Week 1: Simulating quantum advantage with trapped ions

Introduction

In 2019, the Martinis research group at Google ran an experiment on their Sycamore quantum computer, and they claimed it would take a classical supercomputer 10,000 years to simulate the same experiment [1]. Thus, they concluded that this was the world's first example of quantum computational advantage: a quantum computer has drastically outperformed a classical one at a specific task, and the disparity will grow exponentially as the task is made more difficult. Although different approaches have since been suggested to drastically improve on the classical bound of 10,000 years [2, 3], this remains a landmark feat of engineering on the road to general purpose quantum computation.

The task that was chosen for Google's demonstration is sampling from random circuits composed of 1- and 2-qubit gates. While this sampling task was initially believed to provide no "real-world" applications, potential use cases are beginning to appear [4]. It's important to keep in mind that the current state-of-the-art gate-based quantum computers are limited by errors in the individual gates, which compound to produce unreliable results. Indeed, the current generation of quantum computers is often referred to as "noisy intermediate-scale quantum" (NISQ) to emphasize their lack of error correction. Using quantum computers for truly useful tasks (such as Shor's algorithm to factor integers out of reach of classical factoring techniques) is estimated to still be a long way into the future.

However, in a clever twist, Ref. 1 actually takes advantage of the noisy nature of their quantum computer. Their main result is the experimental determination of the linear cross-entropy benchmarking (XEB) fidelity [5, 6]:

$$\mathcal{F}_{\text{XEB}} = 2^N \langle P \rangle - 1 = \frac{2^N}{S} \sum_{i=1}^{S} P(x_i) - 1, \tag{1}$$

where N is the number of qubits, $P(x)$ is the exact probability of observing the bit-string x, and the average in angle brackets is performed over S samples. Ideally, $\mathcal{F}_{\text{XEB}}$ should be 1, as would be the case if the samples $\{x_i\}_{i=1}^{S}$ were drawn from the true distribution P; for example, if their quantum computer had no errors. In practice, $\mathcal{F}_{\text{XEB}} \ll 1$, due to imperfections in the realizations of the circuits. Crucially, the authors were able to extrapolate $\mathcal{F}_{\text{XEB}}$ based on their knowledge of the error rates of their quantum computer!

Superconducting qubits, such as those found in Sycamore are not the only possible foundation for a gate-based quantum computer. Numerous other experimental implementations exist, but this week we will focus on ion trap quantum computers. We can model 1-qubit gates as

$$R(\theta, \varphi) = \begin{pmatrix} \cos\frac{\theta}{2} & -ie^{-i\varphi}\sin\frac{\theta}{2} \\ -ie^{i\varphi}\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}, \tag{2}$$

and 2-qubit gates as

$$M(\Theta) = \begin{pmatrix} \cos\Theta & 0 & 0 & -i\sin\Theta \\ 0 & \cos\Theta & -i\sin\Theta & 0 \\ 0 & -i\sin\Theta & \cos\Theta & 0 \\ -i\sin\Theta & 0 & 0 & \cos\Theta \end{pmatrix}, \tag{3}$$

which can be combined in various ways to create circuits for an ion trap quantum computer [7, 8].

The tasks this week will involve applying these gates to a quantum state on a classical computer, with additional gates to simulate errors. You will see how chaos can arise in individual measurements of random circuits, but also that it can be tamed by averaging over many measurements.

Supervised learning

This section will guide you through several aspects of simulating an ion trap quantum computer using your classical computer. The starting points provided for these tasks make use of Julia (`https://julialang.org/`) and PastaQ (`https://github.com/GTorlai/PastaQ.jl/`), but teams are encouraged to use the environments and libraries they prefer.

Task 1

The output states of deep random circuits are chaotic, in the sense that slight perturbations in the circuits can lead to very different sampling probabilities. Ref. 1 makes an analogy between the visual patterns produced by a laser and samples produced by a quantum computer:

> Owing to quantum interference, the probability distribution of the bitstrings resembles a speckled intensity pattern produced by light interference in laser scatter, such that some bitstrings are much more likely to occur than others.

You will use this analogy to draw "speckle patterns" in order to visualize the states generated by perturbed random circuits.

The example script `run_random_circuit.jl` creates and runs a random circuit using the ion trap R and M gates, resulting in a random matrix product state (MPS), which represents a quantum state $|\psi\rangle$. Extend this script to **extract the coefficients $\langle x|\psi\rangle$ for each bit-string x from the MPS**.

The squared magnitude of each coefficient,

$$P(x) = |\langle x|\psi\rangle|^2, \tag{4}$$

gives the probability of sampling the bit-string x from $|\psi\rangle$. **Plot the probability of each x using a dot whose area is proportional to $P(x)$, like in Fig. 1, for several circuit widths and depths.**

Bonus: Examine the change in the bond dimension of the generated MPS as the random circuits are made deeper. Discuss the connection to entanglement entropy.

Figure 1: A "speckle pattern" displaying the probabilities of obtaining each of the 16 possible outcomes when sampling a 4-qubit state.

Task 2

The outputs of sufficiently deep random circuits are very sensitive to slight perturbations in the circuit parameters, which can be seen by inserting errors. For example, quantum computers are susceptible to bit-flip errors, modelled using

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \tag{5}$$

Modify `run_random_circuit.jl` to **run one random circuit multiple times, but with a single bit-flip error at a random location in the each time**. The resulting speckle patterns should all be quite different; collect them into a collage.

Task 3

As the depth of the random circuits is increased, the distribution of the probabilities $p = P(x)$ for a fixed output bit-string x tends to the exponential distribution[1]:

$$p \sim 2^N e^{-2^N p}, \tag{6}$$

where N is the number of qubits. For example, Fig. 2 shows that for random circuits on 8 qubits, those of depth 512 result in a cumulative distribution quite close to the expected one,

$$1 - e^{-2^N p}, \tag{7}$$

while those of depth 1 do not. **Demonstrate the convergence of the empirical cumulative distribution function of p to the exact CDF of the exponential distribution with increasing circuit depth.**

Task 4

One source of error in experimental realizations of quantum circuits is the inability to perfectly implement gates. This could cause the observed distribution of p to differ from the expected distribution, but visual comparison of entire distributions can be tedious and inexact. A more quantitative approach to determine the difference between distributions is the linear XEB fidelity, $\mathcal{F}_{\text{XEB}}$. This is a single number that encodes the quality of the observed results.

Starting from a sufficiently deep random circuit, **perturb the angle of each 2-qubit gate by a fixed amount $\Delta\Theta$ to simulate a systematic error**. Plot $\mathcal{F}_{\text{XEB}}$ as a function of $\Delta\Theta$.

[1] In this context, it is sometimes referred to as the "Porter–Thomas distribution".

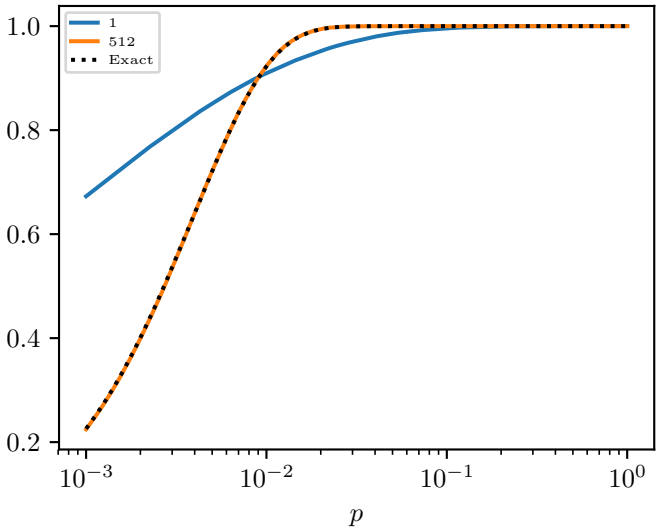

Figure 2: Empirical cumulative distribution functions of the probabilities p for 8-qubit circuits of depth 1 and 512.

Task 5

In the long term, some people expect quantum computing to become feasible on universal, gate-based quantum computers, which would undoubtedly benefit many branches of industry. Even with current NISQ hardware, methods exist to perform useful computations, such as the variational quantum eigensolver (VQE) [9]. However, sampling from random circuits does not lend itself to an obvious use case. **Describe a business that could be built around sampling-based quantum computing with random circuits in the NISQ era.**

Hint: Consider possible uses for certified random numbers.

Unsupervised learning

The following optional tasks are of a broader scope, and might require additional research on your part.

Task 6

Combine your speckle patterns from Task 2 into an animation that shows the changes in state probabilities as a single bit-flip error is moved around in a deep random circuit.

Task 7

Download some samples and circuits from the data repository for Ref. 1 at `https://datadryad.org/stash/dataset/doi:10.5061/dryad.k6t1rj8`. Use these to reproduce $\mathcal{F}_{\text{XEB}}$ values from the paper.

Task 8

Run your classical simulations of quantum circuits on a trapped ion quantum computer.

References

[1] Frank Arute et al. "Quantum Supremacy Using a Programmable Superconducting Processor". In: *Nature* 574.7779 (2019), pp. 505–510. DOI: 10.1038/s41586-019-1666-5.

[2] Edwin Pednault et al. "Leveraging Secondary Storage to Simulate Deep 54-Qubit Sycamore Circuits". In: *arXiv* (2019). arXiv: 1910.09534.

[3] Feng Pan and Pan Zhang. "Simulating the Sycamore Quantum Supremacy Circuits". In: *arXiv* (2021). arXiv: 2103.03074.

[4] Scott Aaronson. *Certified Randomness from Quantum Supremacy*. YouTube. 2018. URL: https://www.youtube.com/watch?v=hf7-Elx1Y4w.

[5] C. Neill et al. "A Blueprint for Demonstrating Quantum Supremacy with Superconducting Qubits". In: *Science* 360.6385 (2018), pp. 195–199. DOI: 10.1126/science.aao4309.

[6] Sergio Boixo et al. "Characterizing Quantum Supremacy in Near-Term Devices". In: *Nat. Phys.* 14.6 (2018), pp. 595–600. DOI: 10.1038/s41567-018-0124-x.

[7] Anders Sørensen and Klaus Mølmer. "Quantum Computation with Ions in Thermal Motion". In: *Phys. Rev. Lett.* 82.9 (1999), pp. 1971–1974. DOI: 10.1103/PhysRevLett.82.1971.

[8] Ivan Pogorelov et al. "A Compact Ion-Trap Quantum Computing Demonstrator". In: *arXiv* (2021). arXiv: 2101.11390.

[9] Alberto Peruzzo et al. "A Variational Eigenvalue Solver on a Photonic Quantum Processor". In: *Nat. Commun.* 5.1 (2014), p. 4213. DOI: 10.1038/ncomms5213.