Coding Project on Hamiltonian Simulation - Part II
Quantum Computation

Reminder on Part I.

We begin with a brief recap of the most important concepts and formulas from Part I that will be useful throughout Part II. The summary below is not comprehensive, you might still need to refer back to other parts of Part I for details.

Quantum Time Evolution. For a Hamiltonian H, the time evolution of a quantum state is given by

$$|\psi(t)\rangle = e^{-iHt}|\psi(0)\rangle.$$

Since H is Hermitian, e^{-iHt} is unitary.

Hamiltonian Simulation. The goal of Hamiltonian simulation is to efficiently approximately simulate quantum time evolution, i.e. implement an approximation of the unitary e^{-iHt} using only poly(n)-many single- and two-qubit gates.

Pauli Strings. An n-qubit Pauli string is a tensor product

$$P = P_1 \otimes \cdots \otimes P_n, \qquad P_i \in \{I, X, Y, Z\}.$$

Pauli Decomposition of a Hamiltonian. Every Hermitian matrix on n qubits can be decomposed as

$$H = \sum_P a_P P,$$

where the sum runs over all n-qubit Pauli strings.

Note that, while there are 4^n-many n-qubit Pauli strings, in practice, any physical Hamiltonian will be a sum of at most poly(n)-many Pauli strings.

Trotterized Hamiltonian Simulation. The first-order Trotter approximation of the quantum time evolution e^{-iHt} of a Hamiltonian with Pauli decomposition $H = \sum_P a_P P$ is given by

$$\left(\prod_P e^{-ia_P P t/r}\right)^r. \tag{1}$$

Thus, trotterized Hamiltonian simulation reduces to implementing exponentials of Pauli strings.

The approximation error scales as

$$O\left(\frac{t^2}{r} m^2 \max_P |a_P|^2\right), \tag{2}$$

where m is the number of non-zero Pauli coefficients.

Rotation Gates. The single-qubit rotation gates are defined by

$$R_P(\theta) = e^{-i\theta P/2}, \qquad P \in \{X, Y, Z\}. \tag{3}$$

In Qiskit, rotation gates are implemented by :

```
qc.rz(theta, qubit)
qc.rx(theta, qubit)
qc.ry(theta, qubit)
```

In general, every exponential $e^{-i\theta P}$ of an n-qubit Pauli string P and any $\theta \in \mathbb{R}$ can be decomposed into a product of $O(n)$ many CNOTs, rotation gates and Hadamards (You only need to know and use the circuit implementation of the Pauli exponentials which you saw in Part I).

Exercises

Exercise 1 *An algorithm for trotterized Hamiltonian simulation*

In Part II, you will piece together all ingredients from Part I to implement approximate Hamiltonian simulation for a particular Hamiltonian of interest, the *transverse-field Ising Hamiltonian* (which we introduce in Exercise 2).

You will begin by writing down (a sketch for) the algorithm that constructs the quantum circuit for trotterized Hamiltonian simulation. We assume that the Pauli decomposition of H consists of at most poly(n)-many Pauli strings with non-zero coefficients.

1. Write down a sketch for a function that takes as input :
   - an n-qubit Hamiltonian, provided in its Pauli decomposition, i.e. as a poly(n)-length list of pairs of Pauli strings (as strings of letters I, X, Y, Z) and their associated coefficient in the Pauli decomposition of H.

     Example : `H = [(0.5,"XII"), (-1,"IZI")]`
   - the number of Trotter steps r
   - the evolution time t

   and outputs the quantum circuit corresponding to the Trotter product approximation (Equation 1) of the unitary e^{-iHt} in terms of O(poly(n))-many CNOTs, Hadamard and rotation gates. You can assume you have access to a function

   `qc.pauli(P, theta)`

   that takes as input a Pauli string `P` and and angle `theta` and appends the circuit corresponding to $e^{-i\texttt{P}\cdot\texttt{theta}}$ (in terms of O(n) CNOTs, Hadamard and rotation gates) to the quantum circuit `qc`. Your sketch can be formulated in words, pseudocode or a mix of both, whatever seems most appropriate at each step of the algorithm. It should be formulated in an analogous way to how you would implement it in Qiskit, as this is what you will do later.
2. Why does the description of the input Hamiltonian have to be provided in the above-mentioned format ? Why do we not just provide the Hamiltonian as a single Hermitian matrix ? We assume that H is such that its Pauli decomposition can be computed efficiently given its description as a single matrix, therefore this is not a consideration here.

Exercise 2 *The Transverse-Field Ising Hamiltonian*

For the implementation part of this project, you will implement the trotterized Hamiltonian simulation algorithm for a particular type of Hamiltonian, the *transverse-field Ising Hamiltonian*. In this exercise, we define the transverse-field Ising Hamiltonian and explain its significance.

The Transverse-Field Ising Hamiltonian

One of the central goals in condensed matter physics is to understand and predict the magnetization properties of materials – for example, why some materials exhibit a stable magnetic orientation while others do not. In this exercise, we are interested in simulating the time-evolution of the magnetization of a piece of magnetic material.

At the microscopic level, a magnetic material can be modeled as a collection of units called *spins*. Each spin is a qubit system, meaning it has two basis states, $|0\rangle$ and $|1\rangle$. These states describe the orientation of the magnetization at the atomic level in the material.

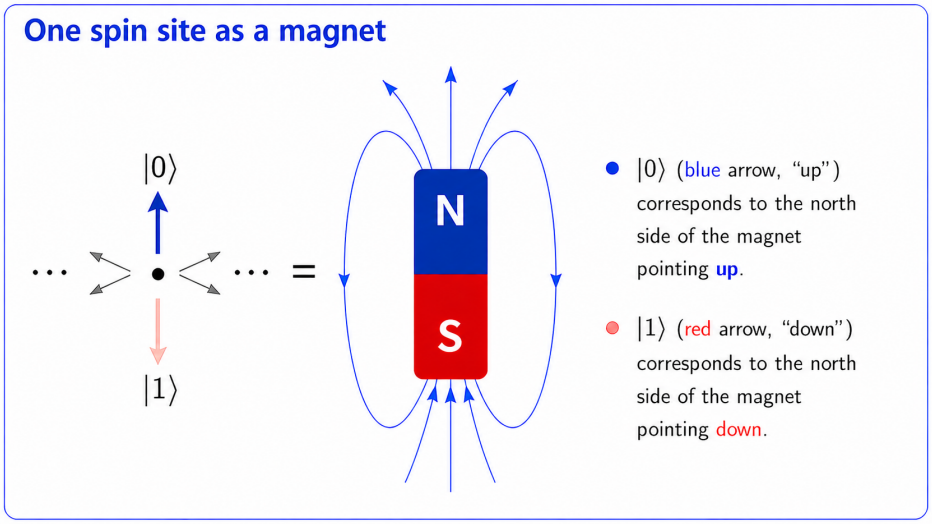

FIGURE 1 – The qubit state $|0\rangle$ in $\mathbb{C}^2$ models a small magnetic site in the material, where the magnetization points "up". Likewise, the state $|1\rangle$ models a magnetization that points "down".

To model a piece of a material capable of magnetization, we consider many qubits associated with the vertices of a graph, typically a line or a two-dimensional grid. In other words, each vertex i is associated with a qubit system $\mathbb{C}^2_i$, where $i \in \{0, ..., n-1\}$. The full configuration is described by the joint state space

$$\mathbb{C}^2_0 \otimes ... \otimes \mathbb{C}^2_{n-1} \cong (\mathbb{C}^2)^{\otimes n} \, . \tag{4}$$

For the sake of simplicity, we will consider a graph in the shape of a *ring*, meaning that qubit $n-1$ is also connected back to qubit 0 as depicted in Figure 2.

The *transverse-field Ising model* is defined by the Hamiltonian

$$H_{Ising} = -J \sum_{i=0}^{n-1} Z_i Z_{i+1(\text{mod n})} - h \sum_i X_i \, . \tag{5}$$

Here, J and h are positive numbers. X_i and Z_i are the Pauli string matrices acting only on the qubit indexed by $i \in \{0, n-1\}$:

$$X_i = \mathbb{I} \otimes ... \otimes \mathbb{I} \otimes \underset{i\text{th place}}{X} \otimes \mathbb{I} \otimes ... \otimes \mathbb{I} \, , \tag{6}$$

$$Z_i = \mathbb{I} \otimes ... \otimes \mathbb{I} \otimes \underset{i\text{th place}}{Z} \otimes \mathbb{I} \otimes ... \otimes \mathbb{I} \, . \tag{7}$$

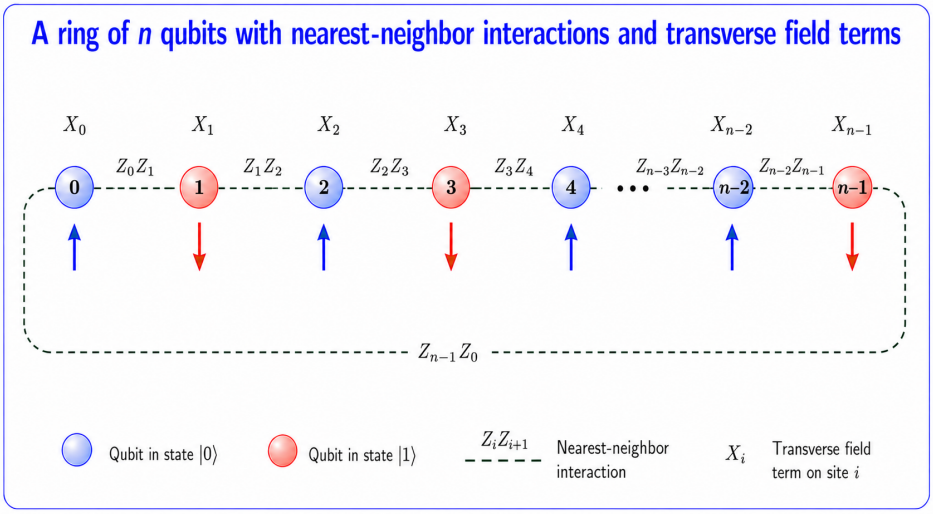

FIGURE 2 – A collection of qubits arranged in a ring. The terms $Z_iZ_{i+1 \pmod n}$ define which qubits neighbor which.

To better understand the Hamiltonian, recall from Part I that when it is given as a Pauli decomposition

$$H = \sum_P a_P P, \tag{8}$$

the corresponding time evolution can be approximated using the Trotter formula :

$$e^{-itH} \approx \left(\prod_P e^{-ia_P P t/r} \right)^r. \tag{9}$$

In other words, the global evolution is approximated by repeatedly applying many simpler unitary operations, each generated by a single Pauli string P. Therefore, to understand the overall dynamics of the system, it is useful to first understand the action of the elementary unitaries

$$e^{-ia_P P t/r} \tag{10}$$

appearing in the approximation. There are only two types for us to inspect :

1. **Interaction terms :** $e^{i(Jt/r)\, Z_iZ_{i+1 \pmod n}}$.
   (a) Verify that Z_iZ_j is diagonal in the computational basis.
   (b) Show how the corresponding unitary $e^{i(Jt/r)\, Z_iZ_{i+1 \pmod n}}$ acts on computational basis states. Namely, compute

$$e^{i(Jt/r)\, Z_iZ_{i+1 \pmod n}} |a\rangle_i |b\rangle_{i+1 \pmod n} \tag{11}$$

   for $a, b \in \{0, 1\}$.

   We conclude that the time evolution generated by the term Z_iZ_j approximately preserves computational-basis states, while potentially adding complex phases to them. In particular, if the material is initiated in the highly ordered (in this context, magnetized) state $|0...0\rangle$, the term Z_iZ_j acts to preserve that order over time.

2. **Transverse field terms :** $e^{i(ht/r)X_i}$**.**
   (a) What is the basis that diagonalizes $e^{i(ht/r)X_i}$?
   (b) Verify that $e^{i(ht/r)X_i}$ acts as the rotation gate $R_{X_i}(\theta)$ seen in (3). For fixed values of h and r, what are the values of t such that $e^{i(ht/r)X_i}$ acts proportionally to the bit flip gate X_i ?

   We conclude that the term X_i is a "quantum noise term". It continuously (in time) mixes the computational-basis states, introducing quantum superpositions to the evolution of the initial state $|0...0\rangle$ from before. The parameter h controls the strength of this super-positioning effect.

As stated before, we are interested in the overall magnetization of the bulk material. That is, given a state $|\psi\rangle$ of the qubit chain, we would like to estimate the overall alignment of all the qubits. To that end, we use the magnetization operator, here represented as the matrix

$$M = \sum_{i=0}^{n-1} Z_i \ . \tag{12}$$

With it, the overall magnetization of the system is quantified by the expectation value

$$\langle M \rangle_\psi = \langle \psi | \, M \, | \psi \rangle = \sum_{i=0}^{n-1} \langle Z_i \rangle_\psi \ . \tag{13}$$

Intuitively, if $|\psi\rangle$ is a computational state, M counts the number of "up" spins minus the number of "down" spins. For general $|\psi\rangle$ we have that $\langle M \rangle_\psi$ is the expected number of "up" spins minus the number of "down" spins when $|\psi\rangle$ is measured in the computational basis.

Exercise 3 *Hamiltonian simulation of the transverse-field Ising Hamiltonian in qBraid*

In this exercise, you will implement the algorithm from Exercise 1 for the transverse-field Ising Hamiltonian. **Complete the coding exercises in qBraid using the provided Jupyter notebook template.**

As the transverse-field Ising Hamiltonian is already specified in terms of two types of Pauli strings only, for simplicity, you can directly implement a function that only takes as input the parameters J, h of (as opposed to additionally providing the Pauli strings as input) and outputs the trotterized Hamiltonian simulation circuit.

1. Write a function that takes as input :
   - the number n of spins (qubits) for the Hamiltonian
   - an n-qubit QuantumCircuit object (this circuit encodes your input state)
   - the parameters J, h for the transverse-field Ising Hamiltonian
   - the number of Trotter steps r
   - the evolution time t

and appends to the input QuantumCircuit the circuit corresponding to the trotterized Hamiltonian simulation of the transverse-field Ising Hamiltonian. Use only basic Qiskit operations and features such as those you have seen in the exercise sheets, i.e. build the circuit by applying gates (only CNOT, Hadamard and rotation gates) to a quantum circuit. **You are NOT allowed to use Qiskit's built-in operations for Hamiltonians/Trotterization**.

2. In order to help you verify that your implementation is correct, we provide below code that implements exact Hamiltonian evolution. Propose and justify a reasonable method for determining whether two quantum circuits are approximately the same. Specifically, describe a classical or quantum algorithm that takes as input the descriptions of two quantum circuits and outputs a number quantifying how close they are, where the output is zero if the circuits are identical.

   There are many valid approaches, it suffices to provide and justify one. Your algorithm does not need to be computationally efficient to receive full points (optional : try to come up with an algorithms whose runtime scales polynomially with the size of the input circuits). We do not require you to provide formal proofs.

3. Using the method you proposed in the previous exercise, compare the circuit you obtain with your `trotterized_Ising` function to Qiskit's built-in implementation of exact evolution (using the code provided below) for parameters $n = 4$, $t = 5$, $h = J = 1$ and $r \in \{1, 2, 5, 10, 50\}$. What do you expect to see given the theoretical error bound (Equation 2) ? Do your results match your expectation ?

   Hint : You can (but don't have to) use the following lines to obtain the unitary associated to a given QuantumCircuit object `qc` :

   ```
   from qiskit.quantum_info import Operator
   U = Operator(qc).data
   ```

   Another potentially useful function is the `.inverse()` *method which returns the inverse of a given quantum circuit and would be used as*

   ```
   qc_inv = qc.inverse()
   ```

We are now ready to use our Hamiltonian simulation algorithm to study the time evolution of the transverse-field Ising model ! To begin with, we will investigate how the all-zeros state (all spins aligned in parallel) evolves in different parameter regimes.

In particular, you will study how the probability p_0 of obtaining the all-zeros outcome when measuring the evolved state $|\psi(t)\rangle$ changes as function of the evolution time t.

4. Fix system size $n = 4$, $J = 1$ and total evolution time $T = 5$.

   For each transverse field strength $h \in \{0.2,\ 1,\ 2\}$ do the following :

   (a) For each Trotter step number

   $r \in \{1,\ 2,\ 5,\ 50\}$, simulate the evolution $|\psi_t\rangle = e^{-iHt}\,|0^n\rangle$ using a Trotterized circuit, for $t_{step} = 50$ time steps.

   At each time step, estimate (with 4000 shots) the empirical probability distribution of measuring the output of the circuit, and save it (in a dictionary).

Denote by $p_0(t)$ the (empirical) probability to measure the all-zeros state at time t of the simulation.

(b) Implement a function `get_p0(counts,num_shots)` that takes as input a `counts` dictionary (from measuring the output of a circuit `num_shots` times) and the corresponding number of shots `num_shots` and outputs the empirical probability of the all-zeros outcome (i.e. the number of counts of the all-zeros outcome divided by the total number of shots).

(c) Create one plot per value of h :

- x-axis : time $t \in [0, T]$ with $t_{step} = 50$ time steps
- y-axis : empirical probability of the all-zeros outcome
- for each value of r, plot $p_0(t)$ (on the same plot)

We include plots that contain the corresponding values of the exact circuit implementation.

(d) Interpret and discuss how the behavior depends on h and on the Trotter step number r, and compare to the value of the exact solution. Does the Trotterization error you observed qualitatively match the theoretical error bound ?

(e) So far, we have asked you to run the Trotterized Hamiltonian simulation algorithm as a function of time for fixed values of r. However, looking at the theoretical error estimate (Equation 2), is keeping r fixed really the most sensible choice if we want the approximation error to remain roughly constant as the evolution time increases ? How should we choose r as a function of the simulation time ?

(f) Choose an integer-valued function $r(t)$ such that $r(T) = 30$ and such that, according to the theoretical Trotter error estimate, the approximation error should remain roughly constant as a function of time. For every value of h, plot the resulting approximation of $p_0(t)$ together with the exact solution, and interpret your results.

Recall that the main property of interest to us is the time evolution of the magnetization $\langle M \rangle_{\psi(t)}$ (see Equation 13) of the time-evolved state $|\psi(t)\rangle$.

5. Given a quantum circuit that prepares a state $|\psi\rangle$, come up with and justify a procedure/algorithm for empirically approximating the quantity $\langle M \rangle_\psi$.

   Hint : First show that for $|\psi\rangle = \sum_{x \in \{0,1\}^n} \sqrt{p_x}\,|x\rangle$, *we have that* $\langle M \rangle_\psi = \sum_{x \in \{0,1\}^n} p_x \langle M \rangle_x$

6. Implement a function `M_(counts,num_shots)` that takes as input a `counts` dictionary (from measuring the output of a circuit `num_shots` times) and the corresponding number of shots `num_shots` and outputs the empirical expectation value of the output state $|\psi\rangle$ of the circuit from which `counts` was obtained.

7. For the same initial state and parameter regimes as in Task 3.4 (a), plot the total magnetisation $\langle M \rangle_\psi$ of the final state as a function of time using the function `M_z` from the previous exercise.

Exercise 4 *Hamiltonian simulation of the Transverse-Field Ising Hamiltonian in qBraid using real quantum device*

Let's see what happens when we run our approximate Hamiltonian simulation algorithm on a real quantum device!

1. Fix parameters $n = 4$, $J = 1$, $h = 0.2$ and total evolution time $t = 2$.

   For each value of $r \in \{1, 2, 3, 4, 5, 10\}$, construct the Trotterized circuit approximating the evolution $|\psi_t\rangle = e^{iHt} |0^n\rangle$ and execute the circuit using both an ideal simulator and the real quantum device IQM Garnet, using **100 measurement shots** each.

   From the measurement outcomes, estimate the probability p_0 of measuring the all-zeros bitstring. Plot p_0 as a function of r for both the simulator and the real quantum device, and interpret your results.

   To accomplish this task you will need qBraid credits. Each group can request 600 credits using the following procedure : log in to qBraid, in the left bar go to account→wallet and click "request credits". This will go to the teaching team and we will approve the request. However, ***note that 600 credits is only about enough to run the previous circuit twice (i.e. twice run it for all values of*** r ***indicated and with*** 100 ***shots each time)*** *so make sure you test your code/circuit, using a simulator, before running it on the real device. If you need more credits, ask for more but please also justify why by sending the instructor an email.*
2. (Optional) If you have credits left over, feel free to run above simulation for other parameters or quantum devices and interpret your results. *Note : Keep in mind that the credit cost depends on the device used, the number of jobs and the number of shots of each job. Thus, we advise you to estimate the cost of your planned job (see on the right sidebar of qBraid Lab, under "Devices") to make sure that you have enough credits to complete your experiment.*
3. (Optional) If you are interested, Qiskit also allows you to simulate the noise of real quantum devices, as well as custom noise models. This allows you to further explore how hardware noise affects your results without the need for qBraid credits.