Quantum Computation

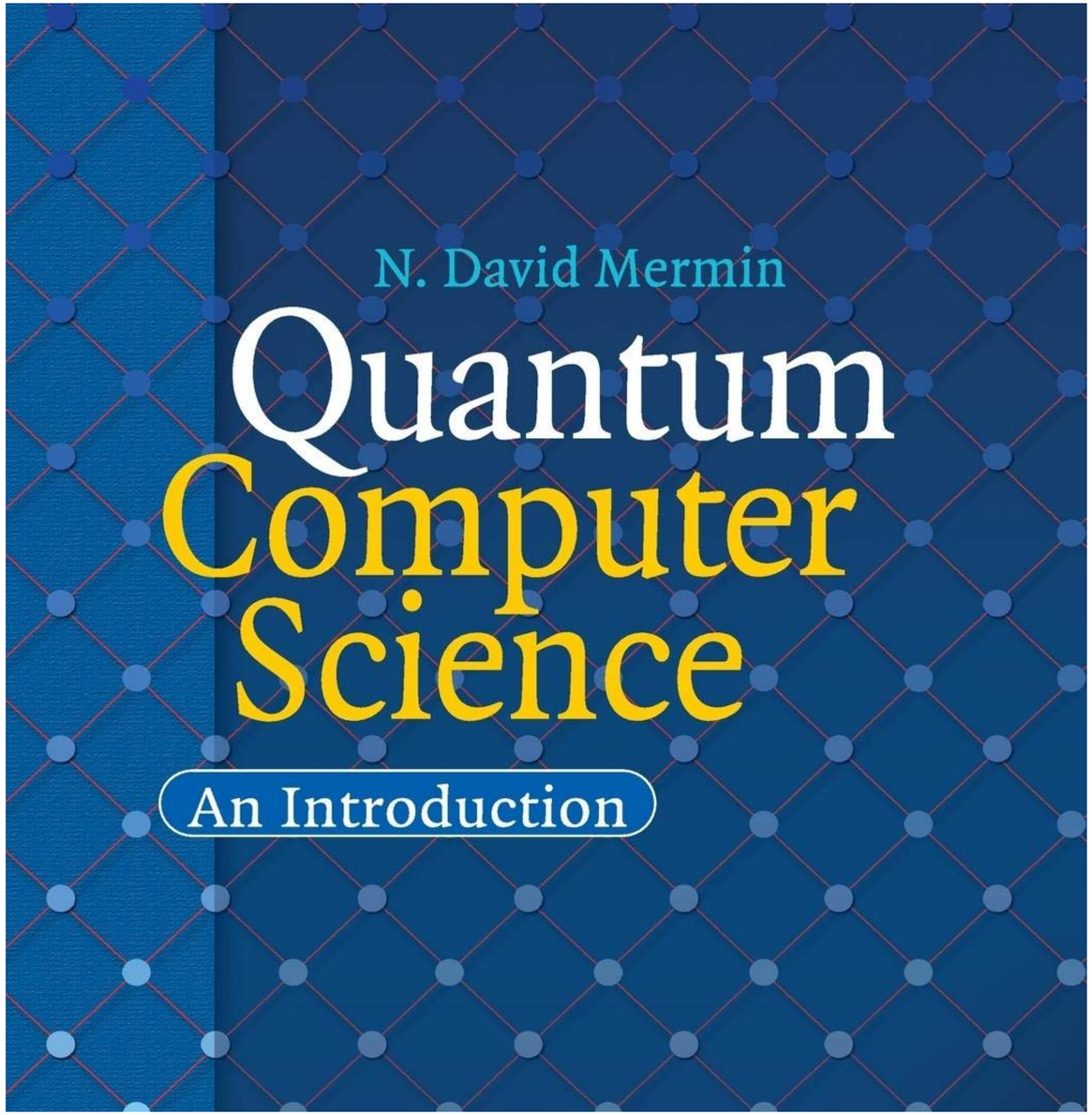
An introduction

Zilin Jiang, Fall 2024

Administrative stuff

What you need to know

- Email: zilinj@asu.edu
  - mention something quantum in your email
  - use for individual administrative questions
- Office hours: Tuesday and Thursday, 1:30pm-2:30pm, WXLR A839
- Homework 60% (6 assignments) + Midterms (9/24 & 10/31, in class) 20% + Final (12/10 2:30pm-4:20pm) 20%



What is quantum computation?

Are quantum computers more “powerful” than classical ones?

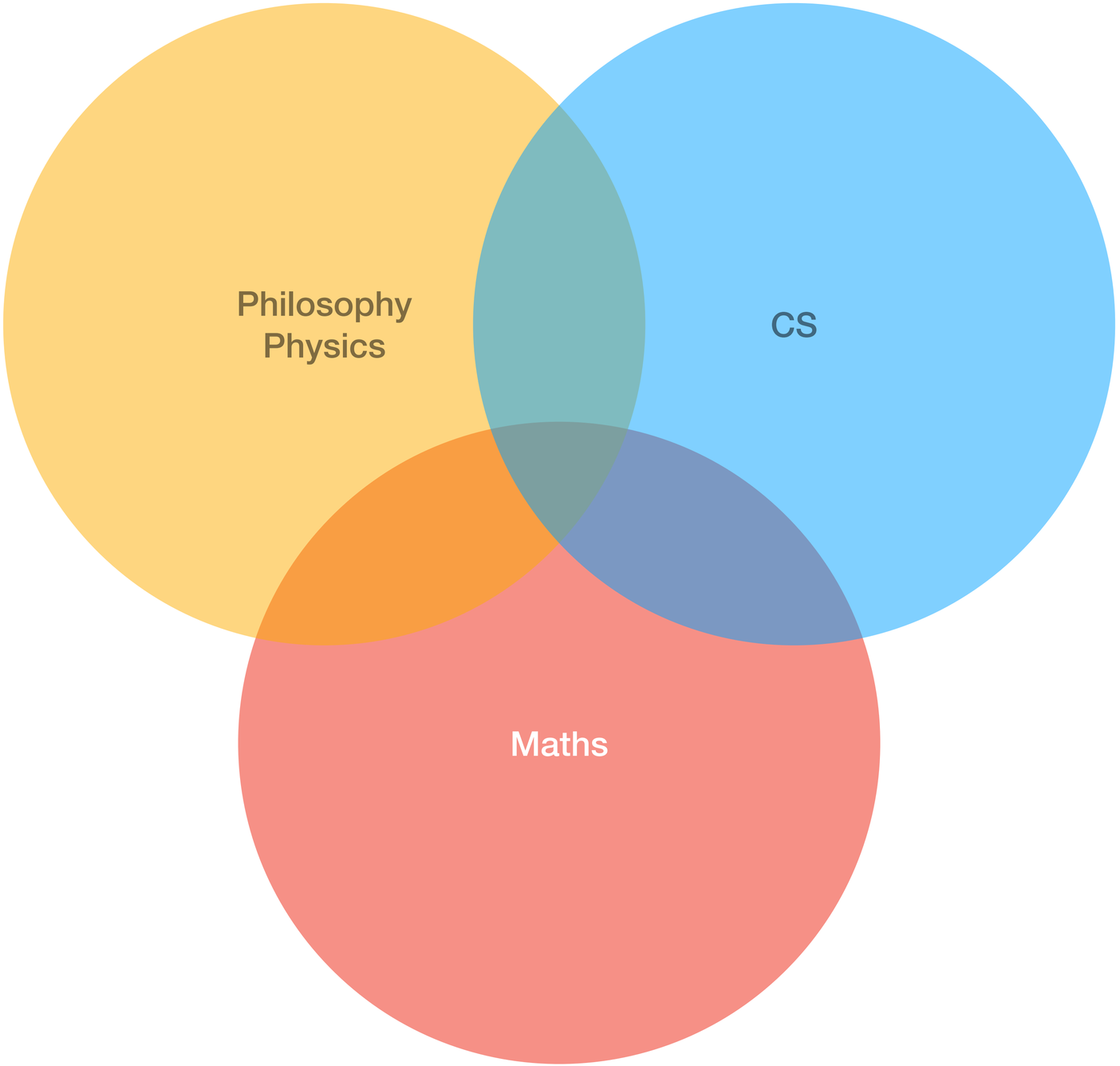
For which computational tasks?

“Quantum computing is … nothing less than a distinctively new way of harnessing nature…”

The Fabric of Reality, by David Deutsch

“… it will be the first technology that allows useful tasks to be performed in collaboration between parallel universe”

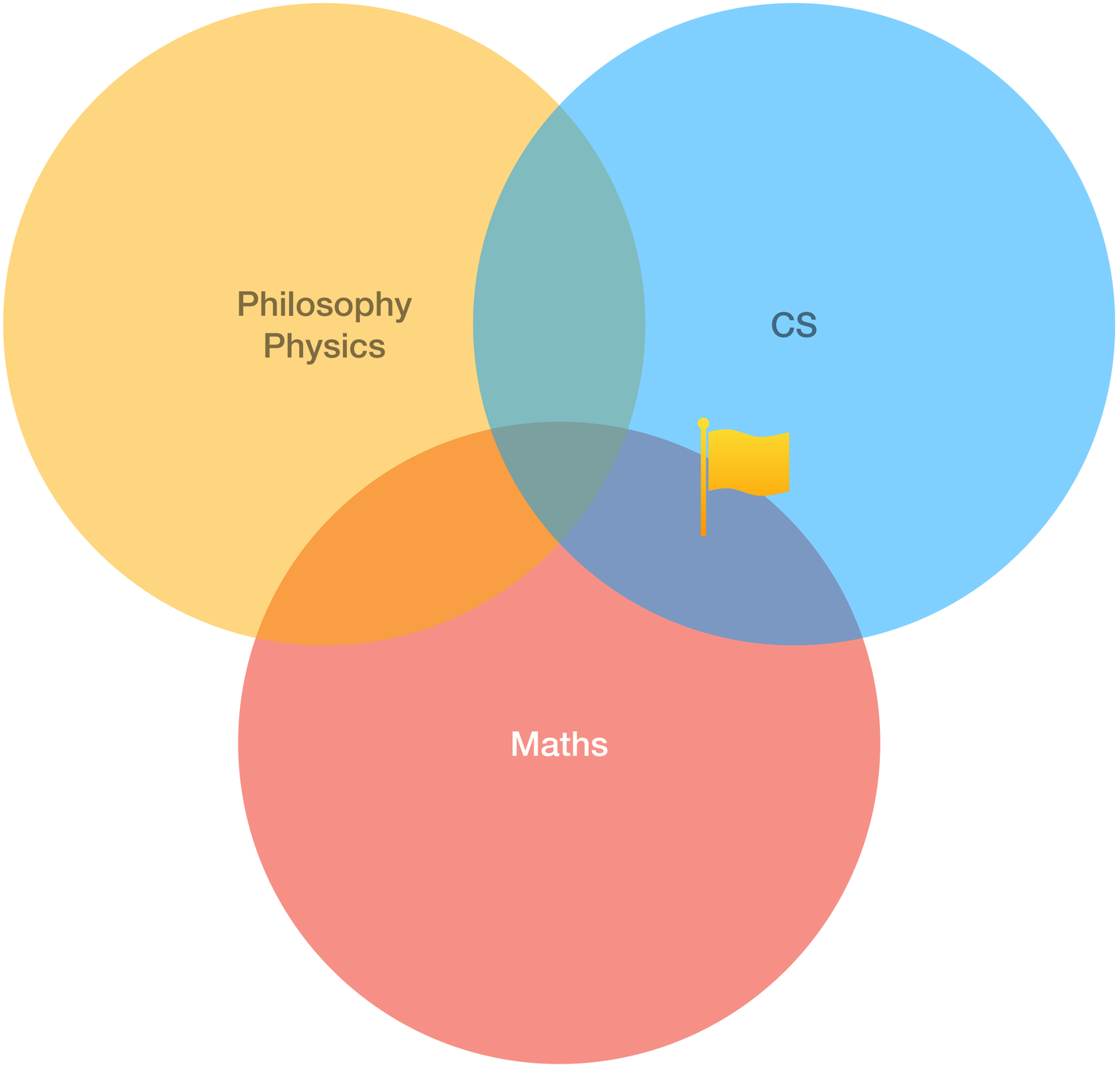
The Fabric of Reality, by David Deutsch


Philosophy
Physics
CS
Maths

“I think I can safely say that nobody understands quantum mechanics”

Richard Feynman, 1965

Feynman was referring to the “meaning” of quantum mechanics for “reality”. By way of contrast, the math of how to do quantum mechanics is perfectly clear, fine, and uncontroversial.


Philosophy
Physics
CS
Maths

Neumann, to a physicist seeking help with a difficult problem: Simple. This can be solved by using the method of characteristics.

Physicist: I'm afraid I don't understand the method of characteristics.

Neumann: In mathematics you don't understand things. You just get used to them.

John von Neumann

Computational challenge 1

Multiplication of 2 given n-digit numbers

Complexity / Efficiency = # of steps algorithm takes and how that scales as a function of input length.

What's the number of steps to multiply two n-digit numbers?

Naive algorithm: $O(n^2)$

Faster algorithm: $O(n \log n)$, use fast Fourier transform

Computational challenge 2

Factoring a given n-digit number

Example: input 35, output 5 x 7

Naive algorithm: $O(n^2)$

Faster algorithm: [Pollard 1996]

$$O(1000000^{\sqrt[3]{n}})$$

RSA-230: cracked in 2018

RSA-1024: worth $100,000; RSA-2048: worth $200,000

No one knows if there's polynomial time algorithm for factoring

Cryptography

Multiplication

Easy, encryption

Factoring

Hard, decryption

Peter Shor, 1994. A quantum computer could factor n-bit number in $O(n^2)$ steps.
Polynomial time!

Uses basic fact of quantum mechanics: given say 1000 photons / electrons, their “joint state” is defined by 2^1000 numbers, stored by Nature.

We would like to hack into Nature’s computer.

Shor’s algorithm from 1994 was based on an earlier quantum algorithm of Daniel Simon.

Who’s Everett, and what’s his interpretation? I was approaching the problem purely from a computer scientist's perspective. I learned the absolute bare minimum of physics I needed to be able to understand the computer science question, which (as I saw it) was, “these crazy people are claiming that if you add these very-weird-yet-theoretically-physically-implementable functions to a computer, then you should be able to do amazing things with them. Prove them right or wrong.” I actually started out trying to prove that quantum computing was useless, and eventually narrowed down the difficult, unsimulateable part [of QC's power] to, “Rotate, compute, rotate”. That helped guide my search for a computationally interesting quantum algorithm.

Daniel Simon

Disclaimer

Large quantum computer has not been built

Largest number factored using Shor’s algorithm is twenty-one

Quantum computation

$\approx$ Classical computation + 1 extra power

Probabilistic Computing

What if classical computing was augmented with randomness?

Deterministic code / circuit + coin flips (outputs 0 or 1 equally likely)

Question: Is probabilistic computing more powerful than classical deterministic computing?

Answer: Yes, by definition, at least as powerful as classical.

Maybe not, when just computing deterministic functions.

Question: Why would you want probabilistic computing for such tasks?

Answer: trading error probability for efficiency.

Some computational tasks can be solved more efficiently if we allow 1/10^100 (un-physically small) probability of error

Example 1

Matrix multiplication

Given two n by n matrices, how many arithmetic operations are needed to compute their product?

Naive algorithm: $O(n^3)$

Strassen's algorithm: $O(n^{2.807355})$

…

Le Gall (2014): $O(n^{2.3728639})$

BIG question: $O(n^2)$?

Question: Can we check efficiently the correctness of the “black box”?

It makes little sense to actually multiply A and B and compare with C

Answer: Yes, if we allow some slight probability of error

“Trade slight probability of error for efficiency”

Freivald's matrix multiplication checker

"Trade slight probability of error for efficiency"

Algorithm:

- Choose a random vector $x \in \{0,1\}^n$
- Calculate $y = A(Bx) - Cx$
- If $y = 0$, output CORRECT; otherwise, output INCORRECT.

Analysis:

- Performs n^2 arithmetic operations
- If$AB = C$, answers CORRECT with probability 100%
- If $AB \neq C$, answers INCORRECT with probability ≥ 50%