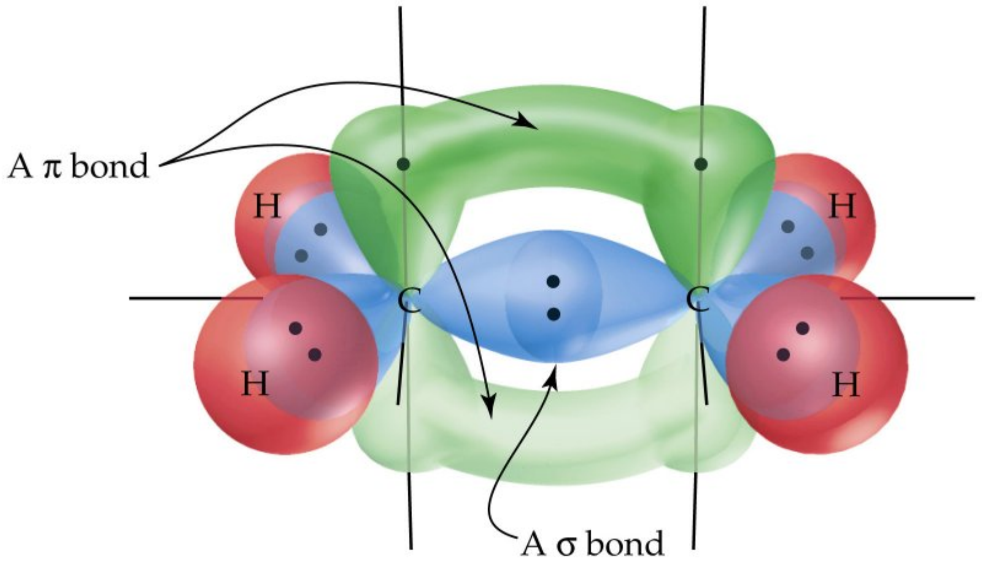
VARIATIONAL QUANTUM EIGENSOLVER
(and its usage in quantum chemistry)

February 2023,

Michailidis Stergios
2020030080

Introduction



A lot of modern problems in mathematics, chemistry and physics come down to finding the minimal eigenvalue of a given matrix. For example in chemistry, the energy of a molecular structure can be expressed as a hermitian matrix, the Hamiltonian. In this case calculating the minimal eigenvalue of the Hamiltonian – or at least estimating it – is the same as calculating the ground energy state of the molecular system. The ground energy state is very important, because it contains information about the properties of the molecule. Examples include its behavior in low temperatures, its magnetic properties, its geometric structure, its equilibrium state and many more that concern spin liquid dynamics, condense matter physics and/or quantum chemistry. So it is important that we can compute the minimum eigenvalues of the Hamiltonian, in an efficient enough way.

The initial problem:

Solving the time independent Schrödinger equation, we have:

$$\langle \Psi(\theta) | \hat{H} | \Psi(\theta) \rangle \geq E_o \equiv \lambda_{min}$$

So the expectation value is bounded by the ground state energy.

In order to find the ground state, we need to find the parameters θ that correspond to the ground state energy.

Hamiltonian of a molecule:

The Hamiltonian of a molecule is the sum of the kinetic energy $\hat{T}$ and potential energy $\hat{V}$

Suppose we have n number of nuclei (protons + neutrons) and e number of electrons.
Let's consider the potential energy first:

$$\hat{V} = \sum_{i,j}^{e} \frac{e^2}{4\pi\varepsilon_0 |r_i - r_j|} + \sum_{i,j}^{n} \frac{Z_i Z_j e^2}{4\pi\varepsilon_0 |r_i - r_j|} - \sum_{i}^{e} \sum_{j}^{n} \frac{Z_j e^2}{4\pi\varepsilon_0 |r_i - r_j|}$$

where the first term refers to the repulsion between electrons, the second terms to the repulsion between the cores and the third term to the attraction between the protons and the electrons. ε_0 refers to the vacuum permittivity constant , the $|r_i - r_j|$ refers to the distance between the particles and Z refers to the atomic number of the atom.

Now consider the kinetic energy:

$$\hat{T} = -\sum_{i}^{n} \frac{\hbar^2}{2m_i} \nabla_i^2 - \sum_{j}^{e} \frac{\hbar^2}{2m_e} \nabla_j^2$$

Where $\hbar$ is h-bar, m is the mass of the particle, and ∇^2 the Laplacian operator.

The sum of $\hat{T}$ and $\hat{V}$ gives us the Hamiltonian. We can now make some assumptions/approximations. For example, let's consider the Born-Oppenheimer approximation. This approximation makes 2 assumptions:

First, it treats the repulsion between the cores of the atoms as a constant. Therefore, the second term of the potential energy equation can be set to a real number C:

$$\hat{V} = \sum_{i,j}^{e} \frac{e^2}{4\pi\varepsilon_0 |r_i - r_j|} + C - \sum_{i}^{e} \sum_{j}^{n} \frac{Z_j e^2}{4\pi\varepsilon_0 |r_i - r_j|}$$

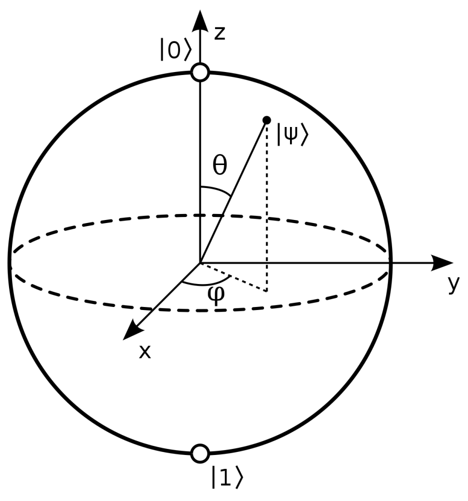
The second assumption is that the cores are stationary relative to the electrons. This renders the first term of the kinetic energy equal to zero:

$$\hat{T} = -0 - \sum_{j}^{e} \frac{\hbar^2}{2m_e} \nabla_j^2 = -\sum_{j}^{e} \frac{\hbar^2}{2m_e} \nabla_j^2$$

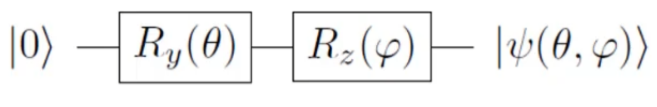
Now the Hamiltonian can be represented by a simpler matrix.
And so the “many electron” Hamiltonian can be mapped to a “many qubit” Hamiltonian.
One such mapping transformation is the Jordan-Wigner transformation.

Parameterized Quantum States:

The simplest parameterized quantum state we have seen is for one qubit , and it is the Bloch sphere representation of the qubits’ Hilbert space:

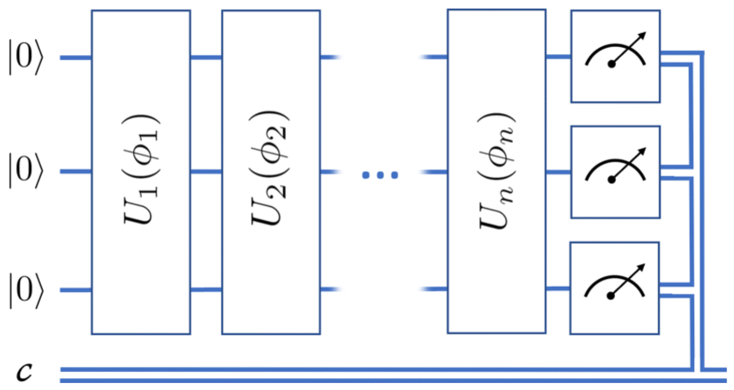


The parameterized state can be written as a circuit:



It is important to note that the gate type is fixed, while the parameter values vary.

This can be extended to multiple qubit states. That way we end up with an architecture which is fixed and it prepares a quantum state depending on the initial parameters:



So we need to represent both the Hamiltonian and the Parameterized state in terms of qubits.
This process is called mapping.

There are many ways to map the real quantum wave function of the system to a parameterized state, such as the *Slater determinant* for multiple particles and the *Jordan–Wigner transformation for the Hamiltonian.*

The Slater determinant, implies that the wave function is equal to a normalizing constant multiplied by the determinant of the matrix containing all the orbitals that contain the electrons. The orbitals contain information about the spin and the position, following the Pauli exclusion principle.

$$\Psi(x_1, x_2, \ldots, x_N) = \frac{1}{\sqrt{N!}} \begin{vmatrix} \chi_1(x_1) & \chi_2(x_1) & \ldots & \chi_N(x_1) \\ \chi_1(x_2) & \chi_2(x_2) & \ldots & \chi_N(x_2) \\ . & . & \cdots & . \\ . & . & \cdots & . \\ . & . & \cdots & . \\ \chi_1(x_N) & \chi_2(x_N) & \cdots & \chi_n(x_N) \end{vmatrix}$$

this represents the list of orbitals that are filled.

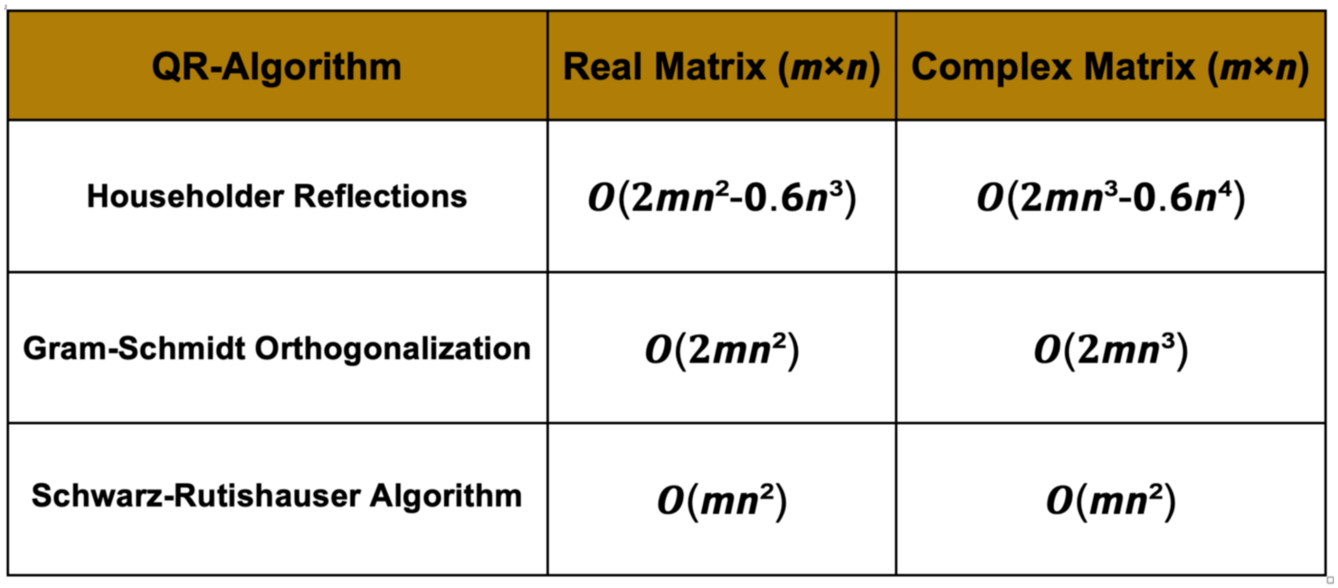
Now we need a more quantum computer friendly representation.
One way of achieving this is by -again- mapping this parameterized state and the simplified electron Hamiltonian using *second-Quantization*, also known as *occupation number representation*.

More information can be found here:
https://phys.libretexts.org/Bookshelves/Quantum_Mechanics/Quantum_Mechanics_III_(Chong)/04:_Identical_Particles/4.03:_Second_Quantization

We have finally expressed the Hamiltonian and the quantum parameterized state as mathematical objects that can be implemented by real world hardware.

Classical Algorithms

There have been a lot of classical algorithms tackling the issue of computing minimum eigenvalues. Notably, the QR algorithm presented by John G. F. Francis and Vera N. Kublanovskaya (working independently) is one of the fastest, with time complexity:



QR-Algorithm	Real Matrix ($m \times n$)	Complex Matrix ($m \times n$)
Householder Reflections	$O(2mn^2\text{-}0.6n^3)$	$O(2mn^3\text{-}0.6n^4)$
Gram-Schmidt Orthogonalization	$O(2mn^2)$	$O(2mn^3)$
Schwarz-Rutishauser Algorithm	$O(mn^2)$	$O(mn^2)$

(image taken from https://towardsdatascience.com/can-qr-decomposition-be-actually-faster-schwarz-rutishauser-algorithm-a32c0cde8b9b)

For our purposes, consider m = n for we have hermitian matrices to deal with.
So in the worst case, using the Schwarz-Rutishauser version of the algorithm we can achieve a $O(n^3)$ time complexity.
Without going into too much detail , the algorithm seems relatively fast! So why are we in need of a quantum algorithm?

Consider the Hamiltonian of a molecular structure. The problem of finding the minimum eigenvalue scales with the dimension of the Hilbert space and the dimension of the Hilbert space itself scales with the number of atoms/electrons:

We can write the Hamiltonian as a linear combination of the pauli matrices and the unitary matrix.
The general form of an n – qubit Hamiltonian is the following:

$$\widehat{H} = \sum_{i\alpha} h^i_\alpha \hat{\sigma}^i_\alpha + \sum_{ij\alpha\beta} h^{ij}_{\alpha\beta} \hat{\sigma}^i_\alpha \otimes \hat{\sigma}^j_\beta + \ldots$$

where *h* are real coefficients and $\hat{\sigma}^{ijk\ldots}_{\alpha\beta\gamma\ldots}$ are the generalization of the Pauli matrices .

Due to linearity, the expectation value of the Hamiltonian is:

$$\langle \widehat{H} \rangle = \sum_{i\alpha} h^i_\alpha \langle \hat{\sigma}^i_\alpha \rangle + \sum_{ij\alpha\beta} h^{ij}_{\alpha\beta} \langle \hat{\sigma}^i_\alpha \otimes \hat{\sigma}^j_\beta \rangle + \ldots$$

Adding more atoms, increases the dimension of the Hamiltonian exponentially.
The more atoms present, the harder it becomes for classical hardware to simulate the relations between them. It is very difficult for a classical computer to simulate molecules that are composed of more that 2-3 atoms.

Using a quantum computer ,we can evaluate the expectation value of a $2^n \times 2^n$ Hamiltonian using n number of qubits.

The variational quantum eigensolver

- *What is the VQE?*

The variational quantum eigensolver (VQE) is a quantum algorithm designed to find the ground state energy of a given Hamiltonian using a quantum computer.

- *How does it work?*

It does this by preparing a trial wave-function (Ansatz) on the quantum computer, measuring its energy, and updating the parameters of the wave-function using classical optimization techniques. The process is repeated until the energy converges to a minimum, which corresponds to the ground state energy of the Hamiltonian.

The VQE is a hybrid algorithm, making use of quantum and classical subroutines.
The Quantum part is responsible for measuring the expectation value of the energy of some parameterized state, whereas the classical part is responsible for altering the parameters to minimize said energy.

It can be broken down into the following steps:

1) Make an educated guess, called ansatz which can be prepared through the parameterized quantum circuit.

2) Implement the Ansatz on a quantum computer to prepare the trial wave function.

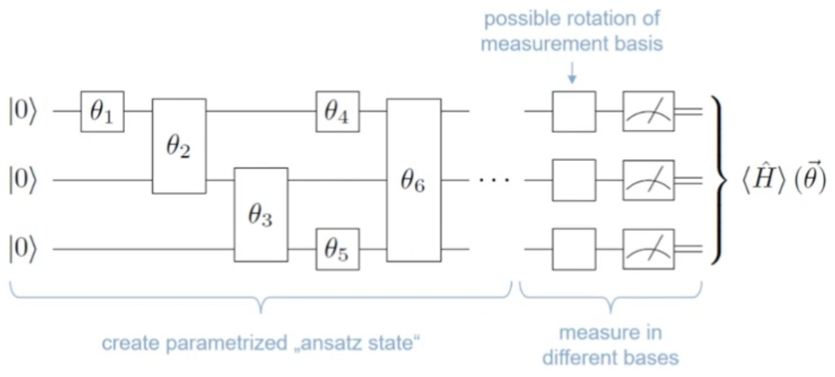
3) Measure the energy of the trial wave function by performing measurements on the quantum computer.

4) Use classical optimization techniques to adjust the parameters of the ansatz to minimize the energy.

5) Repeat steps 2-4 until the energy converges to a minimum, which corresponds to the ground state energy of the Hamiltonian.

Once we have found the ground state energy of the Hamiltonian, we can use it to extract useful information about the molecule, such as its geometry and electronic structure.

-*Reiterating*:
Start with an ansatz state of parameters θ.
The quantum subroutine used measures the energy expectation value of the Hamiltonian of the ansatz parameterized state:



possible rotation of measurement basis
$|0\rangle$
$|0\rangle$
$|0\rangle$
θ_1
θ_2
θ_3
θ_4
θ_5
θ_6
$\langle \hat{H} \rangle (\vec{\theta})$
create parametrized „ansatz state"
measure in different bases

Then, we parse this information to the classical subroutine of the algorithm, which shifts the parameters θ.

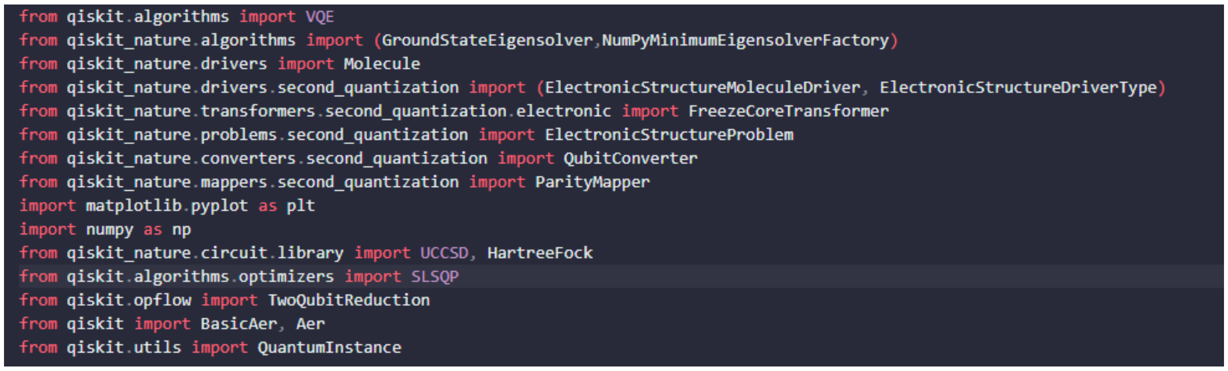
After that, we execute the quantum subroutine again and get new measurements, which feeds into the classical subroutine again to shift the the parameters.

Executing this enough times, should make the energy expectation value approach E_0 , being a good enough result.

To find the exact solution, we would have to explore all possible solutions of the n-qubit Hilbert space and choose the one that corresponds to the lowest energy. This is the brute force method which fails in a system with more than 2-3 atoms.

Qiskit implementation:

Firstly, import some libraries.

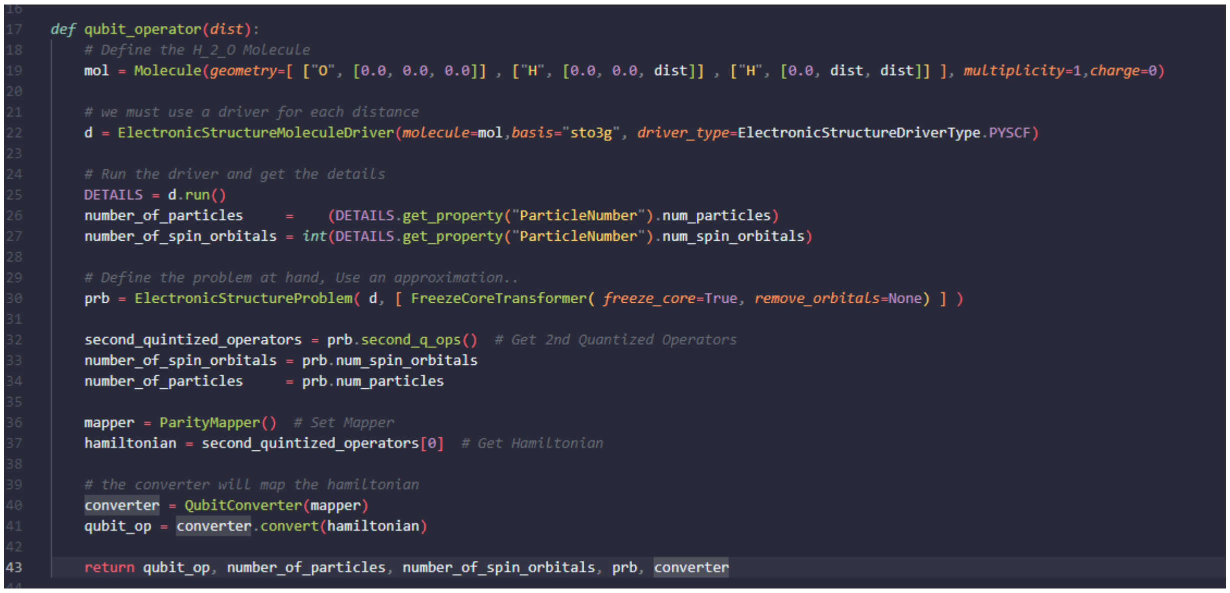


```
from qiskit.algorithms import VQE
from qiskit_nature.algorithms import (GroundStateEigensolver,NumPyMinimumEigensolverFactory)
from qiskit_nature.drivers import Molecule
from qiskit_nature.drivers.second_quantization import (ElectronicStructureMoleculeDriver, ElectronicStructureDriverType)
from qiskit_nature.transformers.second_quantization.electronic import FreezeCoreTransformer
from qiskit_nature.problems.second_quantization import ElectronicStructureProblem
from qiskit_nature.converters.second_quantization import QubitConverter
from qiskit_nature.mappers.second_quantization import ParityMapper
import matplotlib.pyplot as plt
import numpy as np
from qiskit_nature.circuit.library import UCCSD, HartreeFock
from qiskit.algorithms.optimizers import SLSQP
from qiskit.opflow import TwoQubitReduction
from qiskit import BasicAer, Aer
from qiskit.utils import QuantumInstance
```

For the setup, we need our molecule and the driver to execute the experiment for each distance.
In this case, we will simulate a molecule of water H_2O .

Let’s define the function which returns the qubit operator with respect to distance:
Suppose the oxygen atom is stationary, the first hydrogen atom has variable Z coordinate and the last hydrogen atom has variable Z & Y coordinates.

The basis that we will run the experiment will be the **SDG** basis .



```python
def qubit_operator(dist):
    # Define the H_2_O Molecule
    mol = Molecule(geometry=[ ["O", [0.0, 0.0, 0.0]] , ["H", [0.0, 0.0, dist]] , ["H", [0.0, dist, dist]] ], multiplicity=1,charge=0)

    # we must use a driver for each distance
    d = ElectronicStructureMoleculeDriver(molecule=mol,basis="sto3g", driver_type=ElectronicStructureDriverType.PYSCF)

    # Run the driver and get the details
    DETAILS = d.run()
    number_of_particles     =    (DETAILS.get_property("ParticleNumber").num_particles)
    number_of_spin_orbitals = int(DETAILS.get_property("ParticleNumber").num_spin_orbitals)

    # Define the problem at hand, Use an approximation..
    prb = ElectronicStructureProblem( d, [ FreezeCoreTransformer( freeze_core=True, remove_orbitals=None) ] )

    second_quintized_operators = prb.second_q_ops()  # Get 2nd Quantized Operators
    number_of_spin_orbitals = prb.num_spin_orbitals
    number_of_particles     = prb.num_particles

    mapper = ParityMapper()  # Set Mapper
    hamiltonian = second_quintized_operators[0]  # Get Hamiltonian

    # the converter will map the hamiltonian
    converter = QubitConverter(mapper)
    qubit_op = converter.convert(hamiltonian)

    return qubit_op, number_of_particles, number_of_spin_orbitals, prb, converter
```

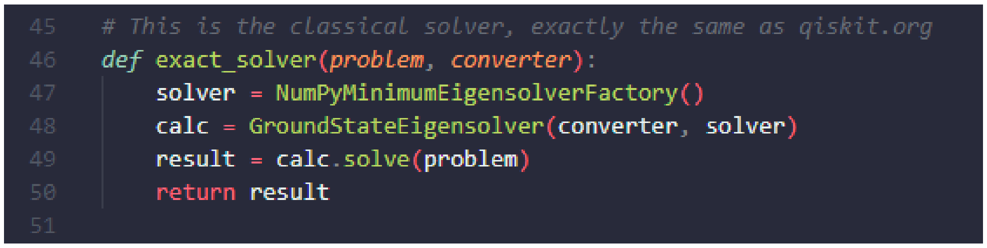
Defining the driver, parse inputs 1.the molecule of the experiment
2.the basis of the experiment
3. the driver type
When we execute the driver with driver.run() the function will return some values (DETAILS).
These values being the number of particles , and the number of spin orbitals of the system.

The problem is an electronics structure problem, defined in the qiskit-nature libraries.
We also use the frozen core approximation.

The next step is to transform the problem to a 2cond quantization problem.
This is done by calling the convert function of the converter object.

Finally, the return values of the function are the converted qubit operator, the number of particles , the number of spin orbitals the electronics structure problem object and the converter object.

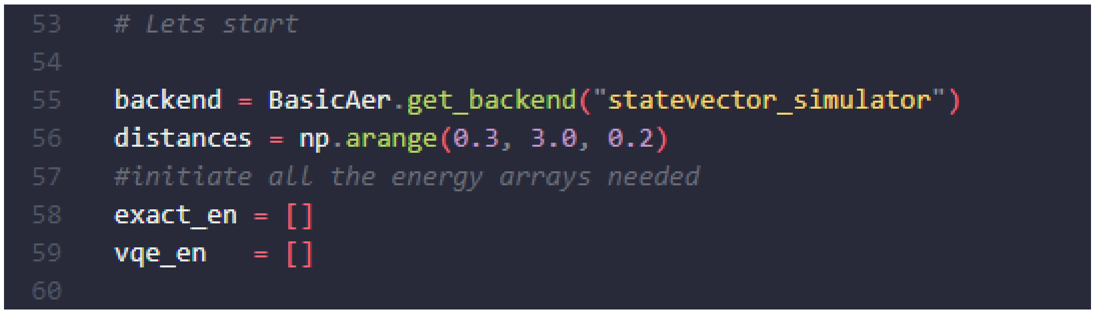
<u>Following this we can define a classical solver (exact same as qiskit.org):</u>



```python
# This is the classical solver, exactly the same as qiskit.org
def exact_solver(problem, converter):
    solver = NumPyMinimumEigensolverFactory()
    calc = GroundStateEigensolver(converter, solver)
    result = calc.solve(problem)
    return result
```

Now we are ready to start the algorithm.

Initiate the simulator and some array fields:



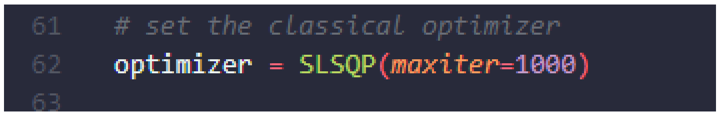
```python
# Lets start

backend = BasicAer.get_backend("statevector_simulator")
distances = np.arange(0.3, 3.0, 0.2)
#initiate all the energy arrays needed
exact_en = []
vqe_en   = []

```

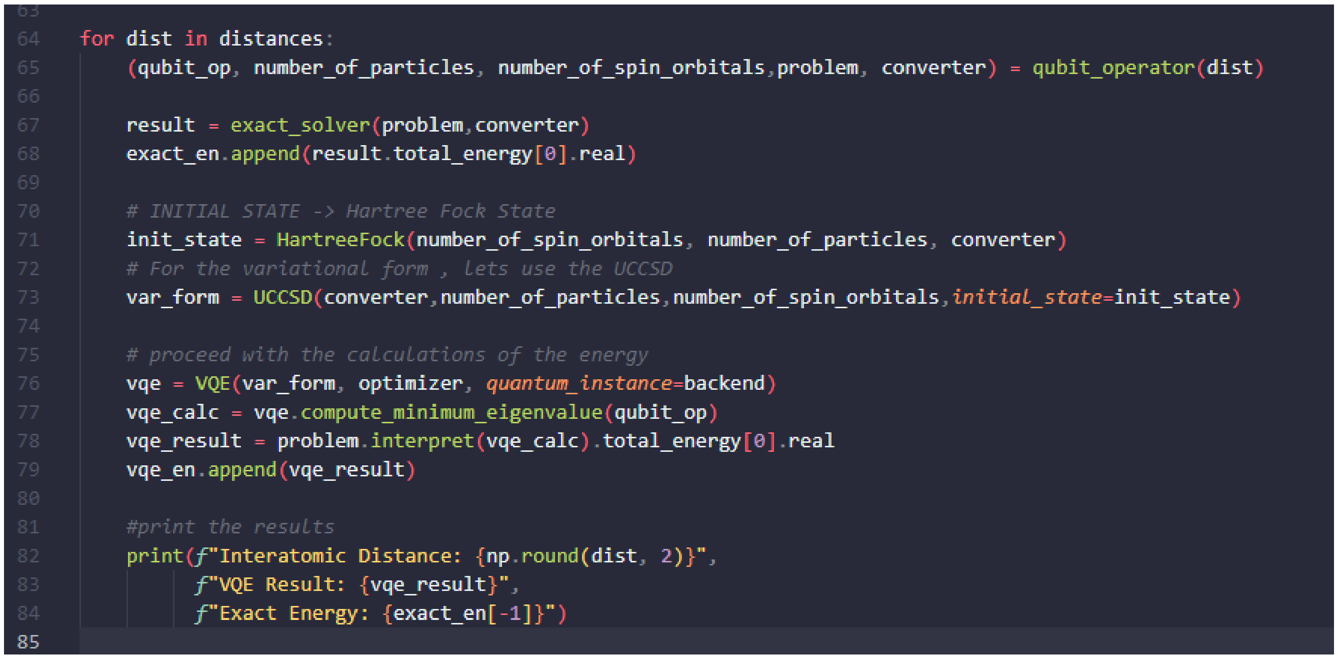
The distances are calculated in Angstrom $10^{-10} m$. From 0.3 angstrom to 3.0 angstrom with intervals of 0.2 angstrom.
The field exact_en is for the energy found by the classical algorithm, whereas the vqe_en field is the energy calculated by the VQE.

Next, let’s select a classical optimizer. We have some choices: `COBYLA, SPSA, SLSQP.`
Using the SLSQP with 1000 iterations:



```python
# set the classical optimizer
optimizer = SLSQP(maxiter=1000)
```

Now let’s code the loop:



```python
for dist in distances:
    (qubit_op, number_of_particles, number_of_spin_orbitals,problem, converter) = qubit_operator(dist)

    result = exact_solver(problem,converter)
    exact_en.append(result.total_energy[0].real)

    # INITIAL STATE -> Hartree Fock State
    init_state = HartreeFock(number_of_spin_orbitals, number_of_particles, converter)
    # For the variational form , lets use the UCCSD
    var_form = UCCSD(converter,number_of_particles,number_of_spin_orbitals,initial_state=init_state)

    # proceed with the calculations of the energy
    vqe = VQE(var_form, optimizer, quantum_instance=backend)
    vqe_calc = vqe.compute_minimum_eigenvalue(qubit_op)
    vqe_result = problem.interpret(vqe_calc).total_energy[0].real
    vqe_en.append(vqe_result)

    #print the results
    print(f"Interatomic Distance: {np.round(dist, 2)}",
          f"VQE Result: {vqe_result}",
          f"Exact Energy: {exact_en[-1]}")

```

In the beginning we need to find the qubit operator for the specific distance. This yields an array of results.

The next thing to dois to run the classical solver and get the result. Parse the result into the exact_en field.

Now let’s define our Ansatz state. Using the Hartree – Fock state we get our initial state.
The variational form of the state is now needed to proceed. Let’s use the UCCSD class with input the Ansatz state, and the other information from the qubit operator.

Next , initiate the VQE object. After that calculate the minimum eigenvalue using its method, keeping only the real solution.

Append the newly calculated energy to the vqe_result array.

Lastly , lets print all the information.

Now the loop will repeat these steps for all different distances.

Referral links:

The original paper by Dr. Alberto Peruzzo:
https://www.nature.com/articles/ncomms5213

Information from Qiskit.org:
https://qiskit.org/textbook/ch-applications/vqe-molecules.html

Importance of computing minimum eigenvalues in other fields:
https://physics.stackexchange.com/questions/591584/why-is-the-ground-state-important-in-condensed-matter-physics

Classical algorithms for eigenvalue computation:
https://towardsdatascience.com/can-qr-decomposition-be-actually-faster-schwarz-rutishauser-algorithm-a32c0cde8b9b
(from LULEA university of technology)

(and wikipedia:)
https://en.wikipedia.org/wiki/Eigenvalue_algorithm
https://en.wikipedia.org/wiki/QR_algorithm

Hamiltonian:
https://en.wikipedia.org/wiki/Hamiltonian_(quantum_mechanics)

Slater determinant:
https://en.wikipedia.org/wiki/Slater_determinant

Laplacian operator:
https://en.wikipedia.org/wiki/Laplace_operator