Special Lectures on Quantum Computing for Quantum Chemistry

Chong Sun, Rutgers University–New Brunswick

1. Introduction

Quantum computing uses quantum systems to perform computing tasks. Unlike classical matter treated as particles, quantum matter has wave–particle duality, where the wave feature allows superposition of states. This feature can, in principle, store more information than classical computers. The efficient representation of information enables efficient operations (quantum gates), which leads to potential quantum advantage in complicated tasks.

Simulating a quantum system is one of these complicated tasks. In 1981, Richard Feynman spoke at a conference on the topic "Simulating physics with computers" and left the famous quote that many quantum computing researchers now cite: *"Nature isn't classical, dammit, and if you want to make a simulation of Nature, you'd better make it quantum mechanical, and by golly it's a wonderful problem because it doesn't look so easy."*

The immediate fruit of this idea is ***analog quantum computing***, where a controllable quantum system is built to mirror another quantum system. One can engineer the Hamiltonian of the analog simulator, and the natural evolution of the simulator represents the time evolution of the Hamiltonian of interest. Examples of analog quantum simulators include ultracold atoms in optical lattices,[1] trapped ions,[2] neutral atom arrays,[3] and photonic networks.[4] Note that the last three platforms can also be designed as digital quantum computers, explained below.

Another important type of quantum computer is the ***digital quantum computer***, which is more universal and not constrained to specific problems. Digital quantum computers resemble the computers we use daily, using sequences of discrete operations (quantum gates) to perform computation. The most well-known digital quantum computers are superconducting devices, which won the Physics Nobel Prize this year. In addition to the three platforms mentioned earlier (trapped ions, neutral atom arrays, and photonic networks), spin qubits, such as NV-center spins, are also an important digital quantum platform. Spin qubits are highly relevant to chemistry, as *molecular spin qubits* are an active research area.[5]

Here is a non-exhaustive list of companies that make quantum hardware: QuEra Computing (neutral atoms), Pasqal (neutral atoms), Atom Computing (neutral atoms), IonQ (trapped ions), PsiQuantum (photonic), Xanadu (photonic), Quantinuum (trapped ions), IBM (superconducting), Google (superconducting), Rigetti (superconducting), Intel (superconducting, spin).

In this lecture, we will focus on digital quantum computers.

2. Quantum computing in a nutshell

This section will prepare you with all the concepts about quantum computing you need for the rest of the lecture.

2.1. Math background

2.1.1. Dirac notation

We use Dirac notation for a quantum state, where $|\phi\rangle$, called a ***ket***, is a quantum state denoted by ϕ, and $\langle\phi|$, called a ***bra***, is the ***Hermitian adjoint*** (complex-conjugate transpose) of $|\phi\rangle$:

$$\langle\phi| = (|\phi\rangle)^\dagger. \tag{1}$$

We use uppercase Latin letters for ***operators***[1]. An operator acts on a quantum state and produces another quantum state, e.g.,

$$A|\phi\rangle = |\phi'\rangle. \tag{2}$$

Operators also have Hermitian adjoints. The Hermitian adjoint of A is denoted by $A^\dagger$, and

$$(A|\phi\rangle)^\dagger = \langle\phi|A^\dagger. \tag{3}$$

One can perform integrals (contraction) with quantum states and operators, just as in previous lectures:

$$\begin{aligned}\langle\phi_1|\phi_2\rangle &= \int \mathrm{d}r\phi_1^*(r)\phi_2(r),\\ \langle\phi_1|A|\phi_2\rangle &= \int \mathrm{d}r\phi_1^*(r)A\phi_2(r).\end{aligned} \tag{4}$$

2.1.2. Matrix representation

You have learned the concept of a ***basis set*** in earlier lectures and that a quantum state can be represented as a linear combination of basis states. Given a basis set, we can collect the linear-combination coefficients into a ***vector*** to represent a quantum state–a ***discrete*** representation.

Let the basis set be $|\phi_1\rangle, |\phi_2\rangle, |\phi_3\rangle$, and

$$|\Psi\rangle = c_1|\phi_1\rangle + c_2|\phi_2\rangle + c_3|\phi_3\rangle, \tag{5}$$

Then we can represent $|\Psi\rangle$ as a column vector, and $\langle\Psi|$ as a row vector:

$$|\Psi\rangle = (c_1, c_2, c_3)^T, \quad \langle\Psi| = (c_1, c_2, c_3). \tag{6}$$

Note that we use lowercase Latin letters for numbers (scalars).

Since an operator transforms one state into another, its discrete representation is a ***matrix***. Using the

[1] You might have seen a small hat on top of operators, $\hat{A}$. Here we omit the hat, which is the most accepted convention nowadays.

same basis, the matrix form of operator A is

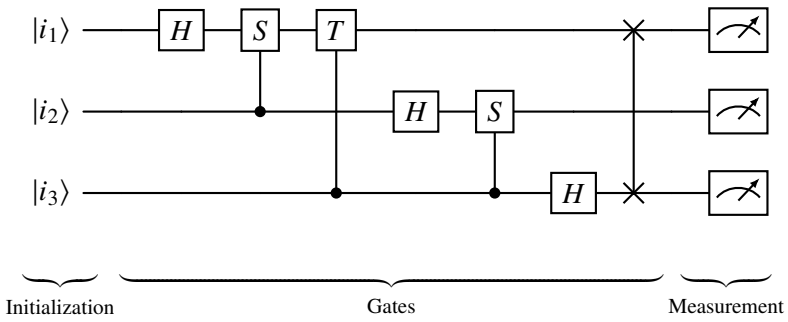
$$A = \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix}, \tag{7}$$

where $A_{ij} = \langle \phi_i | A | \phi_j \rangle$.

2.2. Quantum circuits

A quantum circuit is an analog to a classical circuit. A classical circuit is composed of bits (0 or 1), logic gates (e.g., AND, OR, NOT, XOR), and wires that connect the gates. For humans to use computers, we also need to convert human inputs and outputs into/from a form understood by the circuit. Similarly, a quantum computer is composed of ***qubits*** ($|0\rangle$ or $|1\rangle$), ***quantum gates*** (e.g., X, Y, Z, H), and ***measurement***, which transforms quantum information into classical information for humans to interpret. A quantum computer also has "wires" that connect quantum gates to perform complicated operations.

We first show an example of a quantum circuit and then explain each component.



In the above circuit, each horizontal wire represents a qubit, and the qubits go through three stages. First, a qubit is initialized to a specific state $|i_r\rangle_{r=1,2,3}$, and the full quantum state is $|i_1 i_2 i_3\rangle$. Common initialization choices for a single qubit are $|0\rangle$ or $|1\rangle$. Next, a sequence of gates is applied to one or more qubits to transform the initial state into another state. Finally, measurements are applied to all or some of the qubits, producing classical information that humans can read. The above circuit performs a three-qubit quantum Fourier transform.

2.2.1. Qubits

A qubit is the smallest building block of a quantum computer. Like a classical bit that can take the values 0 or 1, a qubit has two orthogonal eigenstates $|0\rangle$ and $|1\rangle$. However, since a quantum state can be a superposition of multiple eigenstates, a qubit can also take the form

$$|\psi\rangle = a|0\rangle + b|1\rangle, \quad \langle \psi | \psi \rangle = |a|^2 + |b|^2 = 1, \tag{8}$$

where a and b are complex numbers. We take $\{|0\rangle, |1\rangle\}$ as the computational basis, just like basis sets in quantum chemistry. The superposition feature is key to why a quantum state can store information exponentially more efficiently than a classical state, as explained below.

Let us reparameterize a and b as

$$|\psi\rangle = \cos(\theta/2)|0\rangle + e^{i\phi}\sin(\theta/2)|1\rangle, \tag{9}$$

so that $|\psi\rangle$ lives on a unit sphere in three dimensions, shown in Figure 1. This sphere is called the ***Bloch sphere***.

Suppose we need one classical bit for θ and one classical bit for ϕ. Then two classical bits are required to represent a vector on the Bloch sphere[2], while only one qubit is needed. For multiple qubits, the dimension of the state space grows exponentially with the number of qubits, and thus the number of classical bits needed to represent the state also grows exponentially.

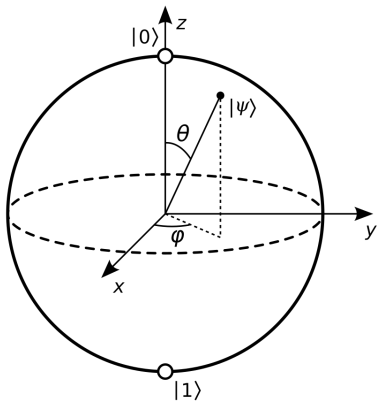


Figure 1: Bloch sphere representation of a qubit. (Figure source: Wikipedia)

To illustrate the exponential relationship, consider the 2-qubit case. An arbitrary 2-qubit state can be written in the 2-qubit computational basis as

$$|\psi_2\rangle = a|00\rangle + b|01\rangle + c|10\rangle + c|11\rangle, \quad \langle\psi|\psi\rangle = |a|^2 + |b|^2 + |c|^2 + |d|^2 = 1, \tag{10}$$

where a, b, c, d are complex numbers. A vector of four complex numbers has eight real degrees of freedom. The normalization constraint removes one degree of freedom, and removing the global phase removes another. Therefore, a 2-qubit Bloch-sphere–like manifold has $8 - 2 = 2^3 - 2 = 6$ degrees of freedom.

Following the same procedure, an N-qubit state has

$$2^{N+1} - 2 \tag{11}$$

degrees of freedom in classical representation. In other words, representing an N-qubit state on a classical computer requires $(2^{N+1} - 2)$ real parameters.

In the following, we will use the computational basis, and an N-qubit state will be represented as a vector of 2^N complex numbers.

1-qubit
$$|0\rangle = \begin{pmatrix}1\\0\end{pmatrix}, \quad |1\rangle = \begin{pmatrix}0\\1\end{pmatrix}, \quad a|0\rangle + b|1\rangle = \begin{pmatrix}a\\b\end{pmatrix}. \tag{12}$$

2-qubit
$$|00\rangle = \begin{pmatrix}1\\0\\0\\0\end{pmatrix}, \quad |01\rangle = \begin{pmatrix}0\\1\\0\\0\end{pmatrix}, \quad |10\rangle = \begin{pmatrix}0\\0\\1\\0\end{pmatrix}, \quad |11\rangle = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}, \quad |\psi_2\rangle = \begin{pmatrix}a\\b\\c\\d\end{pmatrix}. \tag{13}$$

[2] θ and ϕ are real numbers represented as floats on a classical computer. A standard float takes 32 bits and a double takes 64 bits. Here, we use 1 bit per parameter for simplicity, which does not affect the scaling argument.

2.2.2. *Quantum gates*

Quantum gates apply to one or multiple qubits. They are ***unitary operators***, since we require the quantum state to remain normalized.

One-qubit gates

Standard 1-qubit gates include the following six gates:

$$\begin{aligned}
&\text{Pauli-}X \quad \text{—}\boxed{X}\text{—} && X|0\rangle = |1\rangle, \quad X|1\rangle = |0\rangle \\
&\text{Pauli-}Y \quad \text{—}\boxed{Y}\text{—} && Y|0\rangle = i|1\rangle, \quad Y|1\rangle = -i|0\rangle \\
&\text{Pauli-}Z \quad \text{—}\boxed{Z}\text{—} && Z|0\rangle = |0\rangle, \quad Z|1\rangle = -|1\rangle \\
&\text{Hadamard} \quad \text{—}\boxed{H}\text{—} && H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle, \quad H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |-\rangle \\
&\text{Phase} \quad \text{—}\boxed{S}\text{—} && S|0\rangle = |0\rangle, \quad S|1\rangle = i|1\rangle \\
&\pi/8 \quad \text{—}\boxed{T}\text{—} && T|0\rangle = |0\rangle, \quad T|1\rangle = e^{i\pi/4}|1\rangle
\end{aligned} \tag{14}$$

Their matrix forms are

$$\begin{aligned}
X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \\
H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \quad T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}
\end{aligned} \tag{15}$$

We can also define parameterized rotation gates, which rotate the qubit around a coordinate axis. Treating the qubit as a spin makes this intuitive. A rotation gate takes the form

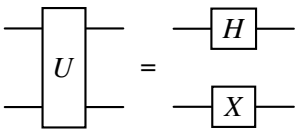
$$R_\alpha(\theta) = e^{-i\frac{\theta}{2}\sigma_\alpha}, \quad \text{—}\boxed{R_\alpha(\theta)}\text{—}, \quad \sigma_\alpha = X, Y \text{ or } Z \tag{16}$$

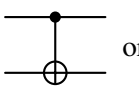
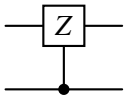
We can multiply one-qubit gates to obtain another one-qubit gate, just like matrix multiplication, e.g.,

$$U = HX, \quad \text{—}\boxed{U}\text{—} = \text{—}\boxed{H}\text{—}\boxed{X}\text{—} \tag{17}$$

Two-qubit gates

A two-qubit gate can be one of two types:

- Stacking two one-qubit gates, e.g, 

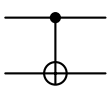
- Controlled gate, e.g.,  or 

For the first type, the two qubits are operated on independently by single-qubit gates, and they do not communicate. In the example above, U is the tensor product $H \otimes X$. A ***Pauli string*** is a tensor product of Pauli gates. For an n-qubit system,

$$P = P_1 \otimes P_2 \otimes \cdots \otimes P_n, \quad P_i = I, X, Y, \text{ or } Z \tag{18}$$

For the second type, the behavior of the target qubit depends on the control qubit, so the qubits become ***entangled***. An entangling gate cannot be written as a tensor product of smaller gates.

Controlled gates are the basic mechanism for entangling qubits. A prototypical example is the CNOT gate:

$$\text{CNOT} \quad \begin{matrix}\text{control qubit} \\ \text{target qubit}\end{matrix} \; \text{  }, \quad \text{CNOT}|0i\rangle = |0i\rangle, \quad \text{CNOT}|1i\rangle = |1\bar{i}\rangle, \tag{19}$$

where $\bar{i}$ denotes flipping the bit value: $\bar{0} = 1$ and $\bar{1} = 0$. Thus, if the control qubit is 0, nothing happens to the target qubit; if the control qubit is 1, the target qubit is flipped.

Multi-qubit gates

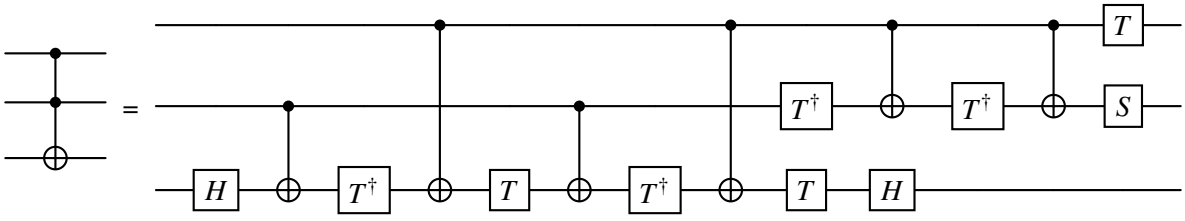
A multi-qubit gate can be decomposed into CNOTs and single-qubit gates, supported by the theorem of ***universal quantum gates***.

Theorem: Any unitary operation on any number of qubits can be decomposed into a sequence of gates from a universal gate set.

We will not prove this theorem, but we list some examples of universal gate sets:

- $\{\text{CNOT}, H, T, S\}$
- $\{\text{CNOT}\} \cup \{\text{all single-qubit gates}\}$

As an example, consider the Toffoli gate. This 3-qubit gate flips the third qubit only when the first two qubits are $|1\rangle$. It can be decomposed into $\{\text{CNOT}, H, T, S\}$ as follows:



Implementing a Toffoli gate requires 10 single-qubit gates and 6 CNOTs! In most quantum hardware, entangling multi-qubit gates are decomposed into one- and two-qubit gates. Therefore, one way to make a quantum circuit less expensive is to avoid entangled multi-qubit gates when possible.

2.2.3. Measurement

Measurement connects quantum information to classical information. A measurement collapses a quantum state onto a measurement basis. Usually, we take the computational basis as the measurement basis.

Consider a one-qubit state with $|a| > 0$ and $|b| > 0$:

$$|\psi\rangle = a|0\rangle + b|1\rangle, \quad \langle\psi|\psi\rangle = |a|^2 + |b|^2 = 1. \tag{20}$$

Measuring $|\psi\rangle$ in the computational basis $\{|0\rangle, |1\rangle\}$ gives

$$|\langle 0|\psi\rangle|^2 = |a|^2, \quad |\langle 1|\psi\rangle|^2 = |b|^2, \tag{21}$$

A single measurement collapses $|\psi\rangle$ onto either $|0\rangle$ or $|1\rangle$. This procedure is a positive operator-valued measure (POVM). However, one measurement alone tells us nothing about the original state.

The outcome is not arbitrary: the probability of measuring 0 is $|a|^2$, and the probability of measuring 1 is $|b|^2$. To extract information, we must ***repeat the measurement***. After many repetitions, we obtain good estimates of $|a|^2$ and $|b|^2$.

This still does not determine the full state, since the relative phase between a and b cannot be recovered. To obtain phase information, we use a different measurement basis, such as $|+\rangle, |-\rangle$. Measuring in additional bases provides different "angles" from which we can recover the missing phase information. The outcomes are then combined to reconstruct the quantum state, following the idea of quantum state tomography (QST).

In practice, we usually do not directly measure the full state $|\psi\rangle$, but rather the expectation value of a physical observable O: $\langle\psi|O|\psi\rangle$, which is always real. The standard approach is to decompose O into a linear combination of Pauli strings, whose eigenstates are known. Each Pauli string defines a measurement basis. By projecting $|\psi\rangle$ onto these bases and repeating the measurement many times, we collect statistics for each Pauli term. The expectation value of O is then reconstructed from the weighted sum of these contributions.

3. Quantum Algorithms

A quantum algorithm is a computational procedure that uses quantum states, unitary operations, and quantum measurements to solve a problem (hopefully) faster or more efficiently than classical methods. The key component is the quantum circuits we discussed earlier. In practice, a quantum algorithm includes both classical processing and quantum circuits.

Well-known quantum algorithms include the quantum Fourier transform (QFT), phase estimation (PE), Grover's search, and Shor's algorithm. We will not discuss these algorithms here; the reader may refer to standard references such as the Nielsen & Chuang textbook.[6] Instead, we will focus on simulating a chemical system. Before introducing quantum simulations, we will discuss the current limitations of quantum computers.

3.1. Limitations of quantum computers

A quantum computer is an open quantum system, which constantly interact with the environment, and thus is bounded to ***decoherence*** and ***dissipation***.

Decoherence

A pure state becomes a mixed state due to entanglement with the environment.

The density matrices of a pure state and a mixed state has different forms

$$\begin{aligned} &\text{pure state:} \quad \rho_{\text{pure}} = |\Psi\rangle\langle\Psi|, \quad |\Psi\rangle = \sum_{i=1}^{N} c_i |\psi_i\rangle, \\ &\text{mixed state:} \quad \rho_{\text{mixed}} = \sum_{i=1}^{N} p_i |\psi_i\rangle\langle\psi_i|, \quad \sum_{i=1}^{N} p_i = 1. \end{aligned} \tag{22}$$

Example A pure state $|\Psi\rangle = \frac{1}{2}(|0\rangle + |1\rangle)$ experiences the following decoherence

$$\rho(0) = \frac{1}{2}\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \rightarrow \rho(t) = \frac{1}{2}\begin{pmatrix} 1 & e^{-t/T_2} \\ e^{-t/T_2} & 1 \end{pmatrix} \rightarrow \rho(\infty) = \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \tag{23}$$

The above process is called *dephasing*. The time constant T_2 is a key metric in quantum computing, known as the ***dephasing time***, and characterizes how quickly phase coherence is lost due to environmental noise.

Dissipation

While decoherence describes the loss of phase information (interference), **dissipation** describes the loss of *energy* from the system into the environment. You can prove that the total energy of the pure and mixed states in Eq. (22) remains the same. In dissipation, population leaves excited states and relaxes toward lower-energy states due to coupling with environmental degrees of freedom.

Example A quantum state is at $|1\rangle$, with dissipation, it relaxes to the state $|0\rangle$ with lower energy.

$$\rho(0) = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \rightarrow \rho(t) = \begin{pmatrix} 1 - e^{-t/T_1} & 0 \\ 0 & e^{-t/T_1} \end{pmatrix} \rightarrow \rho(\infty) = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}. \tag{24}$$

The T_1 value is another key metric known as ***relaxation time***.

Other error sources include *gate errors*, which arise from imperfect control during gate operations; *crosstalk errors*, where operating on one qubit unintentionally disturbs nearby qubits; and *measurement errors*, which result from statistical fluctuations and finite readout fidelity. Another important metric is how fast a gate can be applied relative to the T_1 and T_2 values. Faster gate operations allow the circuit to execute more layers within the coherence time, enabling deeper coherent quantum circuits.

To suppress errors, quantum computers must use ***quantum error correction***. However, error-correcting codes require a large number of physical qubits for every logical qubit. An example of error correction, surface code, is shown in Figure 2, which illustrates the significant amount of qubits needed for error correction. Current devices do not yet have enough high-fidelity qubits to implement full error correction, and therefore cannot reach ***fault-tolerant quantum computing***.

With the above limitations, current quantum computers are referred to as *noisy intermediate-scale quantum* (NISQ) computers, a term introduced by John Preskill.

3.2. NISQ algorithms

Because current quantum computers are noisy and too small to support full error correction (the NISQ era), a more practical path is to use *NISQ algorithms*. NISQ algorithms have the following features:

- **Shallow circuits**: limited circuit depth to fit within the coherence time.
- **Hybrid quantum–classical loops**: the quantum processor prepares states and measures observables, while optimization is performed classically.
- **Heavy use of measurement**: observables are estimated by repeated sampling from the quantum device.
- **Error mitigation instead of error correction**: noise is reduced through calibration and post-processing rather than fault-tolerant codes.

We will introduce one representative NISQ algorithm, the *variational quantum eigensolver* (VQE), in the following section.

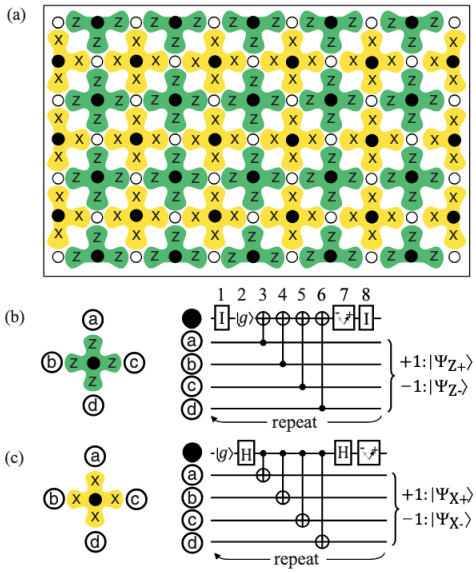


Figure 2: (a) Surface code. The white (black) circles are the data (syndrome) qubits and the green (yellow) clover structures are the vertex (face) plaquettes. (b), (c) Syndrome extraction circuits for the two types of stabilizer generators for the surface code, respectively.[7]

4. Quantum chemical simulation

We showcase how to solve a quantum chemistry problem with quantum simulations. We focus on solving the electronic structures.

4.1. Representing a chemical Hamiltonian

A chemical Hamiltonian contains the kinetic and Coulomb energies of electrons and nuclei. Under the Born–Oppenheimer approximation, the nuclei are treated as fixed, and we focus on the electronic Hamiltonian only. To represent this Hamiltonian on qubits, we first express it in a finite orbital basis $\{|\phi_p\rangle\}$. This converts the continuum electronic problem into a ***discrete*** second-quantized Hamiltonian suitable for quantum computation.

$$H = \sum_{pq} h_{pq}\, a_p^\dagger a_q \;+\; \frac{1}{2} \sum_{pqrs} V_{pqrs}\, a_p^\dagger a_q^\dagger a_s a_r, \tag{25}$$

where the one-electron and two-electron integrals are

$$\begin{aligned} h_{pq} &= \int \phi_p^*(\mathbf{r}) \left(-\frac{1}{2}\nabla^2 - \sum_A \frac{Z_A}{|\mathbf{r} - \mathbf{R}_A|} \right) \phi_q(\mathbf{r})\, \mathrm{d}\mathbf{r}, \\ V_{pqrs} &= \int\int \frac{\phi_p^*(\mathbf{r}_1)\phi_q^*(\mathbf{r}_2)\phi_r(\mathbf{r}_1)\phi_s(\mathbf{r}_2)}{|\mathbf{r}_1 - \mathbf{r}_2|}\, \mathrm{d}\mathbf{r}_1\, \mathrm{d}\mathbf{r}_2. \end{aligned} \tag{26}$$

Here $a_p^\dagger$ and a_p are fermionic creation and annihilation operators satisfying the anti-commutation relateion $\{a_p, a_q^\dagger\} = \delta_{pq}$. $a_p^\dagger$ creates an electron in the p-th orbital, and a_p removes an electron from the p-th orbital.

4.1.1. Fermion–spin transformation

Electrons are fermions, while qubits behave like spins. Therefore, we cannot directly represent the fermionic Hamiltonian in Eq. (25) on qubits. A standard procedure is to apply a fermion-to-spin transformation, which rewrites fermionic creation and annihilation operators in terms of Pauli operators. Below we list several transformations commonly used in quantum simulation software, without giving their explicit formulae. Interested readers may consult the references for more details.

- Jordan–Wigner (JW) transformation.
- Bravyi–Kitaev (BK) transformation.
- Bravyi–Kitaev superfast (BKSF) transformation.
- Parity transformation.

The symmetry-conserving Bravyi–Kitaev (SCBK) mapping[8] used in the hands-on example is a symmetry-aware version of the BK transform. It exploits physical symmetries such as particle-number and spin conservation to reduce qubit count and shorten Pauli strings.

After the chosen fermion-to-spin transformation, the electronic Hamiltonian becomes a linear combination of Pauli strings:

$$H = \sum_k c_k P_k, \tag{27}$$

which can be directly implemented on a quantum computer.

4.2. Variational quantum eigensolver

The variational quantum eigensolver (VQE) is one of the most widely used NISQ algorithms and is especially important in quantum chemistry. VQE estimates the ground-state energy of a Hamiltonian using a hybrid quantum–classical optimization loop. It combines shallow quantum circuits with classical optimization and is therefore suitable for the limited coherence times of current devices.

Given a Hamiltonian as in Eq. (27), we prepare a parametrized guess (***ansatz***) of the ground state

$$|\psi(\boldsymbol{\theta})\rangle = U(\boldsymbol{\theta})|0\rangle, \tag{28}$$

where $U(\boldsymbol{\theta})$ is a multi-qubit parametrized quantum circuit.

The energy of $|\psi(\boldsymbol{\theta})\rangle$ is

$$E(\boldsymbol{\theta}) = \langle\psi(\boldsymbol{\theta})|H|\psi(\boldsymbol{\theta})\rangle = \sum_k c_k \langle\psi(\boldsymbol{\theta})|P_k|\psi(\boldsymbol{\theta})\rangle, \tag{29}$$

We use the quantum computer to prepare $|\psi(\boldsymbol{\theta})\rangle$ and estimate all $\langle\psi(\boldsymbol{\theta})|P_k|\psi(\boldsymbol{\theta})\rangle$, and then use a classical optimizer to minimize $E(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$. VQE converges to the ground state if the ansatz is expressive enough and the classical optimization succeeds. A VQE loop is shown in Figure 3.

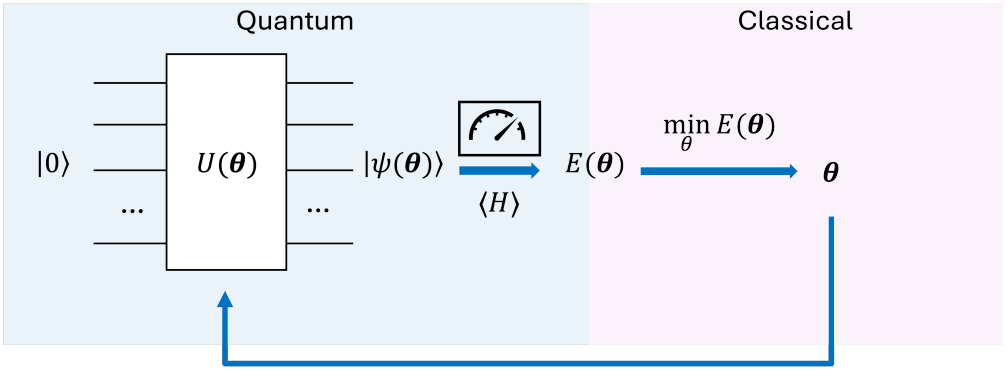


Figure 3: VQE algorithm.

4.2.1. Measuring Pauli strings

How do we measure $\langle\psi(\boldsymbol{\theta})|P_k|\psi(\boldsymbol{\theta})\rangle$ on a quantum computer? Each Pauli string is a tensor product of single-qubit Pauli operators. Therefore, its eigenstates are tensor products of the eigenstates of the individual Pauli operators:

$$\begin{cases} I|0\rangle = |0\rangle \\ I|1\rangle = |1\rangle \end{cases}, \quad \begin{cases} X|+\rangle = |+\rangle \\ X|-\rangle = -|-\rangle \end{cases}, \quad \begin{cases} Y|y_+\rangle = |y_+\rangle \\ Y|y_-\rangle = -|y_-\rangle \end{cases}, \quad \begin{cases} Z|0\rangle = |0\rangle \\ Z|1\rangle = -|1\rangle \end{cases}, \tag{30}$$

where $|y_\pm\rangle = (|0\rangle \pm i|1\rangle)/\sqrt{2}$.

Example The eigenvalues and eigenstates of the Pauli string $P = X \otimes Z$ follow directly from Eq. (30):

$$\begin{aligned} P(|+\rangle \otimes |0\rangle) &= +|+\rangle \otimes |0\rangle, \quad P(|+\rangle \otimes |1\rangle) = -|+\rangle \otimes |1\rangle, \\ P(|-\rangle \otimes |0\rangle) &= -|-\rangle \otimes |0\rangle, \quad P(|-\rangle \otimes |1\rangle) = +|-\rangle \otimes |1\rangle, \end{aligned} \tag{31}$$

Suppose $P_k|p_i^k\rangle = \lambda_i^k|p_i^k\rangle$. Then

$$\langle\psi(\boldsymbol{\theta})|P_k|\psi(\boldsymbol{\theta})\rangle = \lambda_i^k\langle\psi(\boldsymbol{\theta})|p_i^k\rangle\langle p_i^k|\psi(\boldsymbol{\theta})\rangle = \lambda_i^k|\langle p_i^k|\psi(\boldsymbol{\theta})\rangle|^2. \tag{32}$$

Thus, to measure $E(\boldsymbol{\theta})$, we rotate the quantum state into the eigenbasis of P_k and perform projective measurements. VQE is often measurement-heavy, which is characteristic of NISQ algorithms.

4.2.2. Choices of ansatz

The ansatz determines the circuit depth, expressiveness, optimization efficiency, and hardware compatibility of VQE. Common choices include:

- *Hardware-efficient ansatz.* Constructed from single-qubit rotations and local entangling gates. This

ansatz is generally shallow and well-suited for near-term devices, but may involve a large number of variational parameters and can be difficult to optimize.

- *Chemistry-inspired ansatz.* Derived from electronic structure theory, such as the unitary coupled cluster (UCC) family. These ansatzes encode chemical intuition and often require fewer parameters, but they may lead to deeper circuits depending on the truncation level and implementation.

We will not discuss these ansatzes in detail here, as (too) many references exist. In our hands-on practice, we will use the UCC ansatz.

4.2.3. Pros and cons

Advantages: (1) Shallow circuits; (2) Flexible ansatz design; (3) Compatible with error mitigation.

Limitations: (1) Noisy and slow classical optimization; (2) Strongly depends on ansatz quality; (3) Limited scalability.

4.3. Hands-on practice

We provide a hands-on practice of building a VQE algorithm to simulate a simple H_2 molecule, please refer to the colab file.

References

(1) Bloch, I. Ultracold quantum gases in optical lattices. *Nature physics* **2005**, *1*, 23–30.

(2) Brown, K. R.; Chiaverini, J.; Sage, J. M.; Häffner, H. Materials challenges for trapped-ion quantum computers. *Nature Reviews Materials* **2021**, *6*, 892–905.

(3) Bluvstein, D.; Levine, H.; Semeghini, G.; Wang, T. T.; Ebadi, S.; Kalinowski, M.; Keesling, A.; Maskara, N.; Pichler, H.; Greiner, M. **andothers** A quantum processor based on coherent transport of entangled atom arrays. *Nature* **2022**, *604*, 451–456.

(4) A manufacturable platform for photonic quantum computing. *Nature* **2025**, *641*, 876–883.

(5) Bao, Y.; Yu, S. S.; Anderegg, L.; Chae, E.; Ketterle, W.; Ni, K.-K.; Doyle, J. M. Dipolar spin-exchange and entanglement between molecules in an optical tweezer array. *Science* **2023**, *382*, 1138–1143.

(6) Nielsen, M. A.; Chuang, I. L., *Quantum computation and quantum information*; Cambridge university press: 2010.

(7) Fowler, A. G.; Mariantoni, M.; Martinis, J. M.; Cleland, A. N. Surface codes: Towards practical large-scale quantum computation. *Phys. Rev. A* **2012**, *86*, 032324.

(8) Bravyi, S.; Gambetta, J. M.; Mezzacapo, A.; Temme, K. Tapering off qubits to simulate fermionic Hamiltonians, 2017.