Study of Adaptative Derivative-Assemble Peusdo-Trotter Ansatzes in VQE through qiskit API

Max Alteg, Baptiste Chevalier, Octave Mestoudjian,Johan-Luca Rossi

April 2022

Contents

Introduction

The solution of the time-independent Schrodinger equation for molecular systems allows for the prediction of chemical properties. Quantum Computers are expected to make possible the simulation of quantum systems more efficiently and accurately than classical ones, including the simulation of molecules. This is due to the exponential size of the Hilbert Space.

The first algorithm which has been proposed to solve the problem of simulating electronic structure on a Quantum Computer is well-known as the Phase Estimation Algorithm (PEA). Despite that PEA gives the exact ground state for a given problem, the long circuit and the complex gates used in this algorithm make it impractical on any current or near-term Quantum Hardware referred as NISQ (Noisy intermediate Scale Quantum) devices.

In order to answer the problem of NISQ era and allows one to outperform classical computers, Variational Quantum Algorithms (VQAs) were designed. VQAs are hybrid classical-quantum algorithms due to the fact that there are using a quantum hardware for computation while using a classical optimizer for optimization. In this way, VQA algorithms allows the creation of shallower circuits and are thus much more noise resilient. The VQA algorithm we are interested in is the so-called Variational Quantum Eigensolver (VQE) algorithm and was originally designed to simulate electronic structures and to compute the ground state of a given molecule.

VQE is made of two main components. First, an ansatz which has several tunable parameters and his associated quantum circuit. The ansatz is run on the quantum device and aim to simulate the wavefunction, the parameters of the ansatz will be optimize until the expectation value is minimum meaning that one has found the ground state. The second component is then the classical optimizer, it is run on a classical computer and is the often the same as the one used in Machine Learning. At this point, it is clear that if there is a limiting factor for VQE it can only be the ansatz.

The very first ansatz that has originally been used is called UCCSD and it is based on Coupled Cluster Theory truncated at single and double excitations. One other way that has been investigated alongside is the Hardware efficient ansatz (HEA) which was designed to be efficiently implemented on NISQ devices but is not as efficient as UCCSD.

The main issue considering UCCSD is the large amount of parameters to optimize and this leads us to the introduction of Adaptive Derivative-Assembled Pseudo-Trotter ansatz VQE (ADAPT-VQE) which determines a quasi-optimal ansatz with a minimal number of parameters.

Unlike ansatzes introduced above, the key point of ADAPT-VQE is to grow the ansatz at every step, by adding operators chosen from a pre-determined pool of operators one-at-a-time, assuring that the maximal amount of correlation energy is recovered at each step. There exists different kind of ADAPT-VQE depending on the starting pool of operators as the fermionic-ADAPT, the qubit-ADAPT or even the qubit excitation based (QEB).

The goal of this project is to implement the different types of ADAPT-VQE mentioned before. After being sure to understand the theoretical background under all of these concepts, we will implement each algorithm using quiskit. We will also compare all of these algorithms on different criterions such as the number of parameters, the accuracy or the number of CNOT gate used on H_2 and LiH molecules. Then we will have a small discussion about the results we obtained.

Chapter 1

Theoretical Background

1.1 Quantum Chemistry

1.1.1 Hamiltonian and Born-Oppenheimer approximation

Considering a molecule formed by N atoms of atomic number Z_k, located at the position R_k, the *ab initio molecular Hamiltonian* can be written as [1]:

$$\hat{\mathcal{H}} = \hat{T}_n + \hat{T}_e + \hat{V}_n + \hat{V}_{ne} + \hat{V}_{ee} \tag{1.1}$$

where $\hat{T}_n$ and $\hat{T}_e$ are respectively the kinetic energy of the nuclei and of the electrons, and $\hat{V}_n, \hat{V}_{ne}$ and $\hat{V}_{ee}$ are the electric potential in between the nuclei together, the nuclei and the electrons and the electrons together.

Regarding to the Born-Oppenheimer approximation, one can neglect the kinematic energy of the nuclei as well as his electric potential since it is much heavier than the electrons and his motion can then be neglected. Hence one get from equation 1.1 the Hamiltonian :

$$\hat{\mathcal{H}} = \hat{T}_e + \hat{V}_{ne} + \hat{V}_{ee} \tag{1.2}$$

where

$$\hat{T}_e = -\sum_i \frac{\hbar^2}{2m_e} \nabla_i^2 \tag{1.3}$$

$$\hat{V}_{ne} = -\sum_{i,k} \frac{e^2}{4\pi\epsilon_0} \frac{Z_k}{|r_i - R_k|} \tag{1.4}$$

$$\hat{V}_{ee} = \frac{1}{2} \sum_{i \neq j} \frac{e^2}{4\pi\epsilon_0} \frac{1}{|r_i - r_j|} \tag{1.5}$$

with r_i the position of the electron i, m_e the mass of the electron, e the elementary charge, $\hbar$ the reduced Plank constant and ∇_i the Laplace operator for electron i.

1.1.2 Spin-orbital and Slater determinant

Before continuing to develop the hamiltonian, we need a basis to represent the wavefunction into. Basis elements are called *spin-orbitals* and are noted $\phi_j(\xi_j)$, majority of VQE algorithms use the so-called Slater Type Orbitals of minimized size, for instance the minimal STO-3G basis.

As electrons are fermions with half integer spin, the wavefunction have to be *antisymmetric* meaning that $\psi(\xi_1, \ldots, \xi_k, \ldots, \xi_p, \ldots, \xi_N) = -\psi(\xi_1, \ldots, \xi_p, \ldots, \xi_k, \ldots, \xi_N)$, hence two electron can not be simultaneously in the same quantum state, this is what is called the *Pauli exclusion principle*. A relevant idea is then to represent the function as a determinant or a linear combination of determinants (because determinants respect the above properties). [2]

For a wavefunction of n occupied orbitals, one can write the *Slater determinant* representing the many-body basis function :

$$\Psi(\xi_1, \xi_2, \ldots, \xi_n) = \frac{1}{\sqrt{n!}} \begin{vmatrix} \phi_1(\xi_1) & \phi_2(\xi_1) & \cdots & \phi_n(\xi_1) \\ \phi_1(\xi_2) & \phi_2(\xi_2) & \cdots & \phi_n(\xi_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(\xi_n) & \phi_2(\xi_n) & \cdots & \phi_n(\xi_n) \end{vmatrix} \tag{1.6}$$

with the variables $\xi_j = (r_j, \sigma_j)$ representing both position and spin of the electrons. For practical reasons, we will denote the basis function has :

$$|\Psi\rangle = |\phi_1 \phi_2 \ldots \phi_n\rangle \tag{1.7}$$

where $\phi_j = 1$ means the j-th basis function $\phi_j(\xi_j)$ is occupied while $\phi_j = 0$ means that it is unoccupied.

The Slater determinant can be used as an **approximation of the wavefunction** as it is the case in the *Hartree-Fock method* 1.2.3. Note that $\langle \Psi_a | \Psi_b \rangle = \delta_{a,b}$ To be exact, one can also expand the wavefunction in the Hilbert space span by those basis functions as a **linear combination of those determinants**.

1.1.3 First and Second Quantization of the Hamiltonian

First Quantization

Defining the wavefunction in terms of Slater determinants ensure that the antisymmetry property if verified as for any permutation σ : $|\sigma(\phi_1 \phi_2 \ldots \phi_n)\rangle = (-1)^{\pi(\sigma)} |\phi_1 \phi_2 \ldots \phi_n\rangle$ As we know that this property is already respected, one can first quantize the hamiltonian

by projecting into the space spanned by the $\{\phi_i(\xi_i)\}$ and obtain the *one and two body integrals*.

$$h_{pq} = \langle \phi_p | \hat{T}_e + \hat{V}_{ne} | \phi_q \rangle = \int d\xi \phi_p^*(\xi)\Big(-\frac{\hbar^2}{2m_e}\nabla^2 - \sum_k \frac{e^2}{4\pi\epsilon_0}\frac{Z_k}{|r - R_k|}\Big)\phi_q(\xi) \tag{1.8}$$

$$h_{pqrs} = \langle \phi_p \phi_q | \hat{V}_{ee} | \phi_r \phi_s \rangle = \frac{e^2}{4\pi\epsilon_0}\int d\xi_1 d\xi_2 \frac{\phi_p^*(\xi_1)\phi_q^*(\xi_2)\phi_r(\xi_1)\phi_s(\xi_2)}{|r_1 - r_2|} \tag{1.9}$$

And thus the first quantized hamiltonian :

$$\hat{\mathcal{H}} = \sum_{i=1}^m \sum_{p,q=1}^n h_{pq} |\phi_p^{(i)}\rangle \langle\phi_q^{(i)}| + \frac{1}{2}\sum_{i\neq j}^m \sum_{p,q,r,s=1}^n h_{pqrs} |\phi_p^{(i)}\phi_q^{(j)}\rangle \langle\phi_r^{(i)}\phi_s^{(j)}| \tag{1.10}$$

The first quantization is not often used, even though the mapping is quite straightforward, it is not adapted to NISQ computers.

Second Quantization

For second quantization, we need the operators to ensure the antisymetry property. The operators should allow moving from one basis to another (which was obvious in the first quantization), these are the so-called *fermionic creation* $\hat{a}^\dagger$ and *annihilation* $\hat{a}$ operators. Those operators are subject to the algebra defined by following anti-commutation rules:

$$\{\hat{a}_p, \hat{a}_q^\dagger\} = \delta_{p,q} \tag{1.11}$$

$$\{\hat{a}_p^\dagger, \hat{a}_q^\dagger\} = \{\hat{a}_p, \hat{a}_q\} = 0 \tag{1.12}$$

Hence the Slater determinant can be re-written as :

$$|\Psi\rangle = \prod_i (\hat{a}_i^\dagger)^{\phi_i} |vac\rangle = (\hat{a}_1^\dagger)^{\phi_1}(\hat{a}_2^\dagger)^{\phi_2}\dots(\hat{a}_n^\dagger)^{\phi_n} |vac\rangle \tag{1.13}$$

and the *Second quantize hamiltonian* as

$$\hat{\mathcal{H}} = \sum_{pq} h_{pq}\hat{a}_p^\dagger\hat{a}_q + \frac{1}{2}\sum_{pqrs} h_{pqrs}\hat{a}_p^\dagger\hat{a}_q^\dagger\hat{a}_r\hat{a}_s \tag{1.14}$$

1.1.4   Mapping

There are several ways to map the fermionic operators to spin-operators and so to qubits which are spin-$1/2$ objects. Moreover, in general the only operators that can be measured on a Quantum Computer are spin-operators or Pauli operators X,Y,Z.

A map $\mathcal{T} : \mathcal{F}_n \rightarrow (\mathbb{C}^2)^{\otimes N}$ from the n orbitals Fock states space $\mathcal{F}_n$ to the operators acting on spin states of N qubits space $(\mathbb{C}^2)^{\otimes N}$ should preserve the antisymetric behaviour of

fermionic operators. Jordan and Wigner showed that such an isomorphism maintaining the algebraic structure does exist. The mapping concerns the hamiltonian operator used for computing the expectation value in the VQE as well as the ansatzes initially defined in fermionic terms (for instance the UCC).

The most used mapping is the *Jordan-Wigner mapping*, in this mapping one qubit corresponds to one n-spin-orbital. The value $|0\rangle$ for a qubit corresponds to the orbital being unoccupied while the value $|1\rangle$ corresponds to the orbital being occupied.

One wants to map the fermionic operators to pauli matrices. Intuitively, the mapping according to the operators should be the following :

$$\hat{a}_j^\dagger \rightarrow |1\rangle \langle 0|_j = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} = \frac{X_j - iY_j}{2} \tag{1.15}$$

$$\hat{a}_j \rightarrow |0\rangle \langle 1|_j = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} = \frac{X_j + iY_j}{2} \tag{1.16}$$

where X_j and Y_j act on the j-th qubit. But the antisymetry relation is not respected anymore. To do so, we need to add a Z operators string of length j such that the eigenvalue will be $+1$ for even number of occupied orbitals and -1 for odd number of occupied orbitals. The final mapping is then :

$$\hat{a}_j^\dagger \rightarrow \frac{X_j - iY_j}{2} \bigotimes_{k=0}^{j-1} Z_k \tag{1.17}$$

$$\hat{a}_j \rightarrow \frac{X_j + iY_j}{2} \bigotimes_{k=0}^{j-1} Z_k \tag{1.18}$$

The number of pauli string generated in this mapping scale in $\mathcal{O}(N^4)$ where N is the number of qubits.

1.2   Ansatzes

1.2.1   Introduction : Variational Method

The main idea of VQE starts from the variational method which is used in physics to solve several problems. For instance, one can use this method to compute *approximation* of ground state and ground energy of molecules.

Any Hamiltonian $\hat{\mathcal{H}}$ has a lower eigenvalue E_G, the so-called *ground state energy*. So the variational method stands that for any (normalized) state $|\psi\rangle$, the following relation is verified :

$$E_G \leq \langle \psi | \hat{\mathcal{H}} | \psi \rangle \tag{1.19}$$

One can then minimize the expression $\langle\psi|\hat{\mathcal{H}}|\psi\rangle$ over all the states $|\psi\rangle$ in order to find a value closed to the ground state energy (see the complete VQE process in Fig 1.1).

In the VQE algorithm, generating those states is made by the *ansatz*. Finding the good ansatz which generate an interesting bunch of state is an hard-problem due to the properties of the Hilbert space. An ansatz formed by a polynomial number of gate will only generate a polynomial number of states.
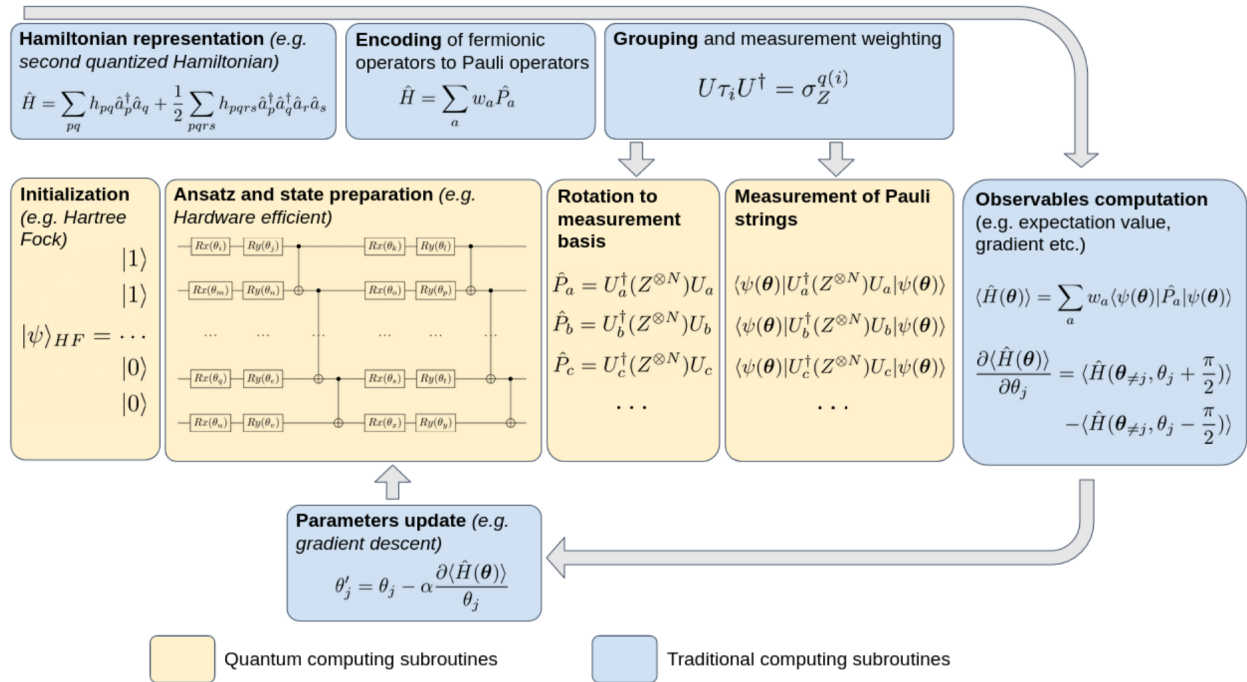

Figure 1.1: A complete plan of the VQE stack [1]

1.2.2 Hardware Efficient Ansatz

As seen before, to approximate the state $|\psi_G\rangle$ corresponding to the ground state energy E_G, the problem of building the right ansatz is the hard part. Thus, when constructing such an ansatz we must balance two opposing goals.

Since our state $|\psi_G\rangle$ belongs to a specific Hilbert space $\mathscr{H}$, ideally we would like our n qubits ansatz to **span as much states** of $\mathscr{H}$ as possible, i.e generate any possible state $|\psi\rangle$ where $|\psi\rangle \in \mathbb{C}^N$ and $N = 2^n$ (n the number of qubits available). This is called the *expressibility* of the ansatz [3]. However we would also like to use as few parameters as possible and so as few to optimise. The main idea of the Hardware efficient ansatz (HEA) is to focus on the first goal.

General construction

Indeed the initial motivation for designing the HEA was for the trial state to be parametrized by quantum gates directly tailored to the quantum device available, and so spanning as much states available to the device. Despite depending on the device used, every HEA follows the same logic: the ansatz is constructed by repeating blocks of inter-weaved single qubit parametrized rotation gates and ladders of entangling gates [1] (see Fig. 1.2). This can be represented as follow :

$$|\psi(\theta)\rangle = \left[\prod_{i=1}^{d} U_{rota}(\theta_i) \times U_{ent}\right] \times U_{rota}(\theta_{d+1}) |\psi_{init}\rangle \tag{1.20}$$

where d is the number of repetition of the scheme, $U_{rota}(\theta_i) = \prod_{q,p} R_p^q(\theta_i^{pq})$, $q = 0, 1, ..., n$ the qubit addresses and $p \in \{X, Y, Z\}$, the selected set of parametrized Pauli rotation.
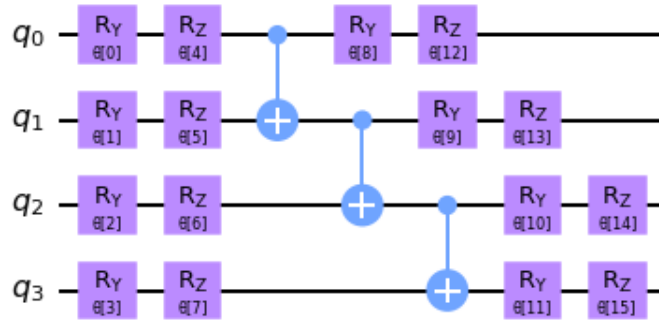

Figure 1.2: Example of one repetition of HEA in Qiskit, using R_y, R_z and $CNOT$ gates, on four qubits.

HEA can be constructed differently depending on the depth of the circuit, the type of rotation gates used and the entanglement setting. This last parameter set the level of entanglement of the circuit, we show here two types of entanglement settings, *linear* where CX gates are applied to adjacent qubits pairs in order, as shown in figure 1.2, and *full entanglement* where a CX gate is applied to each qubits pair in each layer, as shown in figure 1.3.
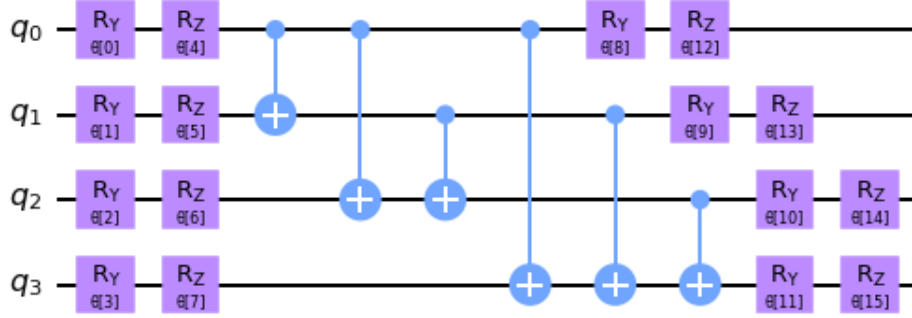
Figure 1.3: Same as figure 1.2 but with *full entanglement*

This way, good approximations on a wide range of states may be obtained despite a number of parameters in $\mathcal{O}(nd)$. Its versatility and ease of construction resulted in it being widely used for numerous small-scale quantum experiments.

1.2.3 Unitary-Coupled Cluster Ansatz

Mean-field method or self-consistent field method (Hartree-Fock)

The solution of the self-consistent field method, the so-called Hartree-Fock state, is an approximation of the wave function under the form of single Slater determinant. From the variational method, one can derive a set of N equation for the N spin-orbitals. This equation is solved by means of an iterative method. In the Mean-field method the effect of all the other individuals on any given individual is approximated as a single effect, reducing the many-body problem to a one-body problem. Finding the one-electron HF wavefunction is equivalent to solve the following equation:

$$\hat{F}\phi_i(1) = \epsilon_i\phi_i(1) \tag{1.21}$$

where $\hat{F}$ is the Fock operator and $\phi_i(1)$ are a set of one-electron wave functions, the HF molecular orbitals.

In the molecular orbital basis the HF state can be written as: $|HF\rangle = \hat{a}_n^\dagger \hat{a}_{n-1}^\dagger \ldots \hat{a}_1^\dagger |vac\rangle$ while in the spin-orbital basis it is written as : $|HF\rangle = (\hat{a}_1^\dagger)^{\phi_1}(\hat{a}_2^\dagger)^{\phi_2}\ldots(\hat{a}_n^\dagger)^{\phi_n}|vac\rangle$.

Hartree-Fock method is unable to approximate the electron correlation effect close to chemical accuracy. To solve the problem above, one can expand the wavefunction into the n-Fock space as a superposition of all determinants. The coefficients of the superposition can be parametrized via two popular methods which are Configuration Interaction (CI) and Coupled-Cluster (CC) [4].

Configuration Interaction (CI)

The full configuration interaction (FCI) expands the wavefunction in the n-Fock space but become quickly intractable due to the number of determinants which is factorial. One can limit this approach to only determinants with a fixed number of excitations from a reference state which is often chosen as the $|HF\rangle$ state : this is the CI method. Excitation operators are defined as :

$$\hat{T} = \sum_{i=1}^{n} \hat{T}_i \tag{1.22}$$

$$\hat{T}_1 = \sum_{\substack{i \in occ \\ a \in virt}} t_a^i \hat{a}_a^\dagger \hat{a}_i \tag{1.23}$$

$$\hat{T}_2 = \sum_{\substack{i>j \in occ \\ a>b \in virt}} t_{ab}^{ij} \hat{a}_a^\dagger \hat{a}_b^\dagger \hat{a}_i \hat{a}_j \tag{1.24}$$

$$\ldots \tag{1.25}$$

where occ and $virt$ space are the occupied and unoccupied sites in the reference state. One cans see the operator $\hat{T}_1$ generate one excitation for one electron of the reference state, the operator $\hat{T}_2$ two excitation and so on. t_a^i and t_{ab}^{ij} correspond to expansion coefficients to minimize. In the exact FCI one has :

$$|FCI\rangle = (1 + \hat{T}) |HF\rangle \tag{1.26}$$

FCI solution can always be approached by increasing the number of excitation operators. The most common used CI is the CISD (Configuration Interaction Single Double) which contains only the terms $\hat{T}_1$ and $\hat{T}_2$.

There are several problems for the Configuration Interaction method :

1. The method converge slowly for highly correlated states

2. It is not size-extensive

Coupled Cluster

The problems of the CI can be solved partially by the Coupled Cluster (CC) using an exponential ansatzs :

$$|\psi\rangle = e^{\hat{T}} |HF\rangle \tag{1.27}$$

In this scenario, the parameters $\vec{t}$ constitute the excitation amplitudes instead of the expansion coefficients. As for CI, CC is truncated for some fixed level, for instance the CCSD :

$$|\psi\rangle = e^{\hat{T}_1 + \hat{T}_2} |HF\rangle \tag{1.28}$$

A major difference between CI and CC is that, for a same level of truncation, CI involve only a polynomial number of determinants while CC involve all determinants in the n-Fock space.

Coupled Cluster also have some problems :

1. It converges only under the assumption of a single reference state, mutli-reference is not possible using this method.

2. The operator $e^{\hat{T}}$ are not unitary ! This implies that it is not possible to use the variational principle.

Unitary Coupled Cluster

The Unitary Coupled Cluster come to solve the problems of CC. We redefine the operators to be unitary [4] :

$$|\psi\rangle = e^{\hat{T}-\hat{T}^\dagger} |HF\rangle \tag{1.29}$$

One can know use the variational principle as follow :

$$E_G = \min_{\vec{t}} \langle HF|e^{-(\hat{T}-\hat{T}^\dagger)}\hat{\mathcal{H}}e^{\hat{T}-\hat{T}^\dagger}|HF\rangle \tag{1.30}$$

Another point to notice is that the space span by this ansatz is the same as the one span by the CC.

We know may want to implement the UCC using a Quantum Computer. This ansatz require 2 steps :

1. Preparing the reference state $|HF\rangle$. One just need to run the Hartree-Fock Algorithm to obtain a good reference with a significant overlap.

2. Application of the UCC unitary $U(\vec{t})$ in order to prepare a wavefunction to evaluate.

Let's focus on the second step, to create such an Unitary one need to map the UCC operators 1.29 to operations that can be performed on a quantum computer using some map (for example Jordan-Wigner). First we need to re-write the operator a little bit :

$$U(\vec{t}) = e^{\sum_j t_j(\hat{\tau}_j - \hat{\tau}_j^\dagger)} \tag{1.31}$$

with $\hat{\tau}_j$ the excitations $\hat{a}_a^\dagger \hat{a}_i$, $\hat{a}_a^\dagger \hat{a}_b^\dagger \hat{a}_i \hat{a}_k \ldots$ and t_j the CC amplitudes. Because excitation operators do not commute, one need to use *trotterization* [5] (more or less to separate the exponential of a sum of non-commutable operators into products of exponentials) to approximate and get for trotter coefficient equal to 1:

$$U_1(\vec{t}) = \prod_j e^{t_j(\hat{\tau}_j - \hat{\tau}_j^\dagger)} \tag{1.32}$$

Applying the Jordan-Wigner transformation one get :

$$(\hat{\tau}_j - \hat{\tau}_j^\dagger) = i \sum_k^{2^{2t_j}-1} P_k^j \tag{1.33}$$

where the P_k^j are some products of Pauli matrices referred as sub-terms. For instance the single cluster excitation terms can be written as :

$$t_i^a(\hat{a}_a^\dagger \hat{a}_i - h.c.) = \frac{it_i^a}{2} \bigotimes_{k=i+1}^{a-1} Z_k(Y_i X_a - X_i Y_a) \tag{1.34}$$

The transformed UCC operator is then :

$$U_1(\vec{t}) = \prod_j \exp\left(it_j \sum_k^{2^{2t_j}-1} P_k^j \right) \tag{1.35}$$

$$= \prod_j \prod_k^{2^{2t_k}-1} \exp(it_j P_k^j) \tag{1.36}$$

second equation is obtained using commutation of the sub-terms derived from same excitation. Equation 1.36 can be implemented on a quantum computer using qubits and the regular model of quantum computing.

The most popular version of UCC is UCCSD where one keep only the Single and Double excitation:

$$\hat{T} \approx \hat{T}_1 + \hat{T}_2 \tag{1.37}$$

The major problem of UCC ansatz lies in the number of parameters to optimize evolving quartically with the number of spin-orbitals.

1.2.4   ADAPT-VQE

The ADAPT-VQE is a VQE method that has the particularity of building the ansatz during the optimisation process. It starts by selecting a pool of operators and adds them progressively to the circuit by choosing at each step the operator that lowers the most the energy of the currently constructed state. Different ADAPT-VQE methods appear by choosing different pools of operators.

The Fermionic-ADAPT-VQE

In the Fermionic ADAPT-VQE [6] the operators of the pool are the ones that appear in the UCC ansatz (1.33). The algorithm starts from a state $|\psi_0\rangle$ (which is often the Hartree-Fock state) and initialize the ansatz to the identity. It then iteratively construct

the states $|\psi_k\rangle = e^{\theta_k^k A_k}...e^{\theta_1^k A_1} |\psi_0\rangle$. To do so it determines at each step the operator from the pool that lowers the most the energy of the current state, ie. the operator $e^{\theta_n A_n}$ with θ_n initialized to 0 such that

$$\frac{\partial}{\partial \theta_n} \langle \psi_k | e^{-\theta_n A_n} H e^{\theta_n A_n} |\psi_k\rangle \big|_{\theta_n=0} = \langle \psi_k | -A e^{-\theta_n A_n} H e^{\theta_n A_n} + e^{-\theta_n A_n} H A e^{\theta_n A_n} |\psi_k\rangle \big|_{\theta_n=0}$$
(1.38)

$$= \langle \psi_k | HA - AH |\psi_k\rangle$$
(1.39)

$$= \langle \psi_k | [H, A] |\psi_k\rangle$$
(1.40)

is maximal. Once the new operator has been chosen, a full VQE optimization is run on all the parameters $\theta_1, ..., \theta_k$ to obtain the lowest energy for a state with the current ansatz. The process stops when adding a new operator to the ansatz has a negligible impact on the energy of the state (ie. when $\forall n, \frac{\partial}{\partial \theta_n} \langle \psi_k | e^{-\theta_n A_n} H e^{\theta_n A_n} |\psi_k\rangle \big|_{\theta_n=0} = \langle \psi_k | [H, A] |\psi_k\rangle \le \epsilon$) where ϵ is a predefined parameter.

The whole process of the Fermionic-ADAPT-VQE can also be seen as trying to approximate the Full Configuration Interraction (FCI)

$$|\psi^{FCI}\rangle = \prod_0^\infty \prod_{n=1}^{N^4} e^{\theta_n^k A_n} |HF\rangle$$
(1.41)

but with the constraint that at each step only one null parameter can be changed.

The Qubit-ADAPT-VQE

The Qubit-ADAPT-VQE [7] follows the exact same steps as the Fermionic-ADAPT-VQE but the operators constructing the ansatz are chosen of the form

$$A = i \bigotimes_{k=1}^{N} P_k$$
(1.42)

where P_i is an operator of the Pauli group. The pool can in fact be reduced by only taking the Pauli strings that appear in the fermionic excitation mapping that verify time-reversal symmetry and removing the Pauli Z operators used for anti-symmetry of the wave function in the Jordan-Wigner mapping since they do not impact the energy estimate. Compared to the Fermionic-ADAPT-VQE this method reduces the number of C-NOT gates in the ansatz but at the cost of a wider pool of operators and more parameters to optimize to achieve a wanted precision.

The QEB-ADAPT-VQE

The Qubit-Excitation-Based(QEB)-ADAPT-VQE presented in [8] is an ADAPT-VQE method that is very similar to the two previous algorithms. However it has two differences with the Ferminionic-ADAPT-VQE:

- The pool of chosen operator is different. The one- and two-body operators from the Fermionic-ADAPT-VQE are replaced by single- and double-qubit-excitation which are generated by the following creation and annihilation operators:

$$Q_j^\dagger = \frac{X_j - iY_j}{2} \qquad\qquad Q_j = \frac{X_j + iY_j}{2} \tag{1.43}$$

  ie. operators that are defined as the Fermionic operators in the Jordan-Wigner mapping but without the Pauli-Z strings that account for the anti-commutation relations. One can then define the skew-hermitian single- and double-qubit-excitation operators

$$T_i^k(\theta) = \theta(Q_k^\dagger Q_i - Q_i^\dagger Q_k) \qquad T_{ij}^{kl}(\theta) = \theta(Q_k^\dagger Q_l^\dagger Q_i Q_j - Q_i^\dagger Q_j^\dagger Q_k Q_l) \tag{1.44}$$

  which are represented on the ansatz by the unitaries

$$U_{ki}(\theta) = e^{T_i^k(\theta)} \qquad U_{klij}(\theta) = e^{T_{ij}^{kl}(\theta)} \tag{1.45}$$

- At step n, when it determines the next operator to add to the ansatz, instead of choosing the one with with highest $\frac{\partial}{\partial \theta_i} \langle \psi_n | e^{-\theta_i A_i} H e^{\theta_i A_i} | \psi_n \rangle \big|_{\theta_i=0}$ it keeps the top k operators (with k a predefined number). It then runs a complete VQE for each of the k new ansatzes obtained by adding the different operators to the current circuit and finaly picks the operator that gave the lowest energy.

The main interest of this method is that it has less CNOT gates than the qubit-ADAPT-VQE but the same amount of parameters as the Fermionic one.

1.3   Other aspects of VQE

1.3.1   Classical Optimisers

Hybrid algorithms like the VQE allegedly uses the best of both worlds, classical optimisation and quantum mechanics. Therefore in order to have a well functioning VQE one also need a sufficient optimiser. There is a plethora of different classical optimisers that are available and implemented within the Qiskit environment. Qiskit suggest that when running a VQE on NISQ hardware one should use the classical optimiser, Simultaneous Perturbation Stochastic Approximation (SPSA) [9]. Otherwise a common optimiser when running on a classical quantum simulator is to to use Constrained Optimisation By Linear Approximation (COBYLA).

Lavrijsen et. all showed in [10] that in order to have a well functioning VQE on quantum noise hardware the classical optimiser must be carefully chosen as well. Furthermore, in order to get valid scientific results on NISQ hardware they argue that choosing a well tuned optimiser is crucial. In their paper they test how different optimisers behave when applying noise to the quantum gates in a classical quantum simulator. The results show that classical optimisers who are unable to take noise in consideration end up loosing chemical accuracy for a lower amount of noise than the optimisers that are noise-aware. Best performing optimiser under the presence of noise for their quantum chemistry example was the Implicit Filtering (ImFil) [11]. ImFil is implemented as in MATLAB but as part of the work with the article Lavrijsen et. all has rewritten the original MATLAB implementation to Python. Which can be acessed by the SCIKIT-QUANT library that consists of all the optimisers they have analysed in the paper [12].

Sequential Least Squares Programming (SLSQP)

Qiskit uses the Scipy package scipy.optimize.minimize SLSQP [13]. Which uses the Sequential Least Squares Programming method. That was developed by Dieter Kreft in 1980s. The algorithm is used to optimise general nonlinear problems that minimize a scalar function.

Constrained Optimisation By Linear Approximation (COBYLA)

Qiskit uses the Scipy package scipy.optimize.minimize COBYLA. Which uses the Constrained Optimisation By Linear Approximation method. That was developed by Powell in 1994 [14]. The algorithm optimises a constrained problem where the function has no derivative. It works by each iteration forms a linear approximation to the objective and constraint function by interpolation at the vertices of a simplex and trust region bound restricts each change to the variable.

Simultaneous Perturbation Stochastic Approximation (SPSA)

Qiskit uses the Scipy package scipy.optimize.minimize SPSA.The SPSA approximates the gradient of the objective function using two measurements per iteration [15]. This is regardless of the dimension of the optimisation problem. The two measurements are made by randomly varying the all of the variables in the problem so called simultaneous perturbation. Comparing with the finite-difference method that need to make $2p$ measurements per iteration when optimising p parameters, varying one variable at the time.

Implicit Filtering (IMFIL)

Qiskit uses the Scikit package where Lavrijsen et. all rewrote the MATLAB implementation to python. The IMFIL algorithm is designed for problems with local minimas that is caused by noise. IMFIL is a sampling method which means that the optimisation is controlled by evaluating the function f at a cluster of points. Depending on the local minima found by

the cluster through interpolation drives the next cluster of points. The optimiser converges when the cluster has reached it's smallest size.

1.3.2 Barren Plateau

In this section, we will introduce the problem of Barren Plateau (BP) which is common in VQA. BP refers to an issue in the choice of the cost function (which for us is the expectation value of the Hamiltonian on the current state) or of the optimizer. When a BP of the cost function is reached by the optimizer, the gradient will vanished in many/all directions causing a lot of troubles for the optimizer to reach the global minimum. It has the effect of the landscape being essentially flat, for instance when reaching a local minima. Then depending on the existence or not of BP, one can choose to use a gradient-based or a gradient-free optimization method. It was shown that deep unstructured parametrized quantum circuits tend to exhibit BPs when they are randomly initialized.

Chapter 2

Implementation

Implementation link: Links for the notebooks are in reference [16], [17], [18], [19].

We have chosen to implement the VQE algorithm using the open-source framework Qiskit Nature [20] on IBMQ's quantum computer and classical simulator. The primary reason for choosing Qiskit as our software development kit over OpenFermion (Google's open source library) is its large community of users and its ease of use. Additionally the ADAPT-VQE algorithm was first implemented using OpenFermion in [6] and it seemed more interesting to try to do it with another development kit.

In fig(2.1) we see the structure of Qiskit Nature and the different parts that are necessary to finally run a quantum algorithm, in our case a VQE.
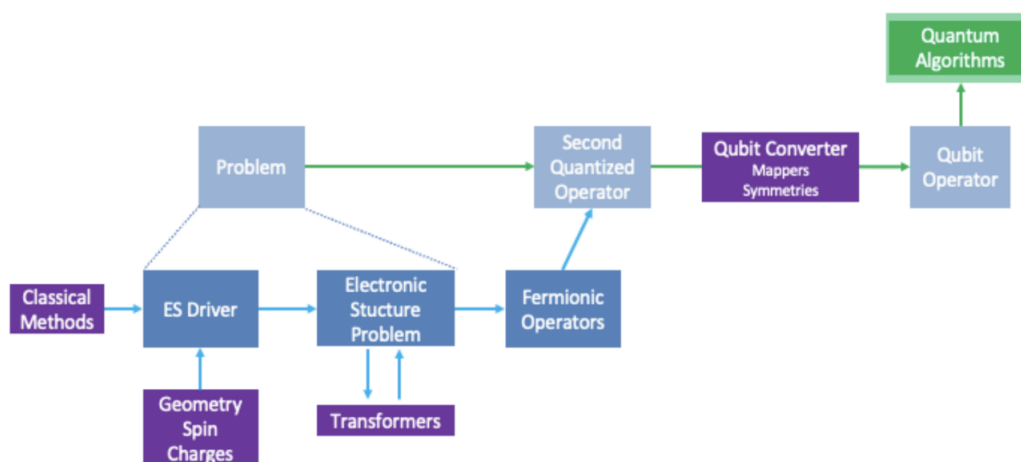

Figure 2.1: Qiskit Nature outline, [21]

2.1 Qiskit general framework

2.1.1 Hamiltonian Mapping

As the goal of our implementation is to find the ground state energy of a particular molecule, we start by describing the energy of our system with the following Hamiltonian

$$\hat{\mathcal{H}} = -\left(\sum_i \frac{\hbar^2}{2m_e}\nabla_i^2 + \sum_{i,k} + \frac{e^2}{4\pi\epsilon_0}\frac{Z_k}{|r_i - R_k|} + \frac{1}{2}\sum_{i\neq j}\frac{e^2}{4\pi\epsilon_0}\frac{1}{|r_i - r_j|} \right). \tag{2.1}$$

We rewrite the Hamiltonian in the second quantisation form (see equation (2.2)) using the molecular orbitals (MOs), ϕ_u together with the annihilation and creation operators (see First and Second Quantization of the Hamiltonian)

$$\hat{\mathcal{H}} = \sum_{pq} h_{pq}\hat{a}_p^\dagger\hat{a}_q + \frac{1}{2}\sum_{pqrs} h_{pqrs}\hat{a}_p^\dagger\hat{a}_q^\dagger\hat{a}_r\hat{a}_s \tag{2.2}$$

$$h_{pq} = \int d\xi\phi_p^*(\xi)\left(-\frac{\hbar^2}{2m_e}\nabla^2 - \sum_k\frac{e^2}{4\pi\epsilon_0}\frac{Z_k}{|r - R_k|} \right)\phi_q(\xi) \tag{2.3}$$

$$h_{pqrs} = \frac{e^2}{4\pi\epsilon_0}\int d\xi_1 d\xi_2\frac{\phi_p^*(\xi_1)\phi_q^*(\xi_2)\phi_r(\xi_1)\phi_s(\xi_2)}{|r_1 - r_2|} \tag{2.4}$$

The MOs are found by solving the Hartree-Fock (HF) state as described earlier in Mean-field method or self-consistent field method (Hartree-Fock). This is done by using the classical solver PySCF *driver* [22] which is an open-source collection of electronic structure modules. To get the second quantized Hamiltonian we use the function Electronic-StructureProblem.second_q_op [23] on the MOs which then returns the operators for the Hamiltonian.

We now have the Hamiltonian of the initial HF state and need to rewrite it in a way a quantum computer can handle. As each qubit in a quantum computer can be used to represent a spin orbital we once again rewrite the Hamiltonian using the Jordan-Wigner mapping (Mapping) which assigns each qubit to a spin orbital by mapping the creation operators as in equation (1.17).

2.1.2 Operators representation

To represent unitaries (and in general the operators we handle during the various implementation) we use different kind of classes. To decompose pre-existing ansatzes (such as UCCSD), we use the `Instruction` class[24] which represent circuit instructions, which in our case are excitation operators.

When we create operators from scratch, we obviously have to start with Paulis strings, then we can use ether the `PauliOp` class [25], which can only represent a tensor of Paulis, or the `SparsePauliOp`[26] that can represent a linear combination of tensor of Paulis. Then

when we want to create unitaries that represent parameterized excitation operator as in 1.45, we have to call `PauliEvolutionGate`[27] on the two class mentioned above, which can represent such unitaries, and then be appended to a circuit.

2.1.3 Aer simulator

To simulate our circuits/ansatzes, we use the `AerSimulator` backend [28] from IBM Qiskit. The default behavior of this simulator is to mimic the execution of an actual noise-free device. Obviously the major drawback of this technology is the computation time which starts being very long when we compute it for molecules different than H_2.

2.1.4 VQE

To run a VQE, we use the qiskit class VQE [29]. This class takes an ansatzes, an optimizer, an initial state (usually the HartreeFock state 1.2.3), a quantum instance such as the `AerSimulator`[28], and an initial point representing the starting point for the vector of parameters. We then call the `compute_minimum_eigenvalue()` method which starts the whole VQE process and returns a data type with mutliple interesting values (optimizer time, number of cost function evaluation, eigenvalue, eigenstate ...).

2.2 Pre-existing Ansatz

2.2.1 Hardware Efficient

As mentioned previously in Hardware Efficient Ansatz the ansatz is made of a series of repeating rotation gates followed by CNOT gates. This is implemented using the EfficientSU2 function [30] that creates a circuit with the specific pattern of gates described before.

2.2.2 UCCSD

For the UCCSD ansatz we only keep the first and second excitation operators as discussed previously. These are generated by using the generate_fermionic_excitations function [31].

2.3 ADAPT-VQE development

This part concerns all the development done to implement the three ADAPT-VQE methods introduced in ADAPT-VQE. There exist an AdaptVQE class [32] in Qiskit wich aims to implement the Fermionic-ADAPT-VQE method, but due to several `raise NotImplementedError` exceptions in the subclasses and a unexplicit description of the method, we have decided

to start from scratch. We follow the scheme in fig(2.2). The main focus in the development of these methods, is the definition of the pool of operators.

2.3.1 Fermionic-ADAPT-VQE

After initializing our state to the Hartree-Fock state as for the UCCSD, we start with an empty ansatz and measure which operator from the pool achieves the largest gradient using the `commutator(op_a, op_b)`[33] class between the operators and the hamiltonian and then append that operator to the ansatz (which is the case for every methods).
In this case building the operator pool is pretty straightforward: since the Fermionic-ADAPT-VQE method use directly the excitation operators of the UCCSD ansatz, we construct one time this ansatz and then decompose every excitation operator to create our pool (cf. Operators representation to see each decomposition step of the ansatz using several Qiskit methods).
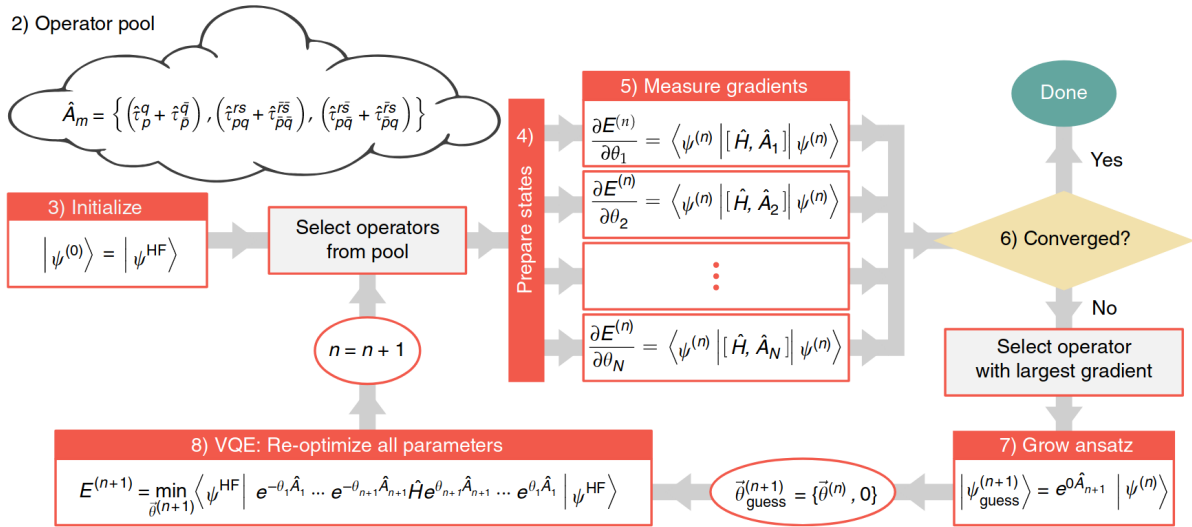

Figure 2.2: Adapt-VQE schematic, [6]

2.3.2 Qubit-ADAPT-VQE

Here the operators considered are all possible combinations of the N-tensor of Pauli operators, N being the number of spin orbitals/qubits of our circuit. The numbers of operator can be widely reduced using only the Pauli strings which appear in the operators of the fermionic pool created before. The fermionic pool was represented as a list of `PauliSumOp`[34] which are made of `SparsePauliOp`[26] that contain a list of strings representing the Pauli strings. We can then extract the strings from the pool and, after removing the redundant elements, create a new pool with the associated `PauliOp`[25] objects. The last step consists in reducing one more time the number of operators by removing randomly

3/4 of them. This can be done without a significant loss of accuracy as it was shown in the article [7].

2.3.3 QEB-ADAPT-VQE

This method lies in between rigorous problem-tailored methods as for the fermionic one, and more machine learning-like methods as for the qubit one, were the pool of operator is not tailored to the physics of the problem, but more aimed to have a wide range of options. Due to this *"in-between"* aspect, this version was the hardest to implement.

Since it's easier to work with unitaries defined on the whole circuit, to create the Q_i described in 1.43, we create the sum of string of paulis $Q'_i = I^{\otimes i-1} \otimes Q_i \otimes I^{\otimes N-i-1}$ defined on the whole qubit space. The same way, we then define the $T_i'^k$ as in 1.44 and finally the whole unitary $e^{T_i'^k}$ as in 1.45 (following Operators representation section as always to create our circuit).

Chapter 3

Results and discussion

The main goal of this chapter is to evaluate the accuracy and computing performance arising from the different ansatzes described above 1.2. The framework of the experiments is sampling the potential energy surface (PES) of different molecules, H_2 and LiH. All of this is done in order to provide a better understanding of all the theoretical notion addressed in the previous sections.

3.1   Comparison of various HEA

In this section, we will compare different HEA constructions trough a simple experiment, computing the PES of H_2, since its scope of application is restricted to small sized quantum experiment. We will evaluate the effectiveness of the different construction methods with 2 criteria:

1. The error in the calculated ground energy E_{HEA} with respect to the FCI solution, E_{FCI}.

2. The number of function evaluations required for convergence

First lets display an overview of the different methods by testing them with different (Classical Optimisers efficient ansatz) (COBYLA, L_BFGS_B, SLSQP,SPSA) in order to have a wider view of the ansatzes peformance, and to be sure as much as possible to not bottleneck it because of a unhadapted optimizer (we do not provide here a deep reflections about the compatibility of the optimizers, but only focus on the best ansatz performances) . The ansatzes considered will be:

- $R_y R_z$ with *linear entanglement*
- $R_y R_z$ with *full entanglement*
- R_x with *linear entanglement*

All of them with 3 repetition of the pattern (to keep the circuits shallow)
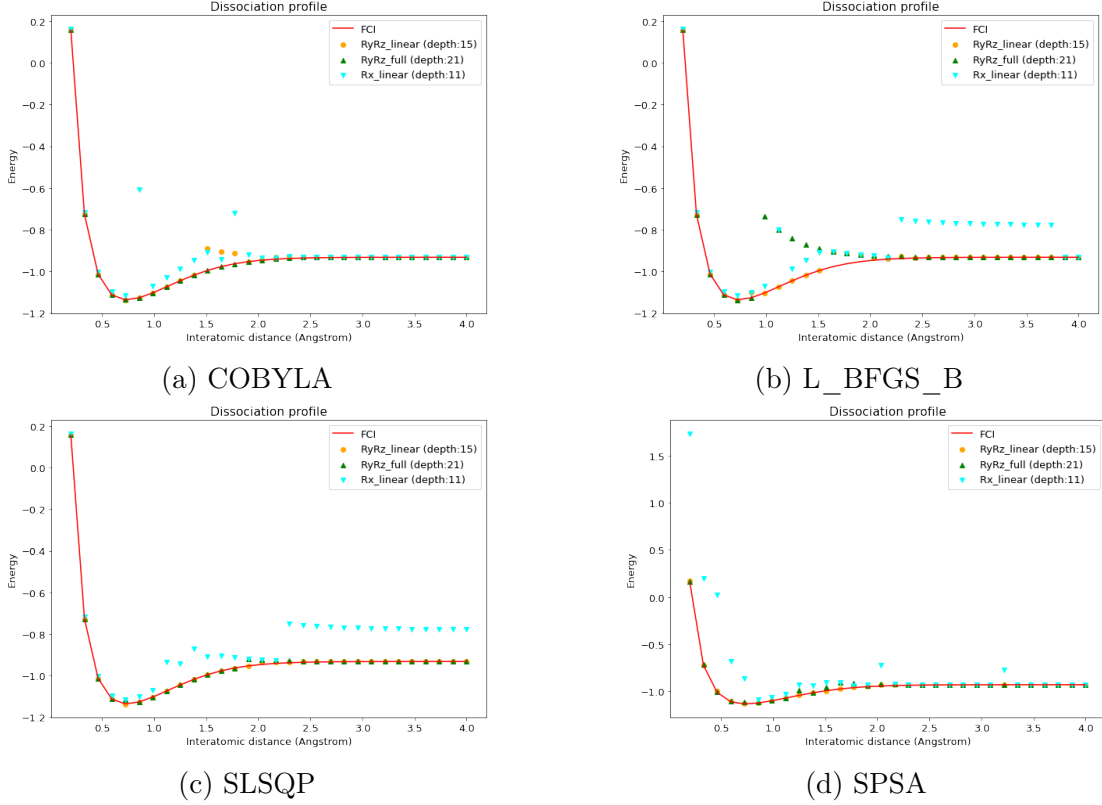

(a) COBYLA

(b) L_BFGS_B

(c) SLSQP

(d) SPSA

Figure 3.1: Energy calculated depending on the different optimizer used, for each HEA cited above with a threshold of 3000 evaluations.

As we can see on Figure3.3, R_x having only one rotation gate, increases drastically the error compared to the FCI results. Indeed, the range of states it can access is reduced, having only one rotation axis per qubit, but with COBYLA 3.1a R_x was optimized well enough to tailor the FCI results for close ($[0.2, 1]$) and far ($[2, 4]$) inter-atomic distance .
For the *fully entangled* and *linearly entangled* R_yR_z the difference is not very noticeable (see [35] for more reflections on entanglement-related aspects of the dissociation of H_2). However for the *fully entangle* HEA in the case of 3.1b, L_BFGS_B was not able to optimize well enough the circuit parameters resulting in a important error in the $[1, 2]$ interval .
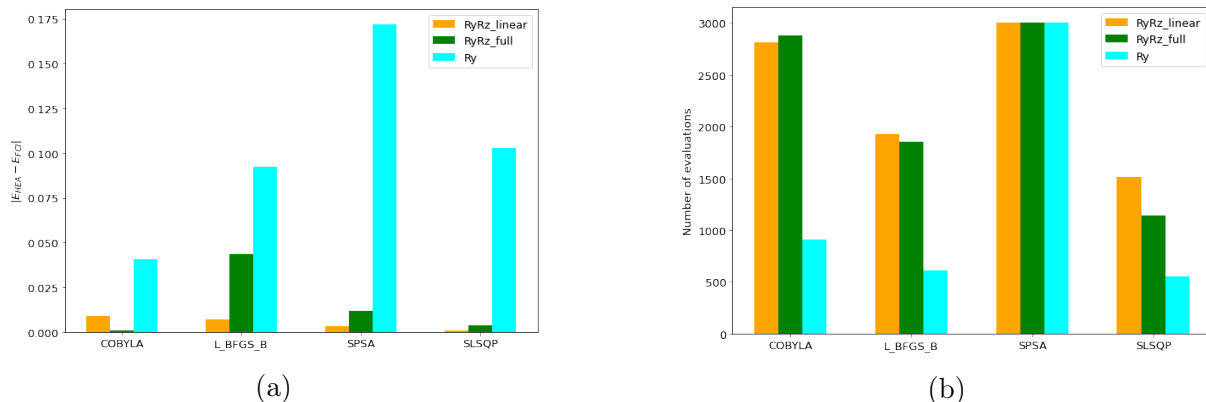In general we get pretty good results for the two R_yR_z HEA with 3.1a, 3.1c and 3.1d.

Figure 3.2: Overview of the error of the energy calculated with respect to the FCI solution and the number of function evaluations

The low number of function evaluations of the R_x HEA on figure 3.2b, confirm the fact that the simplest ansatz containing the solution will also have the best performance, thank to the solution space having less parameters to optimize, of course in this case, it is offset by the important errors brought by this ansatz for certain interatomic distances. We also see in 3.2a the two R_yR_z HEA having pretty small errors.

Overall despite not tailoring the problem considered, HEA ansatz can be interesting in the case of small chemical problems as the one shown above. With shallow circuit depth and few CX gates it is possible to sample H_2 ground energy with good chemical accuracy. Since it has to span a large portion of the Hilbert space to represent with enough accuracy the ground state of the wave function considered, HEA can therefore be quite inefficient requiring in worst cases an exponential depth and so unlikely to be suitable for larger-scale chemical problems.

## 3.2 Evaluation of the accuracy of UCCSD

Now that we've define a lower bound for the performance of the ansatzes with HEA, let's evaluate our first problem-tailored ansatz, the UCCSD andatz. Firstly we will evaluate the performance of the UCCSD for the simulating the PES of H_2, hence to make sure that it is at least as accurate as HEA, and then we will do the same for LiH and BeH, two more complex molecules, unreachable for reasonable HEA (we have actually computed simulation of these molecules for HEA, but the precision was to poor to be displayed or compared with other ansatzes).
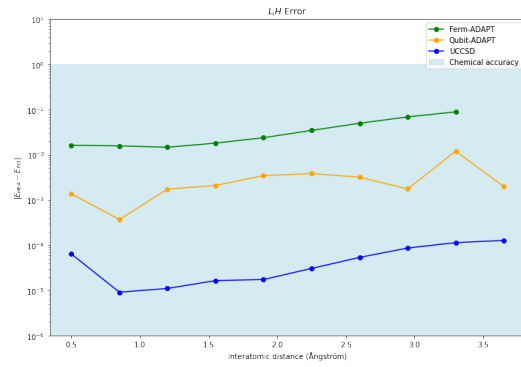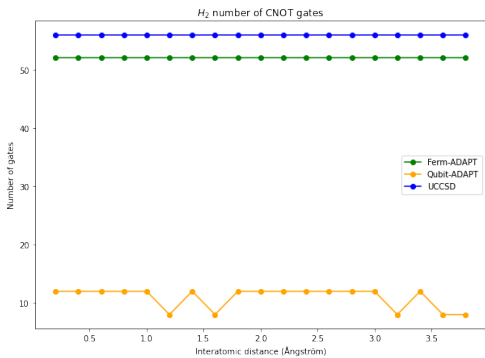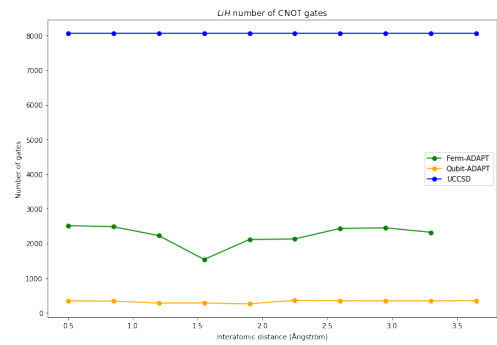
(a)

(b)

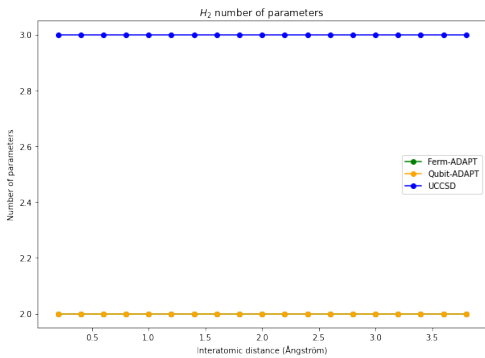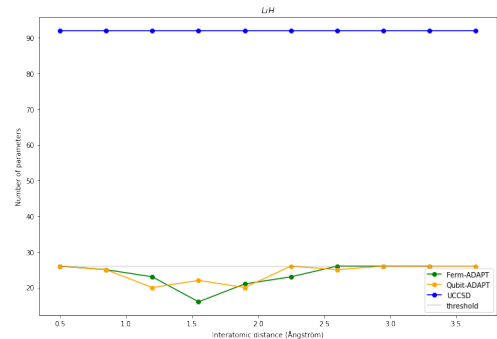(c)

(d)

(e)

(f)

(g)

(h)

H_2

LiH
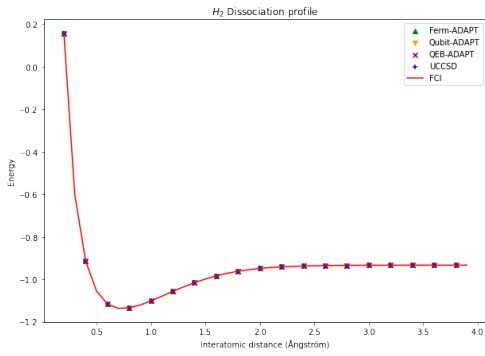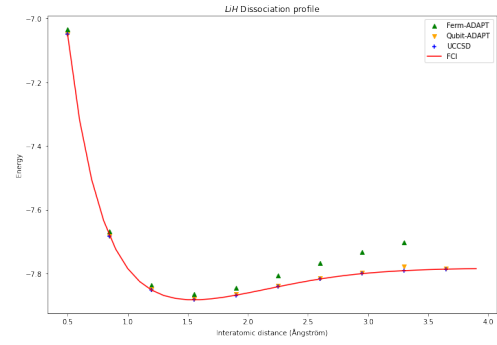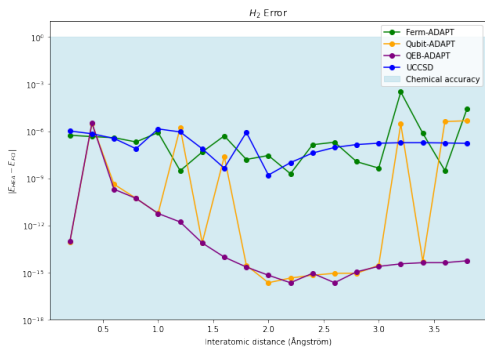
Figure 3.3

3.3 Comparison of ADAPT-VQE Methods

3.3.1 Results for H_2

Accuracy

On figure 3.3c, one can see that each algorithm performs well with an error within the chemical accuracy (which is the accuracy required to make realistic chemical predictions).

Comparing to the energy obtained by FCI (figure 3.3a), all methods fit the curve pretty well. There are still some slight variations due to the number of terms in the ADAPT algorithms which is always lower than a full CI or even a UCC.

When comparing the error for Fermionic ADAPT and UCCSD (figure 3.3a, one would expect the UCCSD error to always be lower than the Fermionic-ADAPT one. In fact, this is not always the case due to conditions over the convergence. Here are plot the best value depending on the parameters given to the optimizer and less parameters are given for the fermionic-ADAPT. We can expect UCCSD to have better result with more flexible conditions. The main issue stays the computation time.

On figure 3.3c, one can see that Qubit and QEB ADAPT performs better than the Fermionic ADAPT and the UCCSD. Indeed, due to the larger pool of operators available, Qubit and QEB algorithms have more options to approximate the correct energy value.

Number of CNOT and of parameters

Because qiskit has many classes of object that are not always compatible, we coded QEB-Adapt with the `HamiltonianGate` [36] class which needs a lot of computation time during the VQE, a better way would be to use `PauliEvolitionGate`[27] used in Qubit-Adapt, wich is way faster to use in the VQE and allows to decompose the circuit, but the pauli string in the QEB-Adapt include complex coefficient wich are not allowed in the `PauliEvolutionGate`. A solution would have been to re-developped QEB-Adapt tailoring the `PauliEvolutionGate` framework. The `HamiltonianGate` does not allow us to decompose the circuit, thus we were not able to retrieve data on the number of parameters or CNOT gates as we did for the other methods.

As we can see in 3.3g, all the methods have a constant number of parameters (number of operators added to the final ansatz) with the one of the Qubit-ADAPT being lower than the one of the UCCSD and the Fermionic-ADAPT. As expected the number of CNOT gates in the ansatz is lower for the Qubit-ADAPt than for the two other methods. It is also the only method that present variations in the number of CNOT gates and it seems interesting to remark that the drops (see 3.3e) in number of CNOT gates are directly linked with the increase in the error that can be seen in the plot above.

3.3.2   Results for LiH

Accuracy

Concerning the QEB-ADAPT, we were not able to run it for any other molecule than H_2 because the number of operators in the pool grows too fast (for instance, it is around 1500 for LiH) which made it way too long for us to compute (in [8] they parallelize this computation).

As for H_2, one can see that each algorithm performs well with an error within the chemical accuracy. Due to very high computation time of these methods (around 24h), we had to put a threshold on the number of parameters (see figure 3.3h), thus we were not able to evaluate the full potential of Qubit-ADAPT and Ferm-ADAPT. Hence we see in 3.3d that UCCSD with still all the parameters has a lower error.

Number of CNOT and of parameters

As explained above the number of operators in both ADAPT methods is thresholded due to computation time and is thus way smaller than the one for UCCSD. However it is interesting to remark that the Qubit-ADAPT has a lower error than the Fermionic one with a smaller number of CNOT gates.

Conclusion

Through this project, we were able to tackle new fields such as quantum chemistry and the concept of Variational Quantum Algorithms, allowing us to discover possible near term applications of NISQ devices and so maybe providing feasible Quantum Advantage. While bringing a theoretical background on quantum chemistry, it allows to have enough knowledge to understand and approach the concepts of VQE, and in our case, especially ansatzes. Through the use of IBM's Qiskit we were able to develop and evaluate every ansatzes that we planned to implement from the simplest Hardware efficient to the hardest one, the QEB ADAPT ansatz. In the end we learned that, through efficient machine learning methods, the Qubit-ADAPT-VQE ansatz was able to bring high accuarcy while keeping a low number of parameters and reasonable number of CNOT gates (2-qubit gates being hard to implement),thus possibly being the most feasible ansatz.

Bibliography

[1]   Jules Tilly Hongxiang Chen Shuxiang Cao Dario Picozzi Kanav Setia Ying Lif Edward Grant Leonard Wossnig Ivan Runggerg George H. Boothh Jonathan Tennysona. "The Variational Quantum Eigensolver: a review of methods and best practices". In: (Nov. 2021). URL: https://arxiv.org/abs/2111.05176.

[2]   Erik Koch. *Many-Electron States*. URL: https://www.cond-mat.de/events/correl13/manuscripts/koch.pdf.

[3]   Kouhei Nakaji Naoki Yamamoto. "Expressibility of the alternating layered ansatz for quantum computation". In: (Apr. 2021). URL: https://arxiv.org/pdf/2005.12537.pdf.

[4]   Jonathan Romero Ryan Babbush Jarrod R McClean Cornelius Hempel Peter Love and Alán Aspuru-Guzik. "Strategies for quantum computing molecular energies using the unitary coupled cluster ansatz". In: (Feb. 2018). URL: https://arxiv.org/abs/1701.02691.

[5]   Grant Kluber. *Trotterization in QM Theory*. 2020. URL: https://web.ma.utexas.edu/users/drp/files/Fall2020Projects/DRP_Final_Project_F2020%5C%20-%5C%20Grant%5C%20E%5C%20Kluber.pdf.

[6]   Harper R. Grimsley et al. "An adaptive variational algorithm for exact molecular simulations on a quantum computer". en. In: *Nature Communications* 10.1 (Dec. 2019), p. 3007. ISSN: 2041-1723. DOI: 10.1038/s41467-019-10988-2. URL: http://www.nature.com/articles/s41467-019-10988-2 (visited on 03/28/2022).

[7]   Ho Lun Tang et al. "Qubit-ADAPT-VQE: An Adaptive Algorithm for Constructing Hardware-Efficient Ans\"atze on a Quantum Processor". In: *PRX Quantum* 2.2 (Apr. 2021). Publisher: American Physical Society, p. 020310. DOI: 10.1103/PRXQuantum.2.020310. URL: https://link.aps.org/doi/10.1103/PRXQuantum.2.020310 (visited on 03/29/2022).

[8]   Yordan S. Yordanov et al. "Qubit-excitation-based adaptive variational quantum eigensolver". en. In: *Communications Physics* 4.1 (Oct. 2021). Number: 1 Publisher: Nature Publishing Group, pp. 1–11. ISSN: 2399-3650. DOI: 10.1038/s42005-021-00730-0. URL: https://www.nature.com/articles/s42005-021-00730-0 (visited on 03/29/2022).

[9]   *Simulating Molecules using VQE*. en. URL: https://community.qiskit.org/textbook/ch-applications/vqe-molecules.html (visited on 04/14/2022).

[10] Wim Lavrijsen et al. "Classical Optimizers for Noisy Intermediate-Scale Quantum Devices". In: *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*. Oct. 2020, pp. 267–277. DOI: `10.1109/QCE49297.2020.00041`.

[11] *4. The Implicit Filtering Algorithm*. en. ISBN: 9781611971903. URL: `https://epubs.siam.org/doi/epdf/10.1137/1.9781611971903.ch4` (visited on 04/14/2022).

[12] *QAT4Chem - Software*. sv. URL: `https://qat4chem.lbl.gov/software` (visited on 04/14/2022).

[13] *scipy.optimize.minimize — SciPy v1.8.0 Manual*. URL: `https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html` (visited on 05/09/2022).

[14] M. J. D. Powell. "A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation". en. In: *Advances in Optimization and Numerical Analysis*. Ed. by Susana Gomez and Jean-Pierre Hennart. Dordrecht: Springer Netherlands, 1994, pp. 51–67. ISBN: 978-90-481-4358-0 978-94-015-8330-5. DOI: `10.1007/978-94-015-8330-5_4`. URL: `http://link.springer.com/10.1007/978-94-015-8330-5_4` (visited on 04/13/2022).

[15] *SPSA Algorithm*. URL: `https://www.jhuapl.edu/spsa/` (visited on 04/14/2022).

[16] Alteg M. Chevalier B. Mestoudjian O. Rossi JL. *Notebook VQE ADAPT*. URL: `https://colab.research.google.com/drive/1kO87dhB_7lVMATXzf7OHDh_-LrBnCVYG?usp=sharing`. (accessed: 09.05.2022).

[17] Alteg M. Chevalier B. Mestoudjian O. Rossi JL. *Notebook HEA*. URL: `https://colab.research.google.com/drive/1fr3Udy08clqKl-OrVQtlPRL3l3cOmFRo?usp=sharing`. (accessed: 09.05.2022).

[18] Alteg M. Chevalier B. Mestoudjian O. Rossi JL. *Notebook UCCSD H2*. URL: `https://colab.research.google.com/drive/1Ql-lg-g7k_PwuM6eNEcmY9SNdk9Xl1eT?usp=sharing`. (accessed: 09.05.2022).

[19] Alteg M. Chevalier B. Mestoudjian O. Rossi JL. *Notebook UCCSD LiH*. URL: `https://colab.research.google.com/drive/1tVOcLinfO5qYvH8jCA1Qj2ioAwOJYmQZ?usp=sharing`. (accessed: 09.05.2022).

[20] *Qiskit Nature*. original-date: 2021-01-27T16:59:03Z. Apr. 2022. URL: `https://github.com/Qiskit/qiskit-nature` (visited on 04/28/2022).

[21] Qiskit. *Introducing Qiskit Nature*. en. Apr. 2021. URL: `https://medium.com/qiskit/introducing-qiskit-nature-cb9e588bb004` (visited on 04/28/2022).

[22] *Home — PySCF*. URL: `https://pyscf.org/` (visited on 04/28/2022).

[23] *ElectronicStructureProblem — Qiskit Nature 0.3.2 documentation*. URL: `https://qiskit.org/documentation/nature/stubs/qiskit_nature.problems.second_quantization.ElectronicStructureProblem.html#qiskit_nature.problems.second_quantization.ElectronicStructureProblem` (visited on 04/28/2022).

[24] *Instruction*. URL: `https://qiskit.org/documentation/stubs/qiskit.circuit.Instruction.html` (visited on 04/30/2022).

[25] *PauliOp*. URL: `https://qiskit.org/documentation/stubs/qiskit.opflow.primitive_ops.PauliOp.html` (visited on 04/30/2022).

[26] *SparsePauliOp*. URL: https://qiskit.org/documentation/stubs/qiskit.quantum_info.SparsePauliOp.html (visited on 04/30/2022).

[27] *PauliEvolutionGate*. URL: https://qiskit.org/documentation/stubs/qiskit.circuit.library.PauliEvolutionGate.html (visited on 04/30/2022).

[28] *AerSimulator*. URL: https://qiskit.org/documentation/stubs/qiskit.providers.aer.AerSimulator.html (visited on 04/30/2022).

[29] *VQE — Qiskit 0.36.1 documentation*. URL: https://qiskit.org/documentation/stubs/qiskit.algorithms.VQE.html (visited on 05/02/2022).

[30] *EfficientSU2 — Qiskit 0.36.1 documentation*. URL: https://qiskit.org/documentation/stubs/qiskit.circuit.library.EfficientSU2.html (visited on 04/30/2022).

[31] *generate_fermionic_excitations — Qiskit Nature 0.3.2 documentation*. URL: https://qiskit.org/documentation/nature/stubs/qiskit_nature.circuit.library.ansatzes.utils.generate_fermionic_excitations.html#qiskit_nature.circuit.library.ansatzes.utils.generate_fermionic_excitations (visited on 04/29/2022).

[32] *AdaptVQE — Qiskit Nature 0.3.2 documentation*. URL: https://qiskit.org/documentation/nature/stubs/qiskit_nature.algorithms.AdaptVQE.html (visited on 04/28/2022).

[33] *qiskit opflow commutator*. URL: https://qiskit.org/documentation/stubs/qiskit.opflow.commutator.html (visited on 04/30/2022).

[34] *PauliSumOp*. URL: https://qiskit.org/documentation/stubs/qiskit.opflow.primitive_ops.PauliSumOp.html (visited on 04/30/2022).

[35] Rodolfo O Esquivel Nelson Flores-Gallegos Moyocoyani Molina-Espíritu A R Plastino Juan Carlos Angulo Juan Antolín Jesús S Dehesa. "Quantum entanglement and the dissociation process of diatomic molecules". In: (Aug. 2011). URL: https://iopscience.iop.org/article/10.1088/0953-4075/44/17/175101.

[36] *HamiltonianGate*. URL: https://qiskit.org/documentation/stubs/qiskit.extensions.HamiltonianGate.html (visited on 04/30/2022).