See The Variational Quantum Eigensolver: A review of methods and best practices for more information.

VQE (Variational Quantum Eigensolvers) uses a parametrized ansatz circuit, here represented as $\Psi(\theta)$ to get the ground state energy of the Hamiltonian.

$$E_{VQE} = \min_{\theta} \langle \Psi(\theta)|\mathbb{H}|\Psi(\theta)\rangle$$

The expectation value $\langle \Psi(\theta)|\mathbb{H}|\Psi(\theta)\rangle$ is done using a quantum computer (or simulator), while the minimization is done using classical computers.

To do VQE calculation in a quantum computer, we need to do the following

- Hamiltonian Preparation: Given a molecule, generate the Hamiltonian $\mathbb{H}$.
- Encoding: Convert the Hamiltonian $\mathbb{H}$ into a quantum circuit.
- Anszatz Selection and Preparation: Prepare a suitable ansatz circuit $\Psi(\theta)$
- VQE Algorithm: Do the minimization

Note 1. To run the code here, need to install the following on Python

```
pip install -U numpy scipy pyscf qiskit qiskit-nature qiskit-aer
```

1 Hamiltonian Preparation

This section concerns the preparation of the second quantization Hamiltonian.

1.1 Second Quantization

1.1.1 Hamiltonian representation for first quantization

The electronic structure can be described using the many-electron Schrödinger equation.

$$\mathbb{H} = \underbrace{-\sum_i \frac{\hbar^2}{2m_e}\nabla_i^2}_{K_e} \underbrace{-\sum_I \frac{\hbar^2}{2m_I}\nabla_I^2}_{K_I} + \underbrace{\frac{1}{2}\sum_{i\neq j}\frac{e^2}{4\pi\epsilon_0}\frac{1}{|\mathbf{r}_i-\mathbf{r}_j|}}_{V_{ee}} + \underbrace{\sum_{i,I}\frac{Z_I e^2}{4\pi\epsilon_0}\frac{1}{|\mathbf{r}_i-\mathbf{R}_I|}}_{V_{eI}} + \underbrace{\frac{1}{2}\sum_{I\neq J}\frac{Z_I Z_J e^2}{4\pi\epsilon_0}\frac{1}{|\mathbf{R}_I-\mathbf{R}_J|}}_{V_{II}}$$

where,

- K_e: Electronic kinetic energy,
- K_I: Ionic kinetic energy,
- V_{ee}: Electron-electron coulomb potential energy,
- V_{eI}: Electron-ion coulomb potential energy,
- V_{II}: Ion-ion coulomb potential energy,
- The sum i and j goes over the number of electrons,
- The sum I and J goes over the number of ions.

Instead of using SI units, we use Hartree units: $e = m_e = \hbar = 4\pi\epsilon_0 = 1$. This way we don't need to worry about the constants.

$$\mathbb{H} = -\sum_i \frac{1}{2}\nabla_i^2 - \sum_I \frac{1}{2m_I}\nabla_I^2 + \frac{1}{2}\sum_{i\neq j}\frac{1}{|\mathbf{r}_i-\mathbf{r}_j|} + \sum_{i,I}\frac{Z_I}{|\mathbf{r}_i-\mathbf{R}_I|} + \frac{1}{2}\sum_{I\neq J}\frac{Z_I Z_J}{|\mathbf{R}_I-\mathbf{R}_J|}$$

Since ions are larger than electrons, we use Born-Oppenheimer approximation where we assume the ions are fixed and we just solve for the electronic degrees of freedom. We have

$$\mathbb{H} = -\sum_i \frac{1}{2}\nabla_i^2 + \frac{1}{2}\sum_{i\neq j}\frac{1}{|\mathbf{r}_i-\mathbf{r}_j|} + \sum_{i,I}\frac{Z_I}{|\mathbf{r}_i-\mathbf{R}_I|} + E_{II}$$

In this case,

- $\nabla_I^2 \approx 0$: assume ions are moving very slowing (compared to the electrons)
- $E_{II} = \frac{1}{2}\sum_{I \neq J} Z_I Z_J / |\mathbf{R}_I - \mathbf{R}_J|$: the ion-ion coulomb potential energy do not have any electronic degree of freedom, so it becomes a constant.

The goal here is to find the (ground state) eigenvalue and eigenfunction of H

$$H\,|\Psi(\mathbf{r}_1, \ldots, \mathbf{r}_N)\rangle = E\,|\Psi(\mathbf{r}_1, \ldots, \mathbf{r}_N)\rangle \tag{1}$$

1.1.2 Wavefunction representation for first quantization

Suppose we have two electrons with wavefunction ψ_a and ψ_b, with the first electron in ψ_a and the second electron in ψ_b. The two-electron wavefunction can be described as the *Hartree product* of the two single-electron wavefunctions:

$$\psi_a(\mathbf{r}_1) \otimes \psi_b(\mathbf{r}_2)$$

Due to the superposition principle, the two-electron wavefunction can be a superposition of Hartree product states

$$\Psi(\mathbf{r}_1, \mathbf{r}_2) = c_1\psi_a(\mathbf{r}_1) \otimes \psi_b(\mathbf{r}_2) + c_2\psi_c(\mathbf{r}_1) \otimes \psi_d(\mathbf{r}_2) + \cdots$$

As a result, given M orthonormal single-electron wavefunctions $\psi_1, \ldots, \psi_M$ (we call them orbitals) and N electrons, we have

$$\Psi(\mathbf{r}_1, \mathbf{r}_2 \ldots \mathbf{r}_N) = \sum_{i_1, i_2, \ldots i_N = 1}^{M} c_{i_1, i_2, \ldots, i_N} \psi_{i_1}(\mathbf{r}_1) \otimes \psi_{i_2}(\mathbf{r}_2) \otimes \cdots \otimes \psi_{i_N}(\mathbf{r}_M) \tag{2}$$

As an example, say $M = 4$ and $N = 2$, we have (pay attention to the subscript below ψ)

$$\begin{aligned}
\Psi(\mathbf{r}_1, \mathbf{r}_2) = {} & c_{11}\psi_1(\mathbf{r}_1) \otimes \psi_1(\mathbf{r}_2) + c_{12}\psi_1(\mathbf{r}_1) \otimes \psi_2(\mathbf{r}_2) + c_{13}\psi_1(\mathbf{r}_1) \otimes \psi_3(\mathbf{r}_2) + c_{14}\psi_1(\mathbf{r}_1) \otimes \psi_4(\mathbf{r}_2) \\
& + c_{21}\psi_2(\mathbf{r}_1) \otimes \psi_1(\mathbf{r}_2) + c_{22}\psi_2(\mathbf{r}_1) \otimes \psi_2(\mathbf{r}_2) + c_{23}\psi_2(\mathbf{r}_1) \otimes \psi_3(\mathbf{r}_2) + c_{24}\psi_2(\mathbf{r}_1) \otimes \psi_4(\mathbf{r}_2) \\
& + c_{31}\psi_3(\mathbf{r}_1) \otimes \psi_1(\mathbf{r}_2) + c_{32}\psi_3(\mathbf{r}_1) \otimes \psi_2(\mathbf{r}_2) + c_{33}\psi_3(\mathbf{r}_1) \otimes \psi_3(\mathbf{r}_2) + c_{34}\psi_3(\mathbf{r}_1) \otimes \psi_4(\mathbf{r}_2) \\
& + c_{41}\psi_4(\mathbf{r}_1) \otimes \psi_1(\mathbf{r}_2) + c_{42}\psi_4(\mathbf{r}_1) \otimes \psi_2(\mathbf{r}_2) + c_{43}\psi_4(\mathbf{r}_1) \otimes \psi_3(\mathbf{r}_2) + c_{44}\psi_4(\mathbf{r}_1) \otimes \psi_4(\mathbf{r}_2)
\end{aligned} \tag{3}$$

In principle, we can just replace $\Psi(\mathbf{r}_1, \ldots, \mathbf{r}_N)$ of Equation 1 with Equation 2 and solve for the ground state energy with respect to the coefficient $c_{11}, \ldots, c_{44}$. Given M orbitals and N electrons, we have N^M coefficients. There are properties of the system that we could use to reduce the number of coefficients for the same M, N values.

Electrons of the many-electron wavefunctions are indistinguishable from each other. In other words, the probablity density of the wavefunction should remain the same when electrons are swapped to different orbitals.

$$|\Psi(\mathbf{r}_1, \mathbf{r}_2)|^2 = |\Psi(\mathbf{r}_2, \mathbf{r}_1)|^2 \implies \Psi(\mathbf{r}_1, \mathbf{r}_2) = \pm\Psi(\mathbf{r}_2, \mathbf{r}_1) \tag{4}$$

The sign in Equation 4 is $+$ for bosons and $-$ for fermions. Since electrons are fermions, we use $-$. Substituting Equation 4 with Equation 3, we have for $a, b = 1, 2, 3, 4$,

$$\begin{aligned}
c_{ab} &= -c_{ba} \\
c_{aa} &= 0
\end{aligned}$$

As a result, we can write Equation 3 as

$$\begin{aligned}
\Psi(\mathbf{r}_1, \mathbf{r}_2) = {} & c_{12}[\psi_1(\mathbf{r}_1) \otimes \psi_2(\mathbf{r}_2) - \psi_2(\mathbf{r}_1) \otimes \psi_1(\mathbf{r}_2)] + c_{13}[\psi_1(\mathbf{r}_1) \otimes \psi_3(\mathbf{r}_2) - \psi_3(\mathbf{r}_1) \otimes \psi_1(\mathbf{r}_2)] \\
& + c_{14}[\psi_1(\mathbf{r}_1) \otimes \psi_4(\mathbf{r}_2) - \psi_4(\mathbf{r}_1) \otimes \psi_1(\mathbf{r}_2)] + c_{23}[\psi_2(\mathbf{r}_1) \otimes \psi_3(\mathbf{r}_2) - \psi_3(\mathbf{r}_1) \otimes \psi_2(\mathbf{r}_2)] \\
& + c_{24}[\psi_2(\mathbf{r}_1) \otimes \psi_4(\mathbf{r}_2) - \psi_4(\mathbf{r}_1) \otimes \psi_2(\mathbf{r}_2)] + c_{34}[\psi_3(\mathbf{r}_1) \otimes \psi_4(\mathbf{r}_2) - \psi_4(\mathbf{r}_1) \otimes \psi_3(\mathbf{r}_2)]
\end{aligned}$$

We can express the term in square brackets as the determinant of a 2×2 matrix called the Slater determinant.

$$\begin{aligned}\Psi(\mathbf{r}_1, \mathbf{r}_2) = c_{12} &\begin{vmatrix}\psi_1(\mathbf{r}_1) & \psi_2(\mathbf{r}_1)\\ \psi_1(\mathbf{r}_2) & \psi_2(\mathbf{r}_2)\end{vmatrix} + c_{13}\begin{vmatrix}\psi_1(\mathbf{r}_1) & \psi_3(\mathbf{r}_1)\\ \psi_1(\mathbf{r}_2) & \psi_3(\mathbf{r}_2)\end{vmatrix}\\ + c_{14} &\begin{vmatrix}\psi_1(\mathbf{r}_1) & \psi_4(\mathbf{r}_1)\\ \psi_1(\mathbf{r}_2) & \psi_4(\mathbf{r}_2)\end{vmatrix} + c_{23}\begin{vmatrix}\psi_2(\mathbf{r}_1) & \psi_3(\mathbf{r}_1)\\ \psi_2(\mathbf{r}_2) & \psi_3(\mathbf{r}_2)\end{vmatrix}\\ + c_{24} &\begin{vmatrix}\psi_2(\mathbf{r}_1) & \psi_4(\mathbf{r}_1)\\ \psi_2(\mathbf{r}_2) & \psi_4(\mathbf{r}_2)\end{vmatrix} + c_{34}\begin{vmatrix}\psi_3(\mathbf{r}_1) & \psi_4(\mathbf{r}_1)\\ \psi_3(\mathbf{r}_2) & \psi_4(\mathbf{r}_2)\end{vmatrix}\\ = c_{12}&|\psi_1\psi_2| + c_{13}|\psi_1\psi_3| + c_{14}|\psi_1\psi_4|\\ + c_{23}&|\psi_2\psi_3| + c_{24}|\psi_2\psi_4| + c_{34}|\psi_3\psi_4|\end{aligned} \tag{5}$$

We will use the notation $|\psi_a\psi_b|$ to indicate the Slater determinant.

$$|\psi_a\psi_b| = \begin{vmatrix}\psi_a(\mathbf{r}_1) & \psi_b(\mathbf{r}_1)\\ \psi_a(\mathbf{r}_2) & \psi_b(\mathbf{r}_2)\end{vmatrix} \tag{6}$$

The Slater determinant can also be naturally extended for systems with more electrons

$$|\psi_a\psi_b\psi_c\cdots| = \begin{vmatrix}\psi_a(\mathbf{r}_1) & \psi_b(\mathbf{r}_1) & \psi_c(\mathbf{r}_1) & \cdots\\ \psi_a(\mathbf{r}_2) & \psi_b(\mathbf{r}_2) & \psi_c(\mathbf{r}_2) & \cdots\\ \psi_a(\mathbf{r}_3) & \psi_b(\mathbf{r}_3) & \psi_c(\mathbf{r}_3) & \cdots\\ \vdots & \vdots & \vdots & \ddots\end{vmatrix}$$

1.1.3 Second Quantization

The motivation for second quantization is that the calculation of Slater determinants is complicated. We use a different notation for the Slater determinants. To write the Slater determinant in second quantizatino, We arrange the M orbitals into a binary string. Occupied orbitals are denoted as 1 while unoccupied orbitals is denoted as 0. As an example, we can write Equation 5 as

$$|\Psi\rangle = c_{12}\,|1100\rangle + c_{13}\,|1010\rangle + c_{14}\,|1001\rangle + c_{23}\,|0110\rangle + c_{24}\,|0101\rangle + c_{34}\,|0011\rangle$$

On the other hand, the Hamiltonian can be written as (more details on *Modern quantum chemistry: introduction to advanced electronic structure theory* by Szabo and Ostlund)

$$\mathbb{H} = \sum_{pq}^{M} h_{pq}c_p^\dagger c_q + \frac{1}{2}\sum_{pqrs}^{M} h_{pqrs}c_p^\dagger c_q^\dagger c_r c_s + E_{II} \tag{7}$$

We start with the integral coefficients h_{pq} and h_{pqrs}

$$\begin{aligned}h_{pq} &= \int \psi_p^*(\mathbf{x})\left[-\frac{1}{2}\nabla^2 + \sum_I \frac{Z_I}{|\mathbf{x}-\mathbf{R}_I|}\right]\psi_q(\mathbf{x})\mathrm{d}\mathbf{x} = \int \psi_p^*(\mathbf{x})h_1(\mathbf{x})\psi_q(\mathbf{x})\mathrm{d}\mathbf{x}\\ h_{pqrs} &= \iint \frac{\psi_p^*(\mathbf{x}_1)\psi_q^*(\mathbf{x}_2)\psi_r(\mathbf{x}_2)\psi_s(\mathbf{x}_1)}{|\mathbf{x}_1-\mathbf{x}_2|}\mathrm{d}\mathbf{x}_1\mathrm{d}\mathbf{x}_2\end{aligned}$$

Note 2. In other books, h_{pqrs} has two additional notations. $(ps|qr)$ is the chemist's notation, while $[pq|sr]$ is the physicist's notation. Note the difference in the index orders.

$$\begin{aligned}h_{pqrs} &= \iint \frac{\psi_p^*(\mathbf{x}_1)\psi_q^*(\mathbf{x}_2)\psi_r(\mathbf{x}_2)\psi_s(\mathbf{x}_1)}{|\mathbf{x}_1-\mathbf{x}_2|}\mathrm{d}\mathbf{x}_1\mathrm{d}\mathbf{x}_2\\ (ps|qr) &= \iint \frac{\psi_p^*(\mathbf{x}_1)\psi_s(\mathbf{x}_1)\psi_q^*(\mathbf{x}_2)\psi_r(\mathbf{x}_2)}{|\mathbf{x}_1-\mathbf{x}_2|}\mathrm{d}\mathbf{x}_1\mathrm{d}\mathbf{x}_2\\ [pq|sr] &= \iint \frac{\psi_p^*(\mathbf{x}_1)\psi_q^*(\mathbf{x}_2)\psi_s(\mathbf{x}_1)\psi_r(\mathbf{x}_2)}{|\mathbf{x}_1-\mathbf{x}_2|}\mathrm{d}\mathbf{x}_1\mathrm{d}\mathbf{x}_2\end{aligned}$$

Note 3. Most programs do not include E_{II} in the Hamiltonian. Most of the time, once the ground state energy is calculated, we add the E_{II} to the calculated ground state energy.

Next, we look into the creation $c_p^\dagger$ and annihiliation c_p operators. The creation operator $c_p^\dagger$ creates an electron at orbital p, while the annihiliation operator c_p removes an electron at orbital p.

$$c_p \left|f_1 f_2 \cdots f_{p-1} 0 f_{p+1} \cdots f_{M-1} f_M\right\rangle = 0 \tag{8}$$

$$c_p \left|f_1 f_2 \cdots f_{p-1} 1 f_{p+1} \cdots f_{M-1} f_M\right\rangle = (-1)^{\sum_{i=1}^{p-1} f_i} \left|f_1 f_2 \cdots f_{p-1} 0 f_{p+1} \cdots f_{M-1} f_M\right\rangle \tag{9}$$

$$c_p^\dagger \left|f_1 f_2 \cdots f_{p-1} 0 f_{p+1} \cdots f_{M-1} f_M\right\rangle = (-1)^{\sum_{i=1}^{p-1} f_i} \left|f_1 f_2 \cdots f_{p-1} 1 f_{p+1} \cdots f_{M-1} f_M\right\rangle \tag{10}$$

$$c_p^\dagger \left|f_1 f_2 \cdots f_{p-1} 1 f_{p+1} \cdots f_{M-1} f_M\right\rangle = 0 \tag{11}$$

We also have some anticommutation relations ($\{A, B\} = AB + BA$)

$$\{c_p, c_q\} = 0$$
$$\{c_p^\dagger, c_q^\dagger\} = 0$$
$$\{c_p^\dagger, c_q\} = \delta_{pq}$$

Note 4. The M orbitals I am referring to here includes the spin degrees of freedom. So $M/2$ orbitals will be spin up and the other $M/2$ is spin down. For the second quantization notation, it is commonly arranged as

$$\left|\underbrace{f_1 \cdots f_{M/2}}_{\uparrow} \underbrace{f_{M/2+1} \cdots f_M}_{\downarrow}\right\rangle$$

The orbital which includes the spin here is called "spin orbitals". On the other hand, we call the $M/2$ orbitals without the spin dependency "spatial orbitals". For closed shell systems (spin up wfc = spin down wfc), if we use spatial orbitals for the $pqrs$ index, the Hamiltonian becomes

$$\begin{aligned}\mathbb{H} = &\sum_{pq}^{M/2} h_{pq}[c_{p\uparrow}^\dagger c_{q\uparrow} + c_{p\downarrow}^\dagger c_{q\downarrow}] \\ &+ \frac{1}{2}\sum_{pqrs}^{M/2} h_{pqrs}[c_{p\uparrow}^\dagger c_{q\uparrow}^\dagger c_{r\uparrow} c_{s\uparrow} + c_{p\uparrow}^\dagger c_{q\downarrow}^\dagger c_{r\downarrow} c_{s\uparrow} + c_{p\downarrow}^\dagger c_{q\uparrow}^\dagger c_{r\uparrow} c_{s\downarrow} + c_{p\downarrow}^\dagger c_{q\downarrow}^\dagger c_{r\downarrow} c_{s\downarrow}] \\ &+ E_{II}\end{aligned}$$

1.2 Obtaining Orbitals Wavefunctions

This section is about the choise of the orbital wavefunction ψ_p that we are going to use. The choice used by PySCF is the Gaussian-Type orbitals (also called linear combinations of atomic orbitals). They are modelled after the solution of the Hydrogen atom with some modifications to simplify the calculations.

Note 5. The atomic orbitals ϕ are usually of the form (differs from basis set to basis set and even from program to program)

$$\phi_{nlm}(\mathbf{x}) = e^{-\alpha_n |\mathbf{x}|^2} Y_{lm}(\mathbf{x})$$

Y_{lm} is the spherical harmonics. The exact values for α, n, l, m differs from basis set to basis set.

In principle, we could directly use the atomic orbitals as a basis for VQE. However, it is not usually a good starting point. Instead, we first do an HF/DFT calculation (which are computationally cheaper but less accurate) and use the result as a basis for the VQE calculation.

1.2.1 Hartree-Fock calculation

Unlike many-body calculation (like VQE) where we deal with multiple Slater determinants (see Equation 5), the HF calculation attempts to get the ground state energy using only one Slater determinant. While this reduces the computational complexity of the calculation, this also reduces the accuracy. Nevertheless, the result of HF calculations gives a good initial guess for more accurate many-body calculations. Given M atomic orbitals and N electrons, we parametrize the orbitals as linear combinations of atomic orbitals,

$$\psi_i = \sum_{j=1}^{M} C_{ji}\phi_j \tag{12}$$

The slater determinant of $\psi_1, \ldots, \psi_N$ is

$$\begin{aligned} |\psi_1, \ldots, \psi_N| &= \begin{vmatrix} \psi_1(\mathbf{r}_1) & \cdots & \psi_N(\mathbf{r}_1) \\ \vdots & \ddots & \vdots \\ \psi_1(\mathbf{r}_N) & \cdots & \psi_N(\mathbf{r}_N) \end{vmatrix} \\ &= \begin{vmatrix} \sum_{j=1}^{M} C_{j1}\phi_j(\mathbf{r}_1) & \cdots & \sum_{j=1}^{M} C_{jN}\phi_j(\mathbf{r}_1) \\ \vdots & \ddots & \vdots \\ \sum_{j=1}^{M} C_{j1}\phi_j(\mathbf{r}_N) & \cdots & \sum_{j=1}^{M} C_{jN}\phi_j(\mathbf{r}_N) \end{vmatrix} \end{aligned}$$

The HF algorithm minimizes the Hamiltonian expectation value with respect to the above Slater determinant by changing the C_{ji} (see Equation 6)

$$E_{HF} = \min_{C_{ji}} \langle |\psi_1 \cdots \psi_N||H||\psi_1 \cdots \psi_N| \rangle \tag{13}$$

If you follow *Modern quantum chemistry: introduction to advanced electronic structure theory* by Szabo and Ostlund, you can get the following eigenvalue equations

$$\mathbf{FC} = \varepsilon \mathbf{SC} \tag{14}$$

where,

- ε is a diagonal $M \times M$ matrix containing the orbital energies. (This is not related to the HF energies and you cannot obtain the HF energy by just summing the orbital energies)
- $\mathbf{C}$ is an $M \times M$ matrix, the j^{th} row and i^{th} column of $\mathbf{C}$ is the coefficient C_{ji} in Equation 12.
- The overlap matrix $\mathbf{S}$ is an $M \times M$, matrix, with entries

$$\mathbf{S}_{ij} = \int \phi_i^*(\mathbf{x})\phi_j(\mathbf{x})\mathrm{d}\mathbf{x}$$

- The Fock matrix $\mathbf{F}$ is an $M \times M$ matrix, with entries

$$\begin{aligned} \mathbf{F}_{ij} &= \int \phi_i^*(\mathbf{x}) \left[h(\mathbf{x}) + J(\mathbf{x}) - K(\mathbf{x})\right] \phi_j(\mathbf{x})\, \mathrm{d}\mathbf{x} \\ h(\mathbf{x}) &= -\frac{1}{2}\nabla^2 + \sum_I \frac{Z_I}{|\mathbf{x} - \mathbf{R}_I|} \\ J(\mathbf{x}) &= \sum_{k=1}^{N} \int \frac{\psi_k^{(old)*}(\mathbf{x}')\psi_k^{(old)}}{|\mathbf{x} - \mathbf{x}'|}\, \mathrm{d}\mathbf{x}' \\ K(\mathbf{x})\phi_j(\mathbf{x}) &= \sum_{k=1}^{N} \psi_k^{(old)}(\mathbf{x}) \int \frac{\psi_k^{(old)*}(\mathbf{x}')\phi_j(\mathbf{x}')}{|\mathbf{x} - \mathbf{x}'|}\mathrm{d}\mathbf{x}' \end{aligned}$$

Note 6. If you look at the Fock matrix, you will see $\psi^{(old)}$ as input. This is because HF is an iterative algorithm.

- Start with an initial guess of C_{ji} to get a set of orbitals ψ.
- Use the initial set of orbitals as $\psi^{(old)}$ to calculate $\mathbf{F}$.
- Solve Equation 14 to get a new set of C_{ji}
- Use that new c_{ji} to get a new set of orbitals ψ
- Repeat until C_{ji} convergences

Once we have the converged C_{ji}, we use Equation 13 to get E_{HF}.

Note 7. While we have N electrons, we have M eigenvalues of the Fock matrix. This is a beneficial for the VQE calculation as we will be using both the N occupied orbitals along with the additional $M - N$ unoccupied orbitals (also called virtual orbitals).

Note 8. DFT calculation replaces J and K with an easier to calculate term E_{XC} called exchange correlation (XC) functional. The algorithm remains mostly the same. There are many different types of functional with each giving different results (eg. PBE, BLYP, etc)

1.3 Obtaining Integral Coefficients

Once we have C_{ji} from the HF/DFT calculation, we can use the orbitals of the HF calculation (ψ of Equation 12) as the basis for the VQE calculation. We can calculate the h_{pq} and h_{pqrs} by substituting ψ with the Equation 12.

$$\begin{aligned}
h_{pq} &= \int \psi_p^*(\mathbf{x}) h_1(\mathbf{x}) \psi_q(\mathbf{x})\, \mathrm{d}\mathbf{x} \\
&= \sum_{ab}^{M} \int \left[C_{ap}\phi_a(\mathbf{x})\right]^* h_1(\mathbf{x}) \left[C_{bq}\phi_b(\mathbf{x})\right] \mathrm{d}\mathbf{x} \\
&= \sum_{ab}^{M} C_{ap}^* C_{bq} \int \phi_a^*(\mathbf{x}) h_1(\mathbf{x}) \phi_b(\mathbf{x})\, \mathrm{d}\mathbf{x} \\
&= \sum_{ab}^{M} C_{ap}^* C_{bq} h_{ab}^{(atomic)}
\end{aligned}$$

This is done similarly for h_{pqrs} so I just skip the details

$$\begin{aligned}
h_{pqrs} &= \sum_{abcd}^{M} C_{ap}^* C_{bq}^* C_{cr} C_{ds} \iint \frac{\phi_a^*(\mathbf{x}_1)\phi_b^*(\mathbf{x}_2)\phi_c(\mathbf{x}_2)\phi_d(\mathbf{x}_1)}{|\mathbf{x}_1 - \mathbf{x}_2|}\, \mathrm{d}\mathbf{x}_1\, \mathrm{d}\mathbf{x}_2 \\
&= \sum_{abcd}^{M} C_{ap}^* C_{bq}^* C_{cr} C_{ds} h_{abcd}^{(atomic)}
\end{aligned}$$

We do this for h_{pq} and h_{pqrs} as PySCF can calculate $h_{ab}^{(atomic)}$ and $h_{abcd}^{(atomic)}$ for us. Below is an example for H_2.

```
import numpy as np
from pyscf import ao2mo, gto, scf

def create_integrals_h2_sto3g():
    # create the H2 molecule
    mol = gto.M(
        atom=[
            ("H", [0, 0, 0]),
            ("H", [0, 0, 0.735]),
        ],
        unit="Angstrom",
        basis="sto-3g",
```

```
    )

    # do the HF calculation
    hf = scf.RHF(mol)
    hf.kernel()

    # number of electrons (N)
    nelec = mol.nelectron
    # number of orbitals (M)
    norb = hf.mo_coeff.shape[-1]
    # orbital coefficients (C_ji)
    c = hf.mo_coeff
    print(f"nOrb : {norb}")
    print(f"nElec: {nelec}")

    # nuclear repulsion energy E_II
    nuclear = mol.energy_nuc()

    # h^(atom)_ab
    h1atom = hf.get_hcore()
    # hpq = sum_ab C_ap^* C_bq h^(atom)_ab
    # since c is all real (no complex number), we don't need to conjugate
    hpq = np.einsum("ap,bq,ab->pq", c, c, h1atom)

    use_slow_version = False
    if use_slow_version:
        # SLOW VERSION OF hpqrs
        # hpqrs = sum_abcd C_ap^* C_bq^* C_cr C_ds h^(atom)_abcd
        # PYSCF USES CHEMISTS NOTATION, SO WE NEED TO DO
        # (ps|qr) = sum_adbc C_ap^* C_bq^* C_cr C_ds (ad|bc)^((atom))
        # since c is all real (no complex number), we don't need to conjugate
        h2atom = mol.intor("int2e")
        hpqrs = np.einsum("ap,bq,cr,ds,adbc->pqrs", c, c, c, c, h2atom)
    else:
        # FASTER VERSION OF hpqrs
        # PYSCF USES CHEMISTS NOTATION, NEED TO CHANGE THE ORDER
        # (ps|qr) -> h_pqrs
        hpqrs = ao2mo.full(mol, c, compact=False).reshape(norb, norb, norb, norb)
        hpqrs = hpqrs.transpose(0, 2, 3, 1)
    return hpq, hpqrs, nuclear, nelec
```

2 Encoding

Further reading: The Variational Quantum Eigensolver: A review of methods and best practices Section 4.1

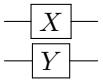
The goal here is to convert the second quantization Hamiltonian into a quantum circuit.

2.1 Pauli operators

One of the basic building blocks of quantum circuits is the Pauli operator, we have the X, Y, and Z gates

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$
$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$
$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Pauli operator can be applied to multiple qubits by applying the Kronecker product (more commonly called tensor product) on the two gates.



$$\implies X \otimes Y = \begin{pmatrix} 0 \times \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} & 1 \times \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \\ 1 \times \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} & 0 \times \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & -i \\ 0 & 0 & i & 0 \\ 0 & -i & 0 & 0 \\ i & 0 & 0 & 0 \end{pmatrix}$$

2.2 Jordan Wigner Encoding

There are multiple ways of converting Hamiltonian to a circuit. This conversion is called encoding. The simplest here is the Jordan-Wigner encoding. The M orbitals are mapped into M qubits, where the occupation of the orbitals is represented by the occupation of each qubit.

$$|f_1 f_2 \cdots f_M\rangle_{\text{fermion}} = |f_1 f_2 \cdots f_M\rangle_{\text{qubits}}$$

Note 9. For example: the state $|0101\rangle$ will have 1 on qubit 2 and 4 and 0 on qubit 1 and 3.

Converting the operators into qubits is more challenging

$$c_p^\dagger = \frac{(X_p - iY_p)}{2} \otimes Z_{p-1} \otimes \cdots \otimes Z_1$$
$$c_p = \frac{(X_p + iY_p)}{2} \otimes Z_{p-1} \otimes \cdots \otimes Z_1$$

The $(X \pm iY)/2$ handles the qubit flipping part, while the Z gates handles the sign changes part (the $(-1)^{\sum_{i=1}^{p} f_i}$ part of Equation 8-11).

Code below shows how to convert the Hamiltonian to a Quantum Circuit (Note that we have spin up and spin down orbitals)

```
from qiskit_nature.second_q.mappers import JordanWignerMapper
from qiskit_nature.second_q.operators import FermionicOp

def create_hamiltonian_quantum_circuit(hpq, hpqrs, mapper):
    # NOTE: we exclude the nuclear term here.
    # We just add the nuclear term to the VQE energy afterwards
    n_spatial_orb = hpq.shape[0]
    n_spin_orb = 2 * n_spatial_orb

    fermionic_op_dict = {}
    for spin in range(2):
        # 0 is UP 1 is DW
        for p_spatial_orb in range(n_spatial_orb):
            for q_spatial_orb in range(n_spatial_orb):
                p_spin_orb = spin * n_spatial_orb + p_spatial_orb
                q_spin_orb = spin * n_spatial_orb + q_spatial_orb

                p_qubit = p_spin_orb
                q_qubit = q_spin_orb
                coeff = hpq[p_spatial_orb, q_spatial_orb]
                fermionic_op_dict[f"+_{p_qubit} -_{q_qubit}"] = coeff
    for spin_ps in range(2):
        for spin_qr in range(2):
            for p_spatial_orb in range(n_spatial_orb):
                for q_spatial_orb in range(n_spatial_orb):
                    for r_spatial_orb in range(n_spatial_orb):
                        for s_spatial_orb in range(n_spatial_orb):
                            p_spin_orb = spin_ps * n_spatial_orb + p_spatial_orb
                            q_spin_orb = spin_qr * n_spatial_orb + q_spatial_orb
                            r_spin_orb = spin_qr * n_spatial_orb + r_spatial_orb
                            s_spin_orb = spin_ps * n_spatial_orb + s_spatial_orb

                            p_qubit = p_spin_orb
                            q_qubit = q_spin_orb
                            r_qubit = r_spin_orb
                            s_qubit = s_spin_orb
                            coeff = hpqrs[
                                p_spatial_orb,
                                q_spatial_orb,
                                r_spatial_orb,
                                s_spatial_orb,
                            ]
                            # multiply by 0.5
                            fermionic_op_dict[
                                f"+_{p_qubit} +_{q_qubit} -_{r_qubit} -_{s_qubit}"
                            ] = (coeff * 0.5)
    fermionic_op = FermionicOp(fermionic_op_dict, num_spin_orbitals=n_spin_orb)
```

```
    qubit_op = mapper.map(fermionic_op)
    return qubit_op
```

3 Ansatz Selection and Preparation

Further reading: The Variational Quantum Eigensolver: A review of methods and best practices Section 6.2

Here, we will discuss two ansatz: The chemically inspired UCC ansatz, and the hardware efficient ansatz.

3.1 UCC ansatz

The ansatz can be written in the form

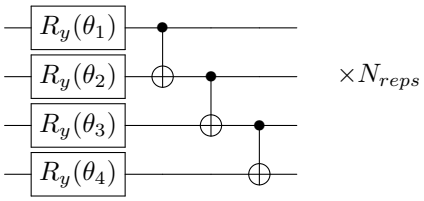
$$
\begin{aligned}
|\Psi(\theta)\rangle &= e^{\hat{T}(\theta)} |\Psi_{HF}\rangle \\
\hat{T} &= \hat{T}_1 + \hat{T}_2 + \cdots \\
\hat{T}_1 &= \sum_{p>q} \theta_{pq} (c_p^\dagger c_q - c_q^\dagger c_p) \\
\hat{T}_2 &= \sum_{p>q>r>s} \theta_{pqrs} (c_p^\dagger c_q^\dagger c_r c_s - c_s^\dagger c_r^\dagger c_q c_p)
\end{aligned}
$$

This ansatz usually requires less parameters, but have more circuit depth. You can turn on the `print_circuit` option and see the circuits required for the H_2 system.

```
from qiskit.circuit import QuantumCircuit
from qiskit_nature.second_q.circuit.library import UCC


def make_ucc_ansatz(n_spatial_orb, n_elec, mapper, print_circuit=False):
    n_spin_orb = n_spatial_orb * 2
    n_elec_per_spin = n_elec // 2

    # first create the HF state
    n_qubits = n_spin_orb
    hf = QuantumCircuit(n_qubits)
    for spin in range(2):
        # loop over occupied spatial orbitals
        for p_spatial_orb in range(n_elec_per_spin):
            p_spin_orb = spin * n_spatial_orb + p_spatial_orb
            hf.x(p_spin_orb)

    # num_particles: we need to separate number of spin up and spin down electrons
    # excitations: sd -> singles + doubles
    # qubit_mapper: The Jordan Wigner mapper
    # initial state is HF
    ucc = UCC(
        num_spatial_orbitals=n_spatial_orb,
        num_particles=(n_elec_per_spin, n_elec_per_spin),
        excitations="sd",
        qubit_mapper=mapper,
        initial_state=hf,
    )

    if print_circuit:
        ucc_draw = ucc.decompose().decompose().decompose()
        print(ucc_draw.draw("text"))
    return ucc
```

3.2 HEA ansatz

There is no standard form for the ansatz, but I use the following



For the code below, I use $N_{reps} = 2$. If you turn on `print_circuit` option, you can see that the circuit is much simpler compared with the UCC ansatz.

```
from qiskit.circuit import ParameterVector, QuantumCircuit


def make_hea_ansatz(n_spatial_orb, reps=2, print_circuit=False):
    n_spin_orb = n_spatial_orb * 2
    n_qubits = n_spin_orb

    n_params = n_qubits * reps
    params = ParameterVector("t", n_params)
    ansatz = QuantumCircuit(n_qubits)

    iparam = 0
    for irep in range(reps):
        for p_spin_orb in range(n_spin_orb):
            ansatz.ry(params[iparam], p_spin_orb)
            iparam += 1
        ansatz.barrier()
        for p_spin_orb in range(n_spin_orb - 1):
            ansatz.cx(p_spin_orb, p_spin_orb + 1)
        ansatz.barrier()

    if print_circuit:
        ansatz_draw = ansatz.decompose()
        print(ansatz_draw.draw("text"))
    return ansatz
```

4 VQE Algorithm

Here we use `scipy.optimize.minimize` to find the minimum energy

```
import numpy as np
from qiskit_aer.primitives import Estimator as AerEstimator
from scipy.optimize import minimize


def do_vqe(ansatz, hamiltonian, nuclear):
    nparams = ansatz.num_parameters

    # theta: set the inital theta to random
    initial_state = np.random.random(nparams)

    # noiseless simulator
    estimator = AerEstimator(run_options={"shots": None}, approximation=True)

    index = 0

    def get_energy(parameter_values):
        nonlocal index
        result = estimator.run(
            circuits=ansatz, observables=hamiltonian, parameter_values=parameter_values
        ).result()
        energy = result.values[0] + nuclear
        index += 1
        parameter_str = " ".join(f"{e:.4f}" for e in parameter_values)
        print(f"#{index:5d}: {energy:.6f}: {parameter_str}")
```

```
        return energy

    res = minimize(fun=get_energy, x0=initial_state, method="bfgs")
    print(f"Energy: {res.fun}")
```

Running the code

```
from qiskit_nature.second_q.mappers import JordanWignerMapper

print_circuit = False
hpq, hpqrs, nuclear, nelec = create_integrals_h2_sto3g()
mapper = JordanWignerMapper()
hamiltonian = create_hamiltonian_quantum_circuit(hpq, hpqrs, mapper)
ansatz = make_ucc_ansatz(hpq.shape[0], nelec, mapper, print_circuit=print_circuit)
# OR
# ansatz = make_hea_ansatz(hpq.shape[0], print_circuit=print_circuit)
do_vqe(ansatz, hamiltonian, nuclear)
```