Quantum Programming (4ETI) **Spring 2020**

Final Project Guidelines

Instructor: Dr. Fedrici Due date: 15^{th} April, 2020

The course project is the capstone of the class. It is preferred that you do this project in group of three, however smaller groups will be accepted as well. There are two ways to approach the project:

- **(Programming Project)** You and your group will choose a topic in quantum computing and implement an interesting program along with a short report (3-5 pages) that is relevant to that topic.
- **(Theory Project)** You and your group will choose a topic, research a few relevant papers, and write a research report (7-10 pages) that covers the theory results for your project.

In both approaches, original research is encouraged, but not necessary. Suggested project topics are listed below. You and your group can either choose from these topics or suggest an alternative of your own design. This is an opportunity to combine your interest and your own background (in machine learning, optimization, distributed systems, physics, *etc*) with what you have learned in this course.

Importantly, you will need to pick a project that is doable in the given time frame (from mid-April to May 10th). You and your group will have to submit a project proposal by April 15th, and I will give feedback on whether the scope is appropriate for the course.

Here are some suggestions on how you can pick topics for your project:

1. Choose a suggested project topic from the list at the end of this document.
2. Take something you're already doing a research in, and explore if there is a quantum version of it.
3. Take a topic covered in the class, and investigate it more deeply.
4. Take a topic in quantum computing you have always wanted to learn about (but was not covered in class).

1. Deliverables

Here is what is expected of you (the last one is optional and the middle ones depend on whether you are doing a programming or a theory project):

1. **(Project Proposal)** Your group will submit a proposal by email that includes the names of your group members. This proposal is due on **April 15th**. This should be at most one page describing your proposed project, along with some relevant papers/resources. A strong initial proposal will include the topic area, the general structure of the intended implementation, as well as the key results that will be intended to use in the research report. I will give a feedback and suggest additional papers or implementation suggestions. Feel free to schedule appointment to discuss your project in person.
2. **(Project Progress Update)** Less than one page written update on your project progress from your group. This is due on **April 28th**. It should describe the progress you have made against your original proposal, including any changes of direction and results so far.
3. **(Programming Project Only: Code Repository)** No later than by **May 10th** your group should upload your final project code to either (i) a publicly viewable repository (github, gitlab, bitbucket, *etc*) or (ii) a private repository which access credentials are provided in your report (see

below). Your code should include full instructions on how to install your project as well as instructions on how to get started and use some of its key features. If you have alternative packaging methods that are needed, please include these in your project proposal.

4. **(Programming Project Only: Short Written Report)** Your group must send by email a final report no later that **May 10th**. It should be 3-5 pages long. It can be longer, but I will read submissions longer than 5 pages only at my own discretion. Please use reasonable margins and font sizes. This report will describe what is interesting about your project and the data/examples you have taken with your implementation to show that it works. It should also include some remarks on potential future extensions of your work. Your report should describe how to find and access the repository with your code.

5. **(Theory Project Only: Written Report)** Your group must send by email a final report no later that **May 10th**. It should be 7-10 pages long. It can be longer, but I will read submissions longer than 10 pages only at my own discretion. Please use reasonable margins and font sizes. This report will describe the research background for your project and what extensions / constructions / proofs you have made in your project. It should also include some remarks on potential future extensions of your work.

6. **(Oral Presentation)** On **May 15h** you will give a 15 min group presentation of your work in front of your colleagues and a selected jury. 5 min Q/A session will take place following your presentation. The exact time slot for each group will be communicated later.

7. **(Online Showcase)** I plan to showcase the students' projects on the course homepage for other students and researchers to see what cool things you have done. Participation in the showcase is optional, and has no effect on your grade. Benefits: your project gets some publicity, and also the quantum computing community benefits from what you've built! You have the option of making either/both/none of the report and code accessible openly.

2. Type of Projects

Here are some examples of types of projects that you can do:

1. **(Compare implementations across frameworks)** In these projects you can take an example algorithm or compilation trick and implement it in few different frameworks. In the course we have mostly used Qiskit (IBM), but there's also pyQuil (Rigetti Computing), Q# (Microsoft), Cirq (Google), and others. For a list of frameworks to potentially work with check out: `https://qosf.org/project_list/`

2. **(Implement a small instance on a real QPU)** Here you will take an algorithm that runs on a simulator and convert it into a small instance that runs on a QPU. This can be tough! Today's QPUs are small and very noisy. However, this type of project is likely to be very rewarding and may even end up as a potential publication after further development and work. For this course you will have access to IBMQ platform.

3. **(Benchmark implementations across emulators)** Some quantum algorithms (Shor's, Grover's, HHL) are too big to be run or even simulated today. In these cases you can write implementations that compile into programs, however these programs can't yet be run except in very small instances. Instead, you can compare different methods of implementing these algorithms and compare program outputs alongside with such metrics as qubits or gate depth.

4. **(Build developer tools)** In the following course project there are lots of developer tool suggestions. A project of this type would show the main feature(s) of the tool and how it helps with quantum programming in the near or far.

5. **(Theory project)** Extend an algorithm from the `https://quantumalgorithmzoo.org/`

3. Course Projects Topics

Here are some potential course project topics. This is by no means an exhaustive list of things you can do. If you end up being curious about an area and are looking for more references then please do ask me by email and I will be happy to give you some suggestions. If you are struggling with choosing a project then I recommend you to choose the Default project (given below). You can also schedule a Zoom meeting to discuss your interests and brainstorm a project.

1. **(Programming Project - Solving Linear Systems of Equations using HHL)** HHL (Harrow, Hassidim, and Lloyd) algorithm is one of the main fundamental algorithms expected to provide a speedup over their classical counterparts. Implement and benchmark a simulated version of the HHL algorithm for a small system of linear equations. What are the sub-routines of this algorithm ? What are the relevant parameters regarding the algorithm complexity ? What are current limitations in implementing this algorithm with current quantum devices ? If you select this project, I will provide you a tutorial to help you to start your project.

2. **(Programming Project - Variational Quantum Linear Solver)** Implement and benchmark a simulated version of the Variational Quantum Linear Solver, a new technique for solving linear systems of equations with hybrid quantum-classical computation.

3. **(Programming Project - Variational Quantum Factorizer)** Implement and benchmark a simulated version of the Variational Quantum Factorizer, a new technique for factoring numbers with hybrid quantum-classical computation.

4. **(Programming Project - Hybrid Quantum-Classical Neural Networks)** A parametrized quantum circuit can be seen as a quantum neural network. Following recent proposals in the field of quantum machine learning, implement and benchmark an hybrid quantum-classical neural network with PyTorch and Qiskit.

5. **(Programming Project - TensorFlow Quantum)** TensorFlow Quantum (TFQ) is a new quantum machine learning library developed by Google for rapid prototyping of hybrid quantum-classical machine learning models. Write a tutorial in the form of a Jupyter notebook that explain basic functionalities of TFQ. Implement and benchmark an hybrid-quantum classical neural network with it.

6. **(Programming Project - Measurement Error Mitigation)** The effect of noise is to give us outputs that are not quite correct. Investigate measurement error mitigation tools provided in Qiskit and implement and compare a few of these methods. How do they fight incoherent noise ? Are they accounting for coherent noise ?

7. **(Programming Project - Target Multiple Architectures)** The Qiskit 0.13 release features support for trapped ion devices via the introduction of XX transpilation between superconducting and trapped ion gate sets, and the qiskit-aqt-provider for communicating with the trapped ions quantum processor from Alpine Quantum Technologies - a company located in Austria. Implement and benchmark simple programs on superconducting and trapped ions quantum processors. Which approach performs best regarding gate fidelity, run time, *etc* ?

8. **(Programming Project - Single Qubit Quantum State Tomography Comparison)** There are several ways to reconstruct a quantum state from tomographic analysis. Implement and compare a few of these methods and use them to perform tomography on qubits on a real QPU.

9. **(Programming Project - Benchmarking Optimizers)** Pick a variational quantum problem, e.g. QAOA with MAXCUT and test several ways of optimizing the variational angles in the program. What classical methods of optimization work best ? Why ?

10. **(Programming Project - Constrained Optimization)** Choose a constrained optimization problem that has a reduction to MAX-CUT. Use this reduction (and any tricks you can find) to make a QAOA implementation of this algorithm that runs on the QASM simulator or could run on a QPU. Benchmark this implementation.

11. **(Programming Project - Bell's Theorem)** Bell's theorem proves that quantum physics is incompatible with local hidden variable theories. Study first the theorem and then test the so-called CHSH inequality on a real quantum hardware using pairs of entangled qubits. How do you interpret the obtained results ?

12. **(Programming Project - Quantum Teleportation)** Quantum teleportation is a process in which quantum information can be transmitted from one location to another, with the help of classical communication and previously shared quantum entanglement between the sending and receiving location. Implement this protocol on a real QPU. What is current state of art in practical implementation of quantum teleportation with quantum optical systems ?

13. **(Programming Project - Superdense Coding)** In quantum information theory, superdense coding (or dense coding) is a quantum communication protocol to transmit two classical bits of information from a sender (often called Alice) to a receiver (often called Bob), by sending only one qubit from Alice to Bob, under the assumption of Alice and Bob pre-sharing an entangled state. Implement this protocol on a real QPU.

14. **(Programming Project - Quantum Random Number Generators)** QRNGs are true random number generators exploiting the randomness of quantum physics. Study different quantum systems we can use to generate true random numbers. Why is quantum physics better than classical physics at generating randomness ? Implement a QRNG on a real QPU and compare its performances against a pseudo random number generator by performing statistical tests on obtained sequences.

15. **(Programming Project - Quantum Educational Game)** Create an educational game based on Qiskit to teach quantum computing to CPE students who didn't take this course. In first levels the player would be requested to build a quantum circuit generating a given quantum state from a gate set. In next levels more advanced quantum algorithms and protocols can be introduced.

16. **(Theory Project - Extend Algorithm)** Extend an algorithm from the `https://quantumalgorithmzoo.org/`. Pick an instance (or family of instances) of the algorithms where you can calculate concrete runtimes analytically.

17. **(Theory Project - Quantum Error Correction)** What is the quantum analog of error correcting codes ? How quantum error correction protects quantum information from noise ? Explain the concepts of syndrome, stabilizer, and the functioning of the Shor code as well as the surface codes for 2D architectures.

4. Important Dates

- **(April 15th)** Project proposals due
- **(April 28th)** Project progress update due
- **(May 10th)** Code repository and short report due
- **(May 15th)** Oral presentation (15+5 min)