NExt ApplicationS of Quantum Computing

D4.2 – QCCC alpha

Document Properties

Contract Number	951821
Contractual Deadline	28-2-2022
Dissemination Level	Public
Nature	Report
Editors	Wassil Sennane, TotalEnergies Marko J. Rančić, TotalEnergies
Authors	Wassil Sennane, TotalEnergies Marko J. Rančić, TotalEnergies
Reviewers	Jan Reiner, HQS Arseny Kovyrshin, AstraZeneca
Date	28-Feb-2022
Keywords	Quantum computing, benzene, post-Hartree-Fock, VQE, quantum algorithms, quantuum computing for quantum chemistry
Reviewed	23.2.2022
Alpha	

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 951821

History of Changes

Release	Date	Author, Organisation	Description of Changes
0.1	24/02/2022	Wassil Sennane, TotalEnergies	A long summary
0.2	26/02/2022	Marko J. Rančić, TotalEnergies	Form adaptation
1.0	28/02/2022	Marko J. Rančić, TotalEnergies	Final form with introduction and executive summary
1.1	02/03/2022	Wassil Sennane, TotalEnergies	Including Github link and corrections from Arseny Kovyrshin

Table of Contents

1 Executive Summary

The code presented in this document allows for the calculation of the ground state energy of benzene under spatial deformations by using a state-of-the-art quantum computing methodology - the variational quantum eigensolver (VQE). Two types of quantum computing ansatze are implemented (the hardware efficient one and the qUCC). The code supports noisy simulations and three types of spatial deformations of the benzene molecule. The code is available on Github : https://github.com/NEASQC/D4.2.

2 Introduction

Quantum computing opens a new era of calculations thanks to quantum superposition and quantum entanglement. While classical computers handle binary information, quantum computers use entangled superposition of states as information carriers. Some algorithms, such as the well-known Grover algorithm and Shor's algorithm, will bring new efficient ways of solving complex problems if implemented on quantum computers. Moreover, the deepest understanding of complex molecules will become possible with new techniques such as the Quantum Phase Estimation (QPE) algorithm and the Variational Quantum Eigensolver algorithm (VQE). This new technology uses very fragile entangled states of the matter, which makes the management of multiple qubits without error correction difficult. As QPE requires quantum error correction we focus solely on the NISQ-friendly VQE.

There are in principle two approaches in which quantum computing is foreseen to bring value: through development of quantum inspired algorithms which are executed on quantum simulators and through algorithms which are executed on actual quantum hardware. Quantum simulators exploit advanced supercomputing platforms to emulate quantum-computing like environments. On the other hand the bottleneck in executing algorithms on actual hardware is quantum noise processes: qubit dephasing, qubit relaxation and readout errors. With two qubit error rates on the order of 1% the maximal number of two qubit gates applied in a circuit is a few hundred.

An open question remains: could any of these two approaches bring immediate value in treating problems beyond for instance very simple molecules? With this software deliverable we try to address these questions by developing a solver for benzene, a cyclic molecule with the formula C_6H_6.

In order to minimize the number of qubits required for a computation, we chose to work with the least computationally demanding basis - the sto-3g. In this basis, each H is represented with a $1s$ orbital, and each C is represented with $1s, 2s, 2p_x, 2p_y, 2p_z$. Therefore, the treatment of the entire molecule would require taking into account 36 orbitals, equivalently 72 spin orbitals for which 72 qubits would be required. Because simulating 72 qubits is beyond the simulation capabilities of our in-house Atos simulator, capable of simulating up to 35 qubits, our approach relies on a reduction of the size of the system via active space selection. Intensive numerical testing showed that those active spaces with an equal number of HOMO and LUMO orbitals lead to lower energies as opposed to active spaces with unequal numbers of HOMO and LUMO orbitals.

Figure 1 describes one of the three possible distorsions. All the distortions studied here have the same spacing between one carbon and its closest neighboring hydrogen atom. This distance is fixed at 1.09 Å as in the equilibrium conformation of benzene. Generally, all fixed parameters have their equilibrium value. The first distortion is a uniform deformation of the molecule (Figure 1 left), in which the distance between two neighboring carbon atoms R_1 varies. In the second distortion, the variable parameter is the distance R_2 between two opposite sides of the hexagon (Figure 1 center). The distance between the carbon atoms which belong to each of these sides does not vary. Moreover, the two carbon atoms which do not belong to one of these sides have their spacing fixed. During the distortion, these two carbon atoms are vertically halfway of the two sides. For the third distortion, the idea is to divide the benzene into 2 identical triplets of carbon-hydrogen pairs. Then, these two parts are laterally moved one from another by varying the parameter R_3 (Figure 1 right).

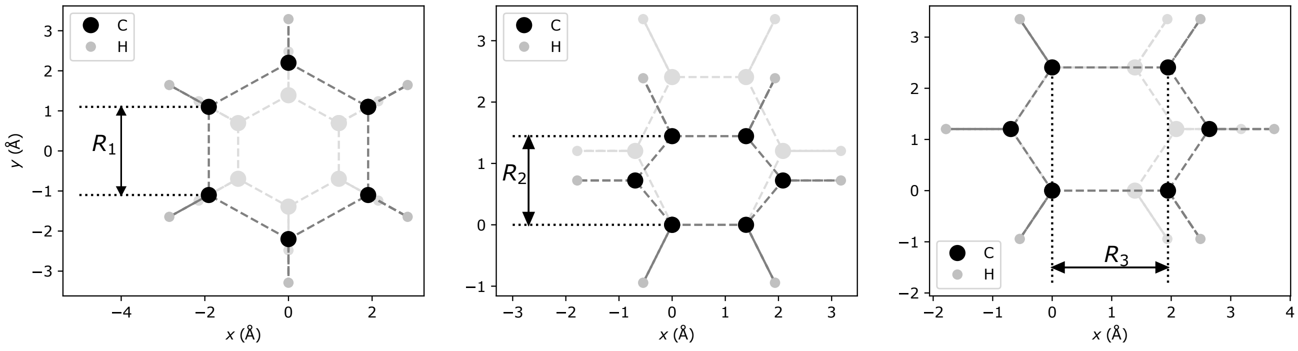
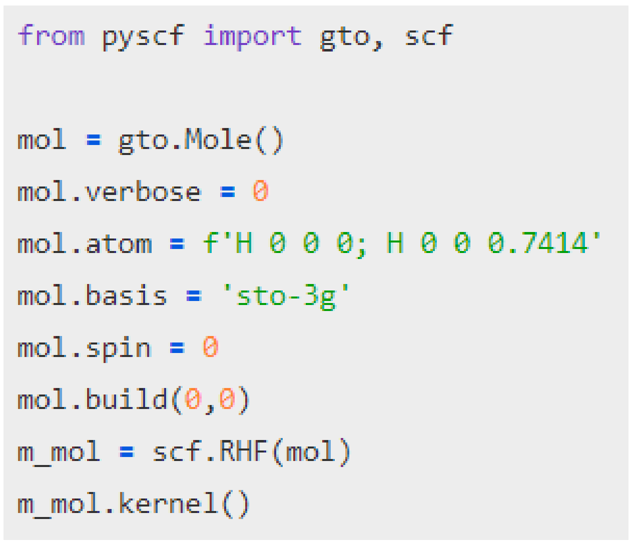


Figure 1: The 3 types of distortions studied.

3 Documentation

This part briefly describes the github codes and their content. **mol** refers to a PySCF object built from a molecule, and **m_mol** is its mean field. For example, in the H2 molecule case :



```python
from pyscf import gto, scf

mol = gto.Mole()
mol.verbose = 0
mol.atom = f'H 0 0 0; H 0 0 0.7414'
mol.basis = 'sto-3g'
mol.spin = 0
mol.build(0,0)
m_mol = scf.RHF(mol)
m_mol.kernel()
```

Figure 2: *How to build a PySCF object*

3.1 Github_calc_Hamilt.py

This section details the functions in ***Github_calc_Hamilt.py***.

ob_tb_integ(mol, m_mol) : function which takes as input a PySCF molecule and its mean field and returns the one-body and two-body integrals.

H_with_active_space_reduction(one_body_integ, two_body_integ, mol, m_mol, nb_homo, nb_lomo): : This function needs as input the one-body integrals, the two-body integrals, the PySCF object with the mean field of a molecule and the number of homo and lomo. The function returns :

- **H_active** : ElectronicStructureHamiltonian object containing the Hamiltonian of the molecule after active space reduction.
- **active_inds** : List that contains the indices of active orbitals
- **occ_inds** : List that contains the indices of occupied (frozen) orbitals
- **noons** : List of natural orbital occupation numbers of the molecule, computed with CISD method.
- **orbital_energies** : list of energies of each orbital
- **nels** : total number of electrons

save_H_into_dict(l1, save_filename, mol, m_mol, nb_homo, nb_lomo, calc_E_exact) : The purpose of this function is, by using **ob_tb_integ** and **H_with_active_space_reduction**, to build and save directly the Hamiltonian of a molecule in a file. As the final goal is to obtain ground state energy curves, this function takes as input a varying parameter **l1**, the name of the saved file, the PySCF object with its mean field and the number of homo and lomo. **calc_E_exact** is False by default : a True value implies the calculation of the exact lowest eigenvalue of the Hamiltonian with a numerical diagonalization. This function returns :

- **dic_H_save** : dictionary, saved in *save_filename.H.pickle* with :
  - 1st key : varying parameter **l1** (e.g. : bond length of the molecule)

- 2nd key : chemical basis set
- 3rd key : **nb_homo**
- 4th key : **nb_lomo**
- Then :
  * **H_active** : ElectronicStructureHamiltonian object containing the Hamiltonian of the molecule after active space reduction.
  * **active_inds** : List that contains the indices of active orbitals
  * **occ_inds** : List that contains the indices of occupied (frozen) orbitals
  * **noons** : List of natural orbital occupation numbers of the molecule, computed with CISD method.
  * **orbital_energies** : list of energies of each orbital
  * **nels** : total number of electrons

display_full_hamilt(dic_H) : This function takes as input a dictionary of Hamiltonians built by **save_H_into_dict** and displays all the computations done.

build_benz_dist_1(alpha, basis) : This function builds a benzene molecule under distortion 1 (cf publication), with **alpha** as varying parameter and **basis** as chemical basis set. The function returns a PySCF object and its mean field.

build_benz_dist_2(alpha, basis) : This function builds a benzene molecule under distortion 2 (cf publication), with **alpha** as varying parameter and **basis** as chemical basis set. The function returns a PySCF object and its mean field.

build_benz_dist_3(alpha, basis) : This function builds a benzene molecule under distortion 3 (cf publication), with **alpha** as varying parameter and **basis** as chemical basis set. The function returns a PySCF object and its mean field.

full_hamilt_computation(dist, alpha, basis, nb_homo, nb_lomo, calc_E_exact) : The purpose of this function is to compute the Hamiltonian of benzene under spatial deformations and save it into a file. This functions takes as input the distortion **dist** to apply, **alpha** the distortion parameter, **basis** the chemical basis set, and **nb_homo** and **nb_lomo** the number of homo and lomo. **calc_E_exact** is False by default : a True value implies the calculation of the exact lowest eigenvalue of the Hamiltonian with a numerical diagonalization. The function uses the previous functions to save the corresponding Hamiltonian in *benzene_dist*{**dist**}*.H.pickle*.

3.2 Github_calc_Energy.py

This section details the functions in ***Github_calc_Energy.py***.

HE_circuit_for_ansatz(theta, nbqbits) : function which takes as input a list of parameters (**theta**) and the number of qubits of the Hamiltonian one wants to estimate (**nbqbits**). The function returns **Qrout**, an object containing $|\psi(\theta)\rangle$ built with the Hardware-efficient method v5 (cf publication).

fun_HE_ansatz(H_active_sp, theta, nbshots, qpu) : This function needs as input a SpinHamiltonian object (**H_active_sp**), a list of parameters (**theta**), **nbshots** the number of shots for quantum measurement and **qpu** the quantum processing unit used. The function returns an estimation of $\langle\psi(\theta)|\,|\mathbf{H\_active\_sp}\,|\psi(\theta)\rangle$ with **nbshots** shots, and $|\psi(\theta)\rangle$ built with the Hardware-efficient method v5 (cf publication).

vqe_he_calc(H_active_sp, d, nbshots) : This function needs as input a SpinHamiltonian (**H_active_sp**), the depth of the quantum circuit **d** and **nbshots** the number of shots for quantum measurement. The function returns, with $|\psi(\theta)\rangle$ built with the Hardware-efficient method v5 (cf publication) with a depth **d** :

$$E = \min_{\theta} \langle\psi(\theta)|\,|\mathbf{H\_active\_sp}\,|\psi(\theta)\rangle$$

The initial guess of θ is chosen randomly. The optimizer is COBYLA, with a maximum of 1000 steps.

ucc_ansatz_calc(H_active, active_inds, occ_inds, noons, orbital_energies, nels) : This functions requires :

- **H_active** : ElectronicStructureHamiltonian object containing the Hamiltonian of the molecule after active space reduction.
- **active_inds** : List that contains the indices of active orbitals
- **occ_inds** : List that contains the indices of occupied (frozen) orbitals
- **noons** : List of natural orbital occupation numbers of the molecule, computed with CISD method.
- **orbital_energies** : list of energies of each orbital
- **nels** : total number of electrons

The function returns **H_active_sp** the Jordan-Wigner Hamiltonian, **qprog** an object containing $|\psi\theta\rangle$ built with the qUCC method, and **theta_0** the initial guess of parameters.

fun_qucc_ansatz(H_active_sp, qrout, theta, nbshots) : This function needs as input a SpinHamiltonian (**H_active_sp**), **qrout** the object containing $|\psi\rangle$ built with the qUCC method, **theta** a list of parameters and **nbshots** the number of shots for quantum measurement. The function returns an estimation of $\langle\psi(\theta)|\,|\mathbf{H\_active\_sp}\,|\psi(\theta)\rangle$ with **nbshots** shots, and $|\psi(\theta)\rangle$ built with the qUCC method.

vqe_ucc_calc(H_active_sp, qprog, theta_0, nbshots) : This function needs as input a SpinHamiltonian (**H_active_sp**), **qprog** the object containing $|\psi\rangle$ built with the qUCC method, **theta_0** the initial guess of parameters and **nbshots** the number of shots for quantum measurement. The function returns, with $|\psi(\theta)\rangle$ built with the qUCC method :

$$E = \min_{\theta} \langle\psi(\theta)|\,|\mathbf{H\_active\_sp}\,|\psi(\theta)\rangle$$

save_E_into_dict(l1, save_filename, mol, m_mol, nb_homo, nb_lomo, calc_E_exact) : The purpose of this function is to use VQE on Hamiltonian previously computed with **Github_calc_Hamilt.py**, and save the result in a new file. The function requires :

- **l1** : varying parameter
- **hamilt_filename** : filename that contains the Hamiltonian
- **save_filename** : filename for saving
- **mol**, **m_mol** : PySCF molecule with its mean field
- **nb_homo**, **nb_lomo** : number of homo and lomo
- **ansatz** : choice of the method to create the quantum circuit (qUCC or HE)
- **nbshots** : number of shots for quantum measurement
- **d** : depth or number of parametrized layers of the circuit (only for HE)
- **N_trials** : number of times one wants to repeat the VQE algorithm.

The function returns :

- **dic_E_save** : dictionary, saved in *save_filename.E.pickle* with :
  - 1st key : varying parameter **l1** (e.g. : bond length of the molecule)
  - 2nd key : chemical basis set
  - 3rd key : **nb_homo**
  - 4th key : **nb_lomo**
  - Then :
    - HF : Hartree-Fock energy.
    - VQE : VQE energies
      - qUCC $\rightarrow$ **nbshots** $\rightarrow$list with **N_trials** of VQE energies
      - HE $\rightarrow$ **nbshots** $\rightarrow$ d $\rightarrow$ list with **N_trials** of VQE energies

display_full_energies(dic_E) : This function takes as input a dictionary of energies built by **save_E_into_dict** and displays all the computations done. This function uses the function **display_results**.

build_benz_dist_1(alpha, basis) : This function builds a benzene molecule under distortion 1 (cf publication), with **alpha** as varying parameter and **basis** as chemical basis set. The function returns a PySCF object and its mean field.

build_benz_dist_2(alpha, basis) : This function builds a benzene molecule under distortion 2 (cf publication), with **alpha** as varying parameter and **basis** as chemical basis set. The function returns a PySCF object and its mean field.

build_benz_dist_3(alpha, basis) : This function builds a benzene molecule under distortion 3 (cf publication), with **alpha** as varying parameter and **basis** as chemical basis set. The function returns a PySCF object and its mean field.

full_energy_computation(dist, alpha, basis, nb_homo, nb_lomo, calc_E_exact) : The purpose of this function is to compute the ground state energy of benzene under spatial deformations and save it into a file. This functions takes as input the distortion **dist** to apply, **alpha** the distortion parameter, **basis** the chemical basis set, and **nb_homo** and **nb_lomo** the number of homo and lomo. **ansatz** is the method used to create the quantum circuit (qUCC or HE), **nbshots** is the number of shots for quantum measurement, **d** is the depth or number of parametrized layers of the circuit (only for HE) and **N_trials** is the number of times one wants to repeat the VQE algorithm. The function uses the previous functions to load the Hamiltonians in *benzene_dist*{**dist**}*.H.pickle*. and to save the corresponding dictionary of VQE energies in *benzene_dist*{**dist**}*.E.pickle*.

4 Examples

4.1 H2 molecule

Figure 3 shows how to obtain the ground state energy curve of H2 molecule with the previous functions.

```
from pyscf import gto, scf

l_R = list(np.linspace(0.2,2,10)) + [5]

for R_H2 in l_R:

    mol = gto.Mole()
    mol.atom = f'H 0 0 0; H 0 0 {R_H2}'
    mol.basis = 'sto-3g'
    mol.spin = 0
    mol.build(0,0)
    m_mol = scf.RHF(mol)
    m_mol.kernel()

    ###################################

    save_filename = 'test_github_neasqc.H2'
    nb_lumo = 1
    nb_homo = 1
    dic_H_save = save_H_into_dict(R_H2, save_filename, mol, m_mol, nb_homo, nb_lumo)

    ###################################

    hamilt_filename = 'test_github_neasqc.H2.H.pickle'
    save_filename = 'test_github_neasqc.H2'
    dic_E_save = save_E_into_dict(R_H2, hamilt_filename, save_filename, mol, m_mol,
                                  nb_homo, nb_lumo, ansatz="qUCC", nbshots=0, d=1, N_trials=1)

    dic_E_save = save_E_into_dict(R_H2, hamilt_filename, save_filename, mol, m_mol,
                                  nb_homo, nb_lumo, ansatz="HE", nbshots=0, d=1, N_trials=1)

    ###################################


l_HF, l_qUCC, l_HE = [], [], []
for R_H2 in dic_E_save:
    E_HF = dic_E_save[str(R_H2)]['sto-3g'][str(nb_homo)][str(nb_lumo)]['HF']
    l_HF.append(E_HF)
    E_qucc = dic_E_save[str(R_H2)]['sto-3g'][str(nb_homo)][str(nb_lumo)]['VQE']['qUCC']['0']
    l_qUCC.append(E_qucc)
    E_HE = dic_E_save[str(R_H2)]['sto-3g'][str(nb_homo)][str(nb_lumo)]['VQE']['HE']['0']['1']
    l_HE.append(E_HE)
```

Figure 3: How to obtain the ground state energy curve of H2 molecule with ***Github_calc_Hamilt.py*** *and* ***Github_calc_Energy.py***

4.2 Benzene molecule

Figure 4 shows results that can be obtained with our code. Our formalism allows us to obtain the ground state energy and particle number of benzene under spatial deformations (Fig. 3 (a)). Furthermore, by comparing Figure 4 (b) and (c) we see that our code managed to produce the correct behavior of the aromatic molecule - benzene has 6 p_z orbitals which are separated from the other deeper lying orbitals. Such orbitals in aromatic molecules suffices to describe the relevant physics and chemistry. Also the quantum computing methodology (Figure 4 (b)) is superior to MP2, CISD and CCSD (Figure 4 (d)) in predicting orbital occupancy - the molecule is further distorted according to the scheme in Figure 4 (a) 6 identical $C - H$ radicals should be created with equal orbital occupancy - quantum computing manages to capture this.

Finally, the energy curves of the quantum computing methodology remain physical with both the hardware efficient and the qUCC ansatz - most classical methodologies fail in describing the effect of distorsion. Classical benchmarks are obtained with PySCF.

This work is part of a paper in the process of submission.

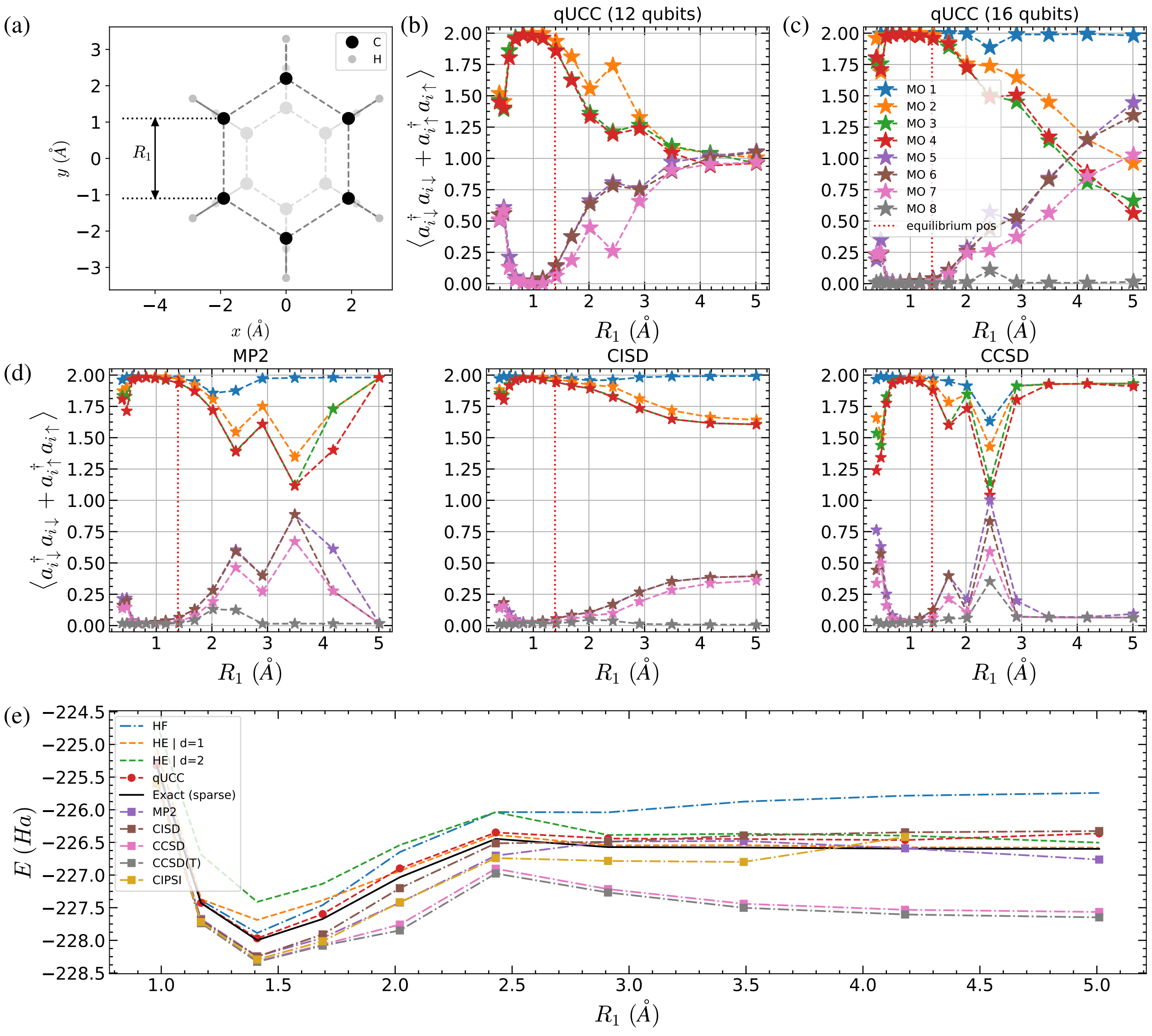


Figure 4: (a) Distortion of benzene, then orbital occupation number obtained with (b) 12 qubits and (c) 16 qubits qUCC method, and (d) estimating orbital occupation numbers with different classical methodologies. Finally (e) shows the energy obtained with different methodologies. The qUCC energy results are performed with orbital freezing with 16 qubits. The "Exact (sparse)" curve is obtained with diagonalizing 16 qubits Hamiltonians.

List of Figures