One Path to Fluency in Quantum Programming

donny@ibm.com, 21-Sep-18

All self-learners have their own starting points and unique goals, but in my experience there are roughly 6 levels in the path to fluency in Quantum Programming ('fluency' defined in level 6 below). Note that I am also traveling along the below path, and would not present myself as fluent in Quantum Programming by any stretch of the imagination. For those starting without classical programming experience, there are many existing tutorials that I will not attempt to replicate here[1]. Note that the following is **completely exclusive** of any acquired understanding of Quantum Information Theory, Quantum Computing hardware, Quantum Characterization, Verification, & Validation (QCVV), and non-elementary Quantum Mechanics - meaning that you will not get any of the above for free and will have to learn them in their own time (each a very significant undertaking).

Levels of Quantum Programming Fluency:

1. Basic classical programming, boolean logic, maybe complexity theory and some machine learning (e.g. a Computer Science undergrad)
2. Basic understanding of quantum mechanics, ability to follow quantum mechanical proofs with significant effort
3. Elementary Quantum Programming
    a. Basic understanding of Quantum Computing primitives - qubits, gates, circuits, rotations - ability to follow a circuit+equation based description of a simple quantum algorithm (Deutsch's, teleportation) with significant effort
    b. Familiarity with and understanding of cornerstone quantum algorithms - Simon's, Grover's, Bernstein-Vazirani, QFT, QPE, Shor's
4. Knowledge of a Quantum Computing software framework, including understanding of the computation model (grammar, operational data structures, compilation, etc.) and implementations of non-trivial algorithms
5. Advanced Quantum Algorithms
    a. Understanding of several advanced Quantum algorithms not found in most textbooks, such as VQE, QAOA, HHL, Hamiltonian simulation, QRAM, matrix manipulation (broadly), machine learning (broadly), or graphs (broadly)
    b. Ability to compose robust or single case implementations of quantum algorithms in software (for **simulators**, not quantum hardware)
6. Ability to write modifications or new algorithms based on known primitive forms (e.g. variation, QFT, search, evolution), both on paper and in software (when feasible)

---

[1] A long long time ago, I did Princeton's Cos126, which should probably do here with enough additional practice and elbow grease.

Steps Between Levels

The following is only a single suggestion among many for how to progress between levels. Common courses, books, or tutorials which I have not tried are included in italics at the reader's own risk.

A note on redundancy: The practice of Quantum Computing education is relatively young and still quite experimental. Each little subunit is its own challenge to explain and distill, and there is little consensus among practitioners about the best explanations or approaches. The game for new entrants is trying several (often n>=2) redundant resources in search of different explanations that decompose or distill a concept in different ways. Ideally one explanation sticks well - if not, the combined approaches often help. In my experience, for the tricker topics (Basic Quantum Mechanics, for example), 2-3 resources might be necessary to achieve an adequate number of "clicked" subunits. I've included notes on suggested redundancy in each section below. Note that a ".5" resource means a less exhaustive review (such as reading alone without completing exercises), though I don't suggest skimming without understanding. If you're going to review anything, try to do so until it clicks.

A note on other resources: There are **many** excellent resources not included below. I've tried to construct a single path through the sea of resources, but you **must** Google aggressively and explore liberally outside of the below to succeed.

1→2. This is more a task of doing exercises and becoming comfortable than learning specific hard elements. The reader should engage in the below until they feel 60-70% comfortable with Quantum Mechanical notation, algebra, and high-level behavior and forms (e.g. linearity, phase, measurement, etc.)
   a. Nielsen and Chuang, "Quantum Computation and Quantum Information," Chapter 2 (Chapter 1 is not about QM, but read it beforehand anyway)
   b. Feynman Lectures on Physics, Volume III - Chapters 1-11
   c. MIT Quantum Physics I and Quantum Physics II (or EdX versions)
   d. Griffiths' "Introduction to Quantum Mechanics"
   e. Suggested redundancy: 2.5

2→3. Elementary Quantum Computing
   a. Nielsen and Chuang, Chapters 1, 4, 5, 6
   b. IBM Q Experience Introductory User Guide
   c. MIT's Quantum Information I Course, Parts 1 and 2
   d. Suggested redundancy: 1.5

3→4. Quantum Information Software
   a. Install Qiskit Terra and Aqua, and execute some of the intro examples (here) to understand the environment setup
   b. Read as much of the Qiskit documentation as you can to get a feel for the computation model of Terra

c. Step through (line by line in the Pycharm debugger) one of the Qiskit algorithm tutorials
d. Step through (line by line in the Pycharm debugger) the Aqua qkernel SVM tutorial or the Aqua variational SVM tutorial (or any other Aqua tutorial). Notebooks are available for these as well, but I'm not sure how nicely they play with the debugger
e. Suggested redundancy: 1 (not much re-reading required)

4→5. Advanced Quantum Algorithms

a. Pick an advanced quantum algorithm, such as VQE, QAOA, or Variational SVM, and learn it from the initial paper through Aqua implementation (all three of the above have implementations in Aqua)
b. LANL's Quantum Algorithm Implementations for Beginners
c. MIT's Quantum Information II Course, Part 3
d. Suggested redundancy: 1.5

5→6. Contributing to Quantum Software or Theory

a. Attempt to modify or expand an existing algorithm in Aqua, write a new Aqua algorithm based on a paper, or write a new algorithm entirely both on paper and in Aqua (suggestion: if you have expertise in another area, such as finance or machine learning, start there). Contact someone on the Aqua team over Qiskit Slack for more explicit guidance about which algorithms or extensions are valuable, and could potentially be checked in when complete

The importance of having an experienced guide

Learning Quantum Programming is a minefield of confusing notation and "am I crazy or is this crazy" moments for the classically trained programmer. In the current Quantum Programming education state of the art, having an outlet for questions is critical. At best it can accelerate learning several-fold, and at worst the reader may be unable to proceed without it. Luckily, the field is still challenging enough that serious self-learners are not very common and the community is quite friendly to these efforts. If you don't have a physicist around to answer your questions, your best bet may be to hire a Quantum Physics (ideally Quantum Information or Quantum Computing) PHD student or post-doc for tutoring sessions where you can ask your accumulated questions. **I cannot stress how invaluable this is enough**.

Feedback

I would greatly appreciate input and feedback, especially from users of the guidance above. Please send comments to donny@ibm.com.