Quantum computing TP
Phase estimation for chemistry

Supervisor: Bertrand Marchand

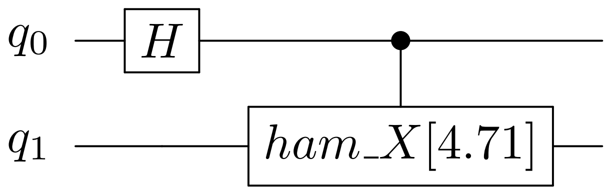
January 2, 2021

Abstract

You need to fill up the helper file and some notebook cells according to the questions. Send the notebooks and helper files in a zip by *** 2021. They will be corrected and graded

1 Quantum Programming Basics

Question 1 Implement the following circuit, simulate it for $\theta = 0.3$, and print all states along with their amplitudes and probabilities.



Where $ham\_X(\theta) = e^{-i\theta X}$ with $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

Hint Define ham_X as an "AbstractGate", then apply it using ".ctrl()" to get a controlled version, as suggested in the "minimal notebook".

2 Iterative quantum phase estimation, reproducing results from [1]

2.1 Hamiltonian simulation

The Hamiltonians whose ground state energy we are trying to compute are of the form:

$$H(R) = g_0 I + g_1 ZI + g_2 IZ + g_3 ZZ + g_4 YY + g_5 XX$$

Because we use **Trotterization**, we will only have to implement the unitary evolutions generated by each of the individual terms: e^{-iXXdt}, e^{-iZIdt}, e^{-iIZdt}, e^{-iZZdt}, e^{-iYYdt} and e^{-iXXdt}.

We could implement them using Abstract gates and a numerical computation of the matrix exponential. But the computation of the matrix exponential does not scale well with the number of qubits, and thus this method cannot be used in the "general case", when working with a large molecule.

What we will do therefore instead is find small parametrized circuits implementing each of the unitary evolutions. The small circuits may only use "usual gates", such as CNOTs and single-qubit Pauli rotations, that are already pre-implemented in PyAQASM/myqlm. We will then use **QRoutines** to paste these circuits into the final algorithm. Such a method does yield circuits of polynomial size for arbitrary Pauli products over n qubits.

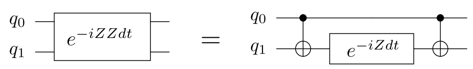
Example: You can check that: $Z \otimes Z = CNOT_{0\to 1} \cdot I \otimes Z \cdot CNOT_{0\to 1}$

with $CNOT_{0\to 1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$, $\otimes$ the Kronecker product and $\cdot$ the standard matrix product.

It allows us to write:

$$\begin{aligned} e^{-iZZdt} &= e^{-i(CNOT_{0\to 1})\cdot(IZ)\cdot(CNOT_{0\to 1})dt} \\ &= CNOT_{0\to 1} \cdot e^{-iIZdt} CNOT_{0\to 1} \\ &= CNOT_{0\to 1} \cdot \left(I \otimes e^{-iZdt}\right) \cdot CNOT_{0\to 1} \end{aligned}$$

i.e:



where $e^{-iZdt} = \begin{pmatrix} e^{-idt} & 0 \\ 0 & e^{idt} \end{pmatrix} = RZ(-dt)$.

RZ is a "standard gate", pre-implemented in PyAQASM. Likewise, RX and RY are pre-implemented.

Question 2 Following the example above, which is already implemented in the notebook, and knowing in addition that:

$$X \otimes X = CNOT_{0\to 1} \cdot X \otimes I \cdot CNOT_{0\to 1}$$

$$SXS^{\dagger} = Y$$

and

$$Y \otimes X = CNOT_{0\to 1} \cdot Y \otimes I \cdot CNOT_{0\to 1}$$

write down the QRoutines carrying out each of the required Hamiltonian simulations, by filling the templates already written in the notebook.

2.2 Iterative Phase estimation

2.2.1 Trotterization

Question 3 Write a function taking as input hamiltonian coefficients, an interval of time dt, and a Trotter number p, and returning a Qroutine which implements the corresponding Trotterized implementation of the Hamiltonian.

Below where your implementation should go, a cell contains a piece of code that tests whether the Trotterization produces a "reasonable" implementation of the Hamiltonian.

2.3 final plots

References

[1] Peter JJ O'Malley, Ryan Babbush, Ian D Kivlichan, Jonathan Romero, Jarrod R McClean, Rami Barends, Julian Kelly, Pedram Roushan, Andrew Tranter, Nan Ding, et al. Scalable quantum simulation of molecular energies. *Physical Review X*, 6(3):031007, 2016.