Quantum Computing PHYS-541, Project 11

Teacher : vincenzo.savona@epfl.ch
Assistant : sara.alvesdossantos@epfl.ch, david.linteau@epfl.ch, shao.chiew@epfl.ch

Variational dynamics on quantum devices

In this project, we wish to learn and practice *variational quantum simulation*. Indeed, quantum devices are expected to be the perfect platform for quantum simulation. Simulating a quantum system on a quantum device means translating the law governing its evolution in time – the Schroedinger's equation – into a set of instructions for our platform. In the case of digital quantum computers, the instructions corresponds to the universal set of gates we have seen during the course. A very good review article on digital quantum simulation can be found here.

However, current quantum devices show limitations in circuit width and depth, due to short coherence times of the qubits and high error rates in manipulating them. This hinders the application of quantum devices to the simulation of quantum systems of practical utility.

To extend capabilities of current hardware beyond its current limits, many researchers focused on the application of the variational method. This strategy is quite versatile and has been proved useful for classical simulation of a variety of quantum systems. In class we have seen an example of them: the Variational Quantum Eigensolver (VQE) for ground state search. Here, we will focus on the use of the variational method for quantum dynamics on quantum device, a topic which has been introduced in this work and extensively reviewed here.

The goal of the project is:

1. Read and understand the main article and present at the exam how the variational dynamics works (very briefly! Strictly less than 5 minutes!).

2. Use the article and the digital quantum simulation review to understand how non-variational quantum simulation is performed exactly and using the Trotter-Suzuki decomposition of the time evolution operator. Use the code given below to have a classical benchmark of the time evolved wave function for a Heisenberg chain of three spins with initial state $|\psi\rangle = |110\rangle$. The code given measures and plots and the occupation probability of $|\psi\rangle = |110\rangle$ in time (look also at Fig. 5b of the digital quantum simulation review).

3. Use the variational time evolution algorithm to reproduce the same simulation of the Hesienberg chain.

4. Compare your results with a non-variational, Trotter-decomposed time evolution circuit. How deep is the variational circuit compared to the non-variational one?

5. Think critically about the limitations of the variational methods in dynamics, what are the main drawbacks? How many circuit do you have to evaluate compared to Trotter?

Code to simulate the system classically via Qiskit

Here below you can find a code to simulate exactly on a classical device the dynamics of the Heisenberg chain. Please note the size of $U_{\text{Heis3}}(t)$ in matrix form. It's representated by an 8×8 matrix. This is because there are $2^3 = 8$ states in the $N = 3$ system. If the simulation were of 50 particles ($N = 50$), $U_{\text{Heis}}(t)$ would be approximately $10^{15} \times 10^{15}$

```
# Import classical libraries to simulate the model
import numpy as np
import matplotlib.pyplot as plt
import scipy.linalg as la

# Import operators and states from qiskit
from qiskit.quantum_info import Pauli, SparsePauliOp, Statevector


## Construct the matrix representation of the Hamiltonian
H_heis3 = SparsePauliOp.from_list([('XXI', 1), ('IXX',1),('YYI', 1), ('IYY',1)
    ,('ZZI', 1), ('IZZ',1)]).to_matrix()

# Define a function that creates the time evolution operator at time t
def U_heis3(t):
    return la.expm(-1j * H_heis3 * t)

# Define array of time points
ts = np.linspace(0, np.pi, 100)

# Define initial state |110>
initial_state = Statevector.from_label('011').data ## Define the initial state
   as |110>, remembering the inverse order of qubits in Qiskit

# Compute probability of remaining in |110> state over the array of time points
 # initial_state with @ gives the bra of the initial state (<110|)
 # @ is short hand for matrix multiplication
 # U_heis3(t) is the unitary time evolution at time t
 # t needs to be wrapped with float(t) to avoid a bug
 # the two @ return the inner product <110|U_heis3(t)|110>
 #  np.abs(...)**2 is the modulus squared of the innner product which is the
   expectation value, or probability, of remaining in |110>
probs_110 = [np.abs((initial_state @ U_heis3(float(t)) @ initial_state))**2 for
    t in ts]

# Plot evolution of |110>
plt.plot(ts, probs_110)
plt.xlabel('time')
plt.ylabel(r'probability of state $|110\rangle$')
plt.title(r'Evolution of state $|110\rangle$ under $H_{Heis3}$')
plt.grid()
plt.show()
```