Womanium Global Quantum Computing & Entrepreneurship Program

Challenge: End-to-end quantum simulation of space telescopes (feat. teleportation)

We begin with an analogy. Imagine that an alien gave you a superluminal spacecraft that could travel to Mars in five seconds. Great, you can get there before Elon Musk! However, you have a problem: The spacecraft only accepts dark matter as fuel. The alien did give you a recipe for converting regular jet fuel into dark matter fuel, but that process takes a thousand years to produce one gallon. You need 5 gallons per trip, so having an exponentially fast spacecraft simply doesn't help you. Disappointed, you cast away your spacefaring dreams and return to your day job as a therapist for cats with split personalities.

This is equivalent to the situation of many quantum algorithms, in particular those that solve differential equations. There are algorithms that provably can compute linear[1] and nonlinear[2] differential equations exponentially faster than classical computers, but they only work for quantum inputs. So if we want the quantum computer to use its exponential powers on a classical dataset, then we need to encode that dataset in quantum form! This is slow! If we have n such data points to load, then we must in general perform n operations to load the data. Thus, even if the simulation itself takes only $O(log(n))$ time, the data loading step ruins everything so that the final complexity is $O(n)$, which is no better than classical computers[3].

This challenge has the goal of highlighting this challenge, and to explore how it can be overcome. This is done by solving a real problem on a quantum computer, albeit scaled down: Predicting the path of the James Webb space telescope by solving a two-dimensional differential equation on a quantum computer. We first focus on the "middle" part of the problem, namely to perform the integration. When that is done, we focus on the problem of loading the input data to this algorithm, and how that affects the

[1] A. W. Harrow, A. Hassidim, S. Lloyd, Quantum algorithm for linear systems of equations. Phys. Rev. Lett. 103, 150502 (2009).
[2] J. P. Liu, H. Ø. Kolden, H. K. Krovi, N. F. Loureiro, K. Trivisa, A. M. Childs, Efficient quantum algorithm for dissipative nonlinear differential equations. Proceedings of the National Academy of Sciences 35, 118 (2021).
[3] It's more accurate to write $$\Omega(\text{poly}(n))$$ and $$O(\text{polylog}(n))$$, but this does not affect the conclusion.

total runtime of the algorithm. Finally, the challenge opens up to exploration on how this challenge can be overcome in practice.

Output

This challenge will produce two outputs:

1. A Jupyter notebook with the implementation of a working quantum differential equation solver. You are free to copy-paste code from the qiskit examples where that is appropriate, but you are encouraged to understand (and demonstrate understanding of) the algorithms being implemented.
2. A slide presentation in PDF format which does two things:
    a. For Task 1-3, the presentation should clearly answer the specific questions to highlight the main takeaways from the challenge.
    b. For Task 4, the presentation should demonstrate creativity and ambition for applying the algorithms to real-life problems.

Evaluation

- Jupyter notebooks are evaluated on whether they successfully work or not, and demonstrate the concepts asked. Each notebook task weighs 20% towards the total evaluation score.
- The presented slides are evaluated on their clarity and demonstration of understanding. Each subtask in Task 1, Task 2 and Task 3 are weighted 3%, and the final task is granted the most weight of 25% in total. Note that Task 4 can and should be completed even if the preceding tasks are not done.

Task 1: The fast algorithm

You are predicting the position of the James Webb space telescope, which we assume to follow a circular orbit around the poles of Earth with a period of $T = 6$ months. Pretending we don't know the analytical solution to this problem, we figure out that its position (x, y) at time t is given by the system of differential equations

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} 0 & -2\pi/T \\ 2\pi/T & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}.$$

Using the backward Euler method[4] of integration, we can convert this to a linear system of equations

$$\begin{pmatrix} 1 & 2\pi\Delta t/T \\ -2\pi\Delta t/T & 1 \end{pmatrix} \begin{pmatrix} x(t+\Delta t) \\ y(t+\Delta t) \end{pmatrix} = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}$$

[4] https://en.wikipedia.org/wiki/Backward_Euler_method

which can then be solved repeatedly to advance the system by the time step Δt each time.

1. How can you encode the position (x, y) of the telescope on a qubit? [Output: Presentation slide, 3% score]
2. Using the resources in https://qiskit.org/textbook/ch-applications/hhl_tutorial.html, demonstrate how a quantum circuit can simulate the orbit of the James Webb telescope using the implicit Euler method of integration. [Output: Jupyter notebook, 20% score]

Task 2: Input encoding

To understand this task, you should read up on big-O notation for computational complexity. Khan Academy has a great summary on https://www.khanacademy.org/computing/computer-science/algorithms/asymptotic-notation/a/asymptotic-notation. Also note that we will trade accuracy for simplicity by avoiding technical details such as whether to use big-O, big-Theta or big-Omega.

Now that we have an exponentially fast quantum algorithm, we will focus on how it scales. Imagine that we do not only want to track one satellite, but n of them.

1. How can you encode the state of n satellites on qubits, i.e. $(x_1, y_1, x_2, y_2, \ldots, x_n, y_n)$? How many qubits do you need? [Output: Presentation slide, 3% score]

Now that we have an encoding scheme for the input state, we focus on the complexity of the algorithm.

2. Make a quantum circuit that prepares the input state. [Output: Jupyter Notebook, 20% score]

With the computational complexity of state preparation determined, we consider the complexity of the linear systems algorithm. In fact, the computation complexity of solving linear systems is $O(n)$ on classical computers and $O(log(n))$ on quantum computers.

3. If a classical computer and a quantum computer both spent 1 seconds on solving an 10×10 linear system, how long would it approximately take them each to solve a $10^{12} \times 10^{12}$ linear system? [Output: Presentation slide, 3% score]
4. Consider your state preparation circuit and how many steps it requires. Can you give a lower bound on its computational complexity in terms of n? What does this mean for the future prospects of quantum differential equation solvers? [Output: Presentation slide, 3% score]

Task 3: Teleportation

In order to avoid the input preparation problem, you have created a quantum-powered satellite equipped with a quantum sensor. Every 24 hours, the satellite scans every satellite orbiting earth and prepares a quantum state that efficiently encodes their positions. However, your quantum computer is located at MIT, so somehow the satellite must be able to transmit its quantum measurements to a quantum computer on good old Earth.

1. Based on the algorithm described in https://qiskit.org/textbook/ch-algorithms/teleportation.html, describe a practical step-by-step guide for how to teleport the measurement quantum state down to Earth. [Output: Presentation slide, 3% score]
2. Combine the quantum teleportation circuit with the linear differential equation solver you created in Task 1. [Output: Jupyter notebook, 20% score]

Task 4: Practical application to new problems

Congratulations, you have now created an (almost) end-to-end quantum algorithm! Notably we have not considered the problem of reading the output, i.e. how to actually get the answer we want from the quantum computer. After tall, the result of the quantum algorithms we consider are quantum states, and those are notoriously hard to read out completely. Due to time constraints we won't dig deeper into that, and we will instead focus on practical applications. Choose one application to focus on in which you would want to apply a quantum differential equation solver, and present/discuss the following questions. [Output: Presentation slides, 25% score]

1. If you had a perfect quantum computer, which simulation problem would you want it to solve?
2. Which algorithms have promise for solving such problems within the next 20 years? Optimism is encouraged.
3. Which data and in what format would that algorithm need fed as input to perform the simulation?
4. How could you collect that data as quantum states or qubits instead of classical bits?
5. Which quantum sensors or collection method would you need to gather the quantum data?
6. How would you transfer the quantum data from the quantum sensor to the quantum computer?
7. What would the impact of your described end-to-end application be, and how could it solve real-life problems?

Contact information

Mentor for this challenge: Herman Kolden (Discord: @hermankolden)