C-QuEE: Cole's Quantum Efficiency Equation

A Metric for Evaluating Real-World Quantum Software Efficiency

Author: Gabriel E.K. Cole

This whitepaper introduces C-QuEE, a tunable and modular metric designed to evaluate the real-world performance of quantum software across four key dimensions: fidelity (noise), resource usage (memory), time efficiency, and input scalability. Unlike hardware-bound metrics such as quantum volume, C-QuEE is software-centric and focuses on how algorithms perform when deployed on real or simulated quantum hardware.

The C-QuEE Metric

C-QuEE =

$$1 - (\alpha\left(\frac{1}{n}\sum_{i=1}^{n} Noise\mathrm{Penalty}_i\right) + \beta\left(\frac{1}{n}\sum_{i=1}^{n} MemoryPenalty_i\right) + \gamma\left(\frac{1}{n}\sum_{i=1}^{n} TimePenalty_i\right) + \delta\left(\frac{1}{n}\sum_{i=1}^{n} ScalibilityPenalty_i\right))$$

Where:

- n = number of test cases

- α, β, γ, δ = tunable weights depending on the use case (do not need to sum to 1)

Penalty Component Definitions

1. NoisePenalty

$NoisePenalty_i$ = 1 - $Fidelity_i$

Fidelity can be calculated using cross-entropy benchmarking, total variation distance, or probability match with the ideal distribution.

2. MemoryPenalty

$MemoryPenalty_i$ = $QubitsUsed_i$ / ExpectedAvailableQubits

ExpectedAvailableQubits is defined as the realistic qubit capacity of the intended deployment hardware (e.g., 27 for IBM Falcon, 20 for IonQ Harmony).

3. TimePenalty

$$TimePenalty_{\,i} = \begin{cases} \dfrac{QuantumRuntime_i}{ClassicalTime_i} \\ \dfrac{QuantumRuntime_i}{QuantumRuntime_{best}} \end{cases}$$

- Quantum Runtime_i = time taken by the quantum algorithm for test case *i*
- Classical Runtime_i = time taken by a classical algorithm solving the same test case
- Benchmark Quantum Best = best known quantum runtime for that class of problem (used when no classical baseline is available)

4. ScalabilityPenalty

$$ScalabilityPenalty_{\,i} = 1 - \min\left(1, \frac{\log_2(\text{InputSize})}{\log_2(\text{QubitsUsed})}\right.$$

This captures how efficiently the algorithm uses qubits relative to the size of the input problem.

Worked Example: C-QuEE Score Calculation

The following is a worked example showing how to compute the C-QuEE score based on input values for a quantum algorithm test case.

Input Data

- Fidelity: 0.90
- Qubits Used: 18
- Expected Available Qubits: 27
- Quantum Runtime: 1.5
- Classical Runtime: 2.0
- Input Size: 512

Penalty Calculations

- NoisePenalty = 1 − Fidelity = 1 − 0.90 = 0.10
- MemoryPenalty = QubitsUsed ÷ ExpectedQubits = 18 ÷ 27 = 0.6667
- TimePenalty = QuantumRuntime ÷ ClassicalRuntime = 1.5 ÷ 2.0 = 0.75
- ScalabilityPenalty = 1 − min(1, $\log_2$ (512) ÷ $\log_2$ (18)) = 1 − min(1, 9 ÷ 4.17) = 0

Final C-QuEE Score = 0.6208

Score Interpretation

Use the table below to interpret the meaning of the computed C-QuEE score:

Score (0-1)	Meaning
0.9-1.0	Excellent (high fidelity, efficient resource use, scalable)
0.7-0.89	Good
0.5-0.69	Moderate
0.3-0.49	Poor
0-0.29	Inefficient or poorly optimized quantum algorithm

In this case, the quantum algorithm scored in the "Moderate" range, indicating potential for improvement in memory or runtime efficiency.

Author & Research Context

This metric was developed by **Gabriel E.K. Cole**, an 18 year-old undergraduate researcher at Oregon State University pursuing a double major in **Computer Science (Artificial Intelligence)** and **Biological Data Science (Computational Biology)**, with a minor in Music Performance.

The work was conducted under **NeurQL**, a research initiative founded by the author to explore the intersection of quantum computing, artificial intelligence, and genomics.

The C-QuEE framework is currently being implemented as part of a software tool to automate benchmarking and visualization of quantum software performance. A companion metric, tentatively named **C-QuEE-C** (for Classical Comparison), is under development to evaluate quantum software against classical algorithms across runtime, scalability, and resource efficiency.

This system aims to enable practical benchmarking across real-world applications in **AI, genomics, and optimization**.

Researchers, developers, or institutions interested in contributing, testing, or integrating C-QuEE into their pipelines are encouraged to reach out:

gabrielcolehi@gmail.com