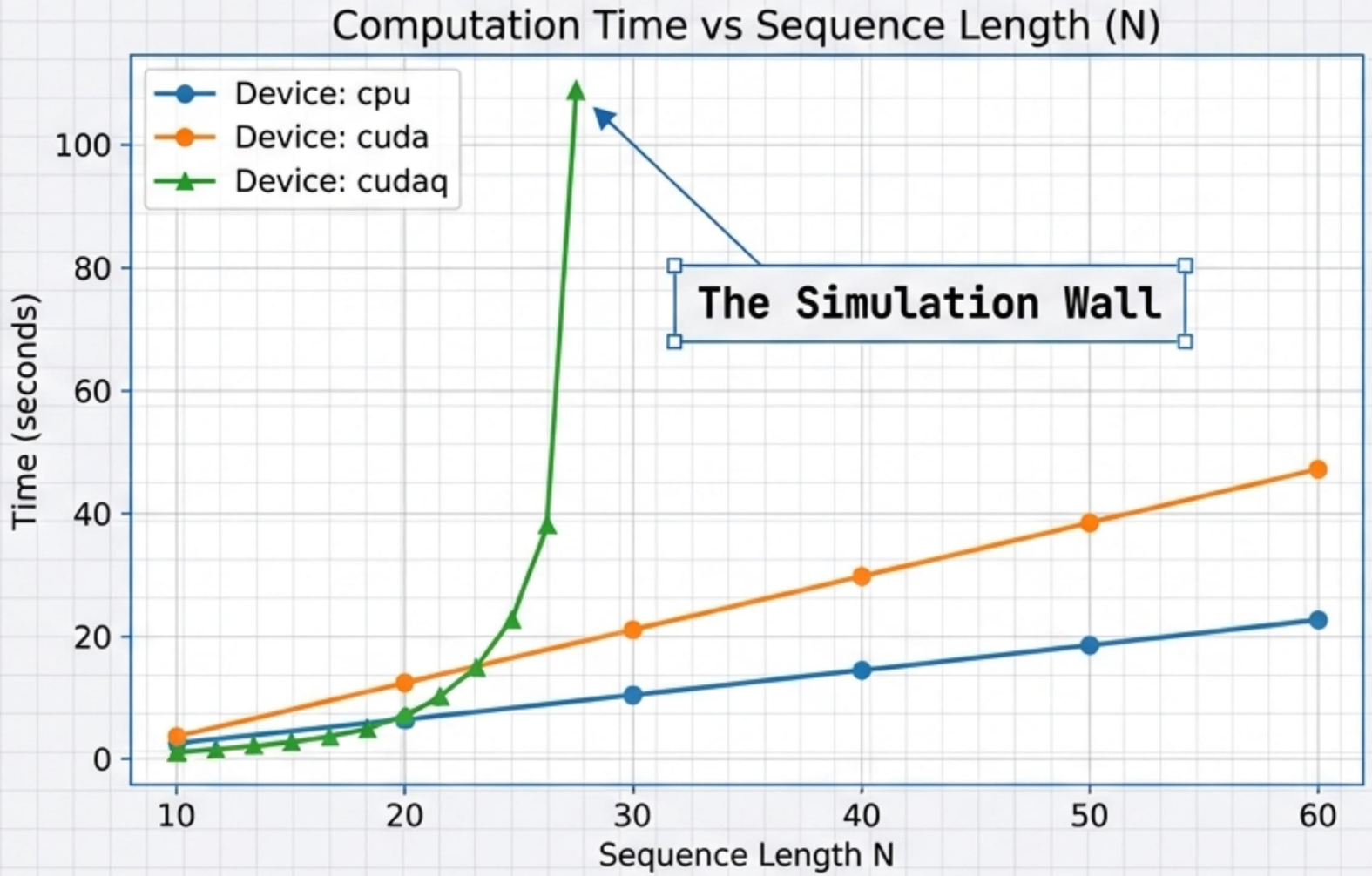
Classical Performance: CPU vs. GPU (A100)

Analysis

- **Method:** Vectorized PyTorch implementation (MTS_Solver_Torch).
- **Observation:** CPU (Blue) exhibits efficient linear scaling (~3$s at N=60).
- **Insight:** For population size=30, the A100 (Orange) overhead outweighs parallelization benefits. The GPU solution is architected for massive scalability (pop > 10,000) where CPU vectorization bottlenecks.



Quantum Simulation on GPU: The Exponential Reality



While Classical MTS scales linearly (seconds), Quantum Simulation scales exponentially.

At N=30, simulation takes ~507 seconds.

This validates the need for real QPU hardware.

Hitting the Hardware Limit: The N=31 Wall

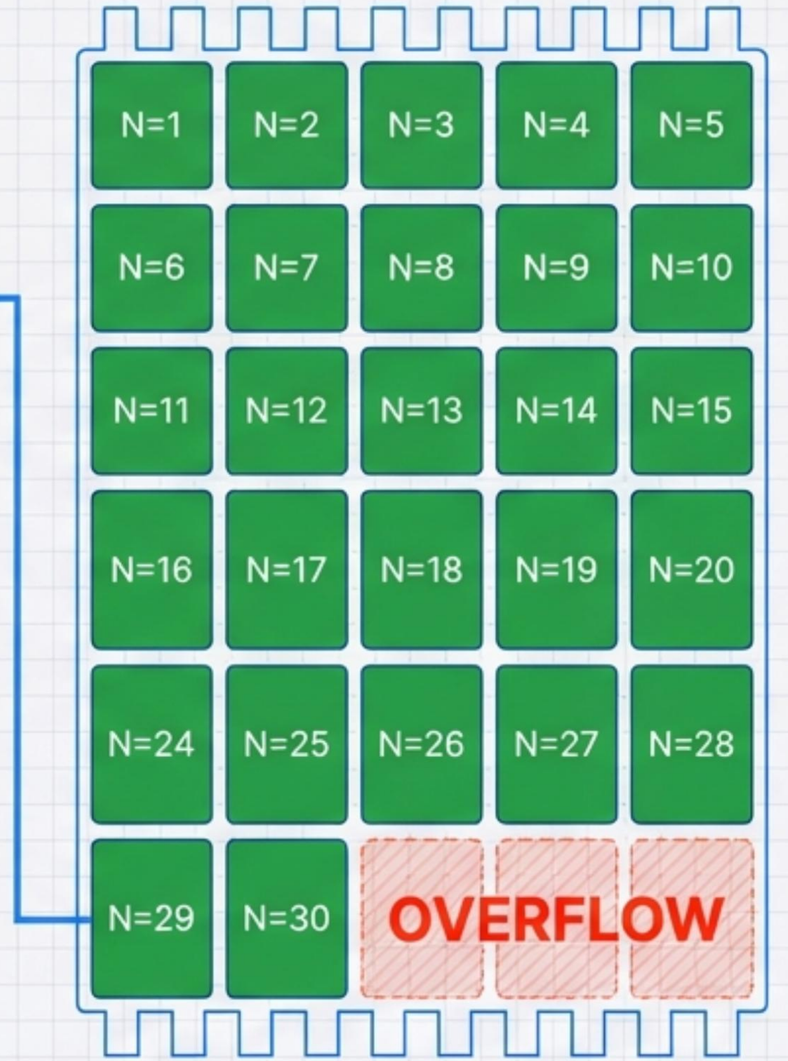
Status Report

✅ N=30: Simulation Successful (~507s). Memory usage ~16GB.

❌ N=31: FAILED.

Error Log

```
Error processing N=31:
requested size is too big
```



Root Cause Analysis

- State vector size doubles with every qubit.
- 2^{31} complex floats exceeds single-device addressable space for tensor operations.
- Conclusion: NVIDIA A100 capabilities successfully saturated.

Solution Quality: Merit Factor Analysis

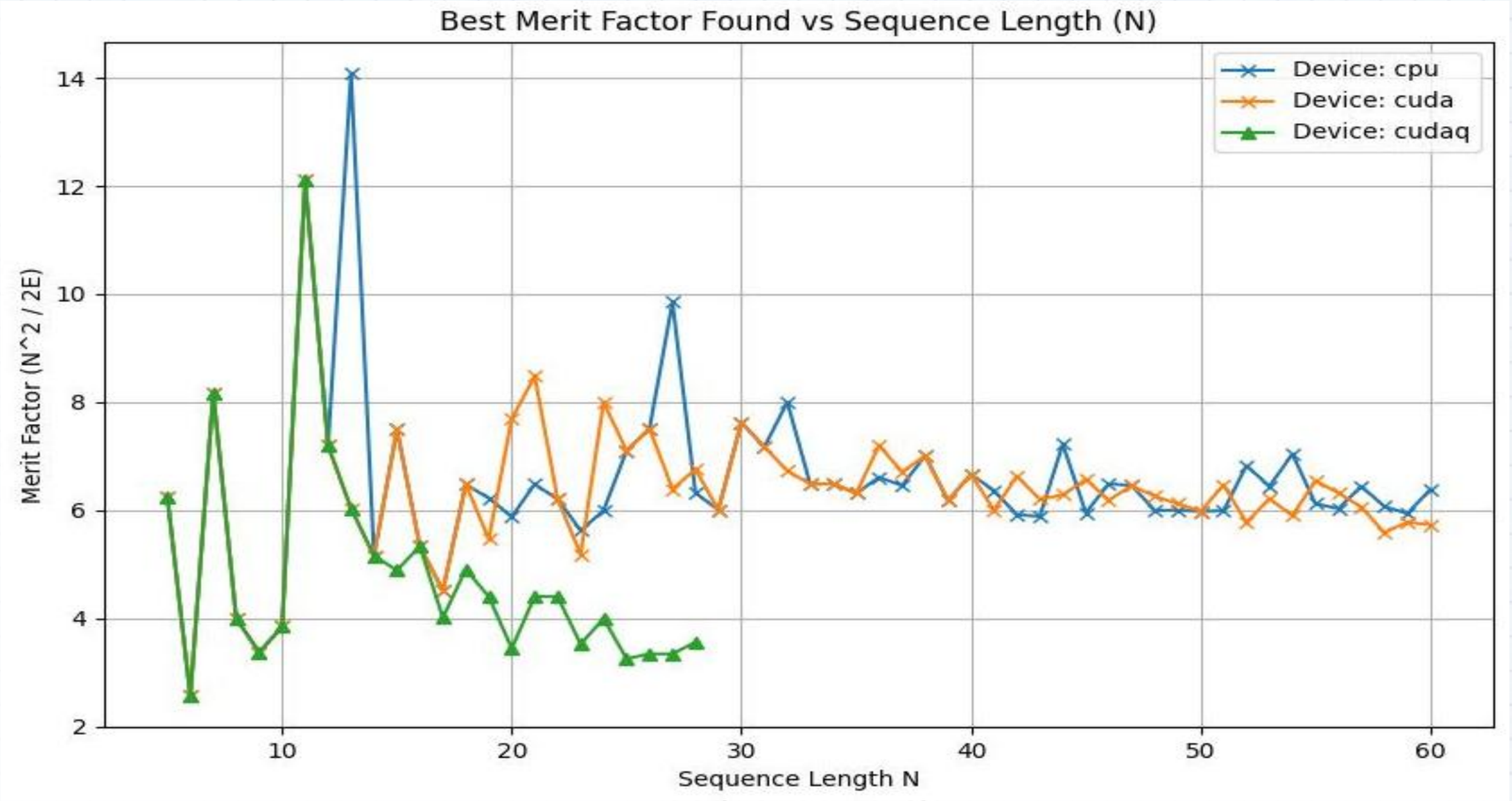
Finding the right answer matters more than speed.

METRIC

Merit Factor F = `N^2 / (2E)` (Higher is Better)

OBSERVATIONS

- Hybrid/Classical approach consistently finds high-quality solutions, overcoming local minima.
- Peaks observed at N=11 (MF ~12) and N=13.



The Agentic Workflow

"Letting agents build the code while I built the architecture."

JetBrains Mono Regular
AI Agents broke down LABS Hamiltonian into G_2 and G_4 interaction lists.

JetBrains Mono
Translated theoretical Trotterization formulas (Eq. B3) into `cudaq` kernels.

JetBrains Mono
Generated `pytest` scaffolding to test complex math against knowns.

Conclusion & Future Outlook

Verified

Constructed a fully tested Hybrid Quantum-Classical pipeline from scratch.

Benchmarked

Demonstrated the linear scalability of Classical GPU methods vs the exponential wall of Quantum Simulation.

Bounded

Defined the exact simulation limit (N=31) on A100 hardware.

Next Step: Physical Hardware

The code is deployment-ready. Moving from Simulation (CUDA-Q) to Real QPU is the only way to bypass the N=31 memory wall and unlock quantum advantage.

Resources & Codebase

Repository: https://github.com/musfarmuhamed/NVIDIA-SOLO

Key files:

- `SOLO_01_quantum_enhanced_optimization_LABS.md` (Phase 1 Report)
- `LABS_Quantum_GPU.py` (CUDA-Q Implementation)
- `LABS_Classical_MTS_GPU.py` (PyTorch Implementation)
- `test_labs_mts.py` & `test_labs_quantum.py` (Verification Suite)

Team SOLO: Musfar Muhamed Kozhikkal

NVIDIA iQuHack 2026