Quantum Computing for Quantum Chemistry : examination sheet

Axel Courtat, Daniele Loco and Jerome Foret

November 2023

The examination proposed for the *Quantum Computing for Quantum Chemistry* lectures done by **Qubit Pharmaceuticals** shall be done within a week : delivered the 14th of December, the deadline for the results submission will be the 22th at 11:59pm.

1 Subject of the examination

The examination will focus on the quantum algorithms explained and studied during the lectures, with direct applications to small molecular systems.
In more details, it will mainly focus on the **Variational Quantum Eigensolver (VQE)** algorithm, completed by a smaller work on its well-known adaptive modification **ADAPT-VQE**.

You will be given a folder containing several scripts providing an implementation of a basic VQE algorithm adapted for quantum chemistry calculations. All the scripts are in the native language Python, and using respectively the package **Openfermion** for the molecular settings, the SDK **Qiskit** for the quantum computing part of the algorithm and **Scipy** for the classical optimization part of the algorithm. In more details:

- main.py: the main python script to launch all the computations. Use the command *python main.py -h* to see the options of the parser !
- VQE.py: contains the main class VQE that provides all of the VQE process (both quantum and classical)
- molecule.py: mainly using Openfermion to setup the studied molecular system
- excitations.py: contains the circuits for both the single and double fermionic and qubit-efficient excitations
- ansatze.py: contains the UCCSD ansätz for the VQE algorithm

- useful_functions.py: contains several functions used by the other files

You are asked to return before the end of the deadline given above, an archive (zip, tar) containing all of the works you have done and send it in one email to the following email addresses:

- axelc@qubit-pharmaceuticals.com
- daniele@qubit-pharmaceuticals.com
- jerome@qubit-pharmaceuticals.com

2 Expected work

2.1 Molecular systems

For all of the examination, you will work with 4 molecular systems. Each of them will use sligthy different molecular settings, that can be adjusted through Openfermion in the molecule.py file. Explicitely :

1. **H_2 molecule**:
   - contains 2 electrons and computed with 4 orbitals
   - set *occupied_indices = range(0)*
   - set *active_indices = range(0, molecule.n_orbitals)*
   - set *energy_threshold = 1e-10* in VQE.py
   - for a test, use the bond-length 0.74
2. **H_4 molecule**
   - contains 4 electrons and computed with 10 orbitals
   - set *occupied_indices = range(1)*
   - set *active_indices = range(1, molecule.n_orbitals)*
   - set *energy_threshold = 1e-10* in VQE.py
   - for a test, use the bond-length 0.74
3. **full LiH molecule**
   - contains 4 electrons and computed with 12 orbitals
   - set *occupied_indices = range(0)*
   - set *active_indices = range(0, molecule.n_orbitals)*
   - set *energy_threshold = 1e-3* in VQE.py
   - for a test, use the bond-length 1.42
4. **full BeH_2 molecule**

- contains 6 electrons and computed with 14 orbitals
- set *occupied_indices = range(0)*
- set *active_indices = range(0, molecule.n_orbitals)*
- set *energy_threshold = 1e-2* in VQE.py
- for a test, use the bond-length 1.11

5. **frozen-core LiH molecule** bonus system!

   - contains 4 electrons and computed with 10 orbitals
   - set *occupied_indices = range(1)*
   - set *active_indices = range(1, molecule.n_orbitals)*
   - set *energy_threshold = 1e-2* in VQE.py
   - for a test, use the bond-length 1.42

   Beware, generating true energy of a system studied in the active space (without all of the electrons considered) is quite tricky with Openfermion, do not spend a lot of time on that if you have other things to do!

REMARK:
The H_2 and H_4 systems are the simplest one both in terms of constructing the ansatze and computationally. Instead the LiH and BeH_2 systems, will have simulations taking a LOT of time, keep that in mind!

2.2 Algorithmic and software development tasks

We are asking you to realize the following tasks, that comprised both algorithmic and software development aspects.
The tasks are ordered in terms of difficulty, either in conceptual sense of technical one.

2.2.1 Algorithm refactoring

The algorithm you will be provided is intentionally not working !

We are asking you to make the necessary modifications to fix all the problems we made. It either can be pure pythonic issues, qiskit issues and basic VQE workflow issues.
To be more precise, we want you to fix the scripts implementing the VQE, using the UCCSD excitation method, already implemented in the given code for the simplest molecular system: **the H_2 molecule**.

REMARK:
You are strongly advised to use the lecture slides and any documentation available for specific modifications (like the Qiskit documentation for example).

2.2.2 Scaling the algorithm

Once you have fixed the VQE algorithm with the test system H_2, we ask you to enable the algorithm to study more complex molecular systems, listed above. For that *some modifications shall be necessary* in the functions defining the **VQE workflow** (either about the quantum part of the classical optimization part). You are asked to find the right ones, perform them and successfully test them, preferably on the H_4 and LiH molecular systems.

Remark:
It is highly recommended to use the lectures notes (i.e. slides) for that purpose! Also, do not hesitate, if needed, to use the references given in the related bibliography.

2.2.3 Changing the ansätz

From the previous task, you may have understand that the simulations can be very long, and the convergences are quite difficult, maybe not even possible (even by changing the energy_threshold).
As an attempt to partially fixing these issues, we are asking you to change the fermionic UCCSD ansatz in the QEB UCCSD ansatz, explained in the lectures. To do so, you will have to create a new ansatz (adapt from the already exisiting one !) that are implementing such excitations.
Test your modifications on the H_2 and frozen-core LiH molecules.

Remark:
For that purpose, the use of the lectures notes (i.e. slides) is obviously mandatory!

2.2.4 Naively adding noise in the simulation

Currently in the implementation you shall not have used the number of shots for the simulation. You are thus performing an exact computation of the expectation values!
However the real QPUs, especially in the NISQ era, are very noisy. Hence, we are asking you to make a quick modification to the computation of the expectation values to incorporate non-exact results, through the *shots* variable.

Compare how it impacts the simulation for the test system H_2 in terms of iterations and energy results.

2.2.5 Generating important results

After all of the modifications and related tests that you have performed, we want you to produce several specific result files, as the ones we are commonly

doing in our work at Qubit Pharmaceuticals.
More precisely, we are asking you to provide additional modifications to the scripts in order to produce the following wanted results :

1. **H_2 molecule with fermionic UCCSD ansätz**: exact Potential Energy Surface (PES) curve
   - run the VQE algorithm for the following bond-lengths : from 0.24 to 3.00 by incrementing with a step of 0.1
   - plot the FCI energy and VQE energy curves for all of these bond-lengths in the same representation !
2. **H_2 molecule with QEB UCCSD ansätz**: exact Potential Energy Surface (PES) curve
   - run the VQE algorithm for the following bond-lengths : from 0.24 to 3.00 by incrementing with a step of 0.1
   - plot the FCI energy, fermionic UCCSD VQE energy and QEB UCCSD VQE energy curves for all of these bond-lengths in the same representation !
3. **H_2 molecule with UCCSD ansatze**: noisy Potential Energy Surface (PES) curve
   - add noise (see section 2.2.4) to the simulation
   - run the VQE algorithm for the following bond-lengths : from 0.24 to 3.00 by incrementing with a step of 0.1
   - plot the exact fermionic UCCSD VQE energy and noisy fermionic UCCSD VQE energy curves for all of these bond-lengths in the same representation !
   - plot the exact QEB UCCSD VQE energy and noisy fermionic UCCSD VQE energy curves for all of these bond-lengths in the same representation !
4. **H_4 molecule with fermionic UCCSD ansätz**: exact Potential Energy Surface (PES) curve
   - run the VQE algorithm for the following bond-lengths : from 0.24 to 3.00 by incrementing with a step of 0.2
   - plot the FCI energy and VQE energy curves for all of these bond-lengths in the same representation !
5. **LiH molecule with fermionic UCCSD ansätz**: exact Potential Energy Surface (PES) curve
   - run the VQE algorithm for the following bond-lengths : 0.54, 0.74, 1.42, 2.00

- store the FCI and VQE energies in a .csv file, and add a the relative energy difference

6. **BeH$_2$ molecule with fermionic UCCSD ansätz**: exact Potential Energy Surface (PES) curve
    - run the VQE algorithm for the following bond-length : 1.11
    - store the FCI and VQE energies in a .csv file, and add a the relative energy difference

REMARK:
We recall that for the LiH and BeH$_2$ systems, the simulations can take a LOT of time, keep that in mind!

2.2.6 ADAPTE-VQE implementation

As already pointed in the lectures, VQE is suffering from several issues that make its implementation currently not practical on NISQ devices and coming with a (exponentially) heavy classical computational cost.
As studied, the ADAPT-VQE is a successful modification that are tackling some of these issues! Hence, we are asking you to implement the mandatory modifications for having an adaptive workflow for the VQE algorithm.

In more details, you will need to implement, in order:

1. a function to compute the gradient of the energy expectation value with respect to the excitation parameter
2. a modification to the compose_ansatz function in the VQE.py script, to not anymore add all excitation operators at once, but one each time to respect the adaptive construction
3. a new workflow function, named adapt-vqe and replacing the vqe function, in the VQE.py script. This adapt-vqe function shall implement the workflow of the ADAPT-VQE algorithm presented in the slide 80 of the second part of the lectures
4. a possible update to the vqe_step function, in order to work perfectly with the ADAPT-VQE workflow
5. a stopping criteria not only based on the energy convergence

If you have been successful with the ADAPT-VQE implementation, we would like you to provide us some results for the H$_2$ and LiH molecules, as you should have done previously