Coding Project on Hamiltonian Simulation - Part I
Quantum Computation

Introduction

This project consists of two parts. Part I is this document, and contains of theory exercises around the idea of simulating the time evolution of a quantum system : we learn about Hamiltonians, matrix exponentials, and the Trotter expansion. Part II will be released next week and consists of mostly implementation exercises : we will specify a concrete family of Hamiltonians, ask you to implement their Trotterized time evolution in Qiskit, and run some simulations that will produce concrete numerical results (both on real and simulated hardware).

The project should be solved in pairs. For the theory part, you will upload a pdf (ideally compiled from latex, but a scan of neatly handwritten solutions is also acceptable) to Moodle. For the experimental part, you will submit a jupyter notebook (we will provide a template) as well as a pdf rendering of it with the results of your code executions.

Part I is due Friday May 22nd by midnight, and Part II is due Friday May 29th by midnight.

On the use of AI : you are allowed to search the internet, and query AI chatbots, to help you understand background related to the problems. You are not allowed to input any text from the project or directly ask any of the questions. Your solution should be written by you, not an AI. Any solution that looks AI-generated will be checked against an AI detection tool ; in the case of positive results we may ask the students to give an oral presentation of their project solution, to check understanding.

Time evolution of a quantum system

A fundamental problem in physics is finding the *time evolution of a system* : given a physical system and its initial quantum state, what will be the quantum state at an arbitrary time t after the state was prepared ? When we say "a physical system" in this discussion, we have in mind systems such as a molecule, an array of atoms, or a laser beam. The state of all those systems will be regarded as a normalized vector in a Hilbert-space of a finite dimension, the same definition as in our course.

The initial quantum state of a system is often denoted $|\psi(0)\rangle$, "the quantum state at time $t = 0$", and the state at later times is denoted as $|\psi(t)\rangle$. The goal is therefore, to find

for all $t \in \mathbb{R}_{\geq 0}$ a mapping $\mathcal{M}_t$ such that

$$|\psi(t)\rangle = \mathcal{M}_t(|\psi(0)\rangle) \ . \tag{1}$$

To find $\mathcal{M}_t$, physicists must use their knowledge of the physical laws governing the quantum system at hand. The description of the physical laws of the system is given in the form of a Hermitian matrix H, known as a *Hamiltonian*. The dependence of the quantum state in time is dictated by the Schrödinger equation, a differential equation given by

$$i\frac{\partial}{\partial t}|\psi(t)\rangle = H|\psi(t)\rangle \ . \tag{2}$$

The derivative $\partial/\partial t\,|\psi(t)\rangle$ is taken component-wise, and so the state $|\psi(t)\rangle$ is the unknown *function* we aim to determine. A solution to that equation will amount to a solution of the time evolution problem. Indeed, in the field of differential equations, an equation of the form (2) has a solution determined only by the initial state $|\psi(0)\rangle$. One can show that it is given by

$$|\psi(t)\rangle = e^{-iHt}|\psi(0)\rangle \ . \tag{3}$$

Here, e^{-iHt} denotes the *matrix exponential* of $-iHt$, which is the matrix analogue of the usual exponential function. For a square matrix A, the matrix exponential is defined by the power series

$$e^A = \sum_{k=0}^{\infty} \frac{A^k}{k!} = I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \cdots \ . \tag{4}$$

Thus, we have found the mapping we were looking for ; $\mathcal{M}_t$ is a linear map, and we can write

$$\mathcal{M}_t = e^{-iHt} \ . \tag{5}$$

We can compare this result to what we have learned in the course. We saw that when applying a quantum circuit to a state, the relation between the input state $|\psi_{in}\rangle$ and the output state $|\psi_{out}\rangle$ is given by a *unitary matrix* U,

$$|\psi_{out}\rangle = U|\psi_{in}\rangle \ . \tag{6}$$

In the exercises that follow, you will show that matrices of the form e^{-iHt} are indeed unitary for any Hermitian matrix H.

Hamiltonian Simulation

The goal of Hamiltonian simulation is to *simulate* the time evolution of a quantum system. Concretely, we seek an efficient quantum algorithm that, given a description of a Hamiltonian H, an initial state $|\psi(0)\rangle$ and time $t \in \mathbb{R}_{\geq 0}$, prepares the state

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

In this project, we restrict attention to systems of n qubits. Thus, $|\psi(0)\rangle$ is an n-qubit state and H is a $2^n \times 2^n$ Hermitian matrix that in addition has an efficient description (as a

linear combination of Pauli strings, see later on). We call a quantum algorithm *efficient* if, for every n, it uses only poly(n)-many single- and two-qubit gates.

This objective raises two key questions :

1. How can we implement the unitary e^{-iHt} using only poly(n)-many single- and two-qubit gates ?
2. Can such a decomposition be found in poly(n)-time ?

In general, there is no efficient and exact implementation of the unitary e^{-iHt} in terms of single- and two-qubit gates. However, if we relax the requirement of an exact simulation to an approximate simulation, we can answer both questions above in the affirmative.

In this project, you will study an approximation method for Hamiltonian simulation, called *Trotterization*, along with the key tools needed to construct an algorithm that achieves the goals outlined above.

Using classical computation, this problem can in general not be solved efficiently. Thus, the problem of Hamiltonian simulation is an instance of a problem in which quantum computers offer a computational speed-up.

Overview

In Part I of this project, we cover the theoretical foundations for Trotterized Hamiltonian simulation.

In Exercise 1, you will study key properties of the matrix exponential and see how certain exponentials can be implemented using simple single- and two-qubit gates. In Exercise 2, you will learn about the Trotter product formula and how it leads to an algorithm for approximating time evolution of quantum systems.

In Part II (upcoming), we will walk you through implementing Trotterized Hamiltonian simulation for a specific type of Hamiltonian using Qiskit.

Exercises

Exercise 1 *Matrix Exponentials*

In this exercise, you will study some basic properties of matrix exponentials and how they arise in quantum circuits. You may assume that all series converge and do not need to justify convergence.

We begin by establishing several useful identities that simplify the analytic computation of matrix exponentials and are useful for proving further properties.

1. **Exponential of a diagonal matrix.** Let

$$D = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$$

be an $n \times n$ diagonal matrix.

Show that its matrix exponential is given by

$$e^D = \operatorname{diag}\left(e^{\lambda_1}, \ldots, e^{\lambda_n}\right).$$

2. **Transformation of exponential under basis change.** Let $M = BAB^{-1}$ for $A, B \in \mathbb{C}^{n\times n}$, and B invertible. Show that

$$e^M = Be^AB^{-1}\ .$$

Hint : First prove that $M^k = BA^kB^{-1}$ for all $k \geq 0$. Then use the defining formula (4).

Putting the results of 1. and 2. together, we obtain the following corollary :

Exponential of a diagonalisable matrix. Let $M \in \mathbb{C}^{n\times n}$ be diagonalizable, i.e.

$$M = BDB^{-1},$$

where $D = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$ is a diagonal matrix.

Then $e^M = Be^DB^{-1}$ where $e^D = \operatorname{diag}\left(e^{\lambda_1}, \ldots, e^{\lambda_n}\right)$.

3. **Matrix exponential of tensor product of matrix with identity.** In the context of quantum computation, it is important to understand how the matrix exponential acts on tensor products of matrices.

   Show that for any square matrix A and the identity matrix I,

$$e^{A\otimes I} = e^A \otimes I \tag{7}$$

4. **Matrix exponential of tensor products of arbitrary matrices.** Show (by providing a simple counterexample) that in general, the relation $e^{A\otimes B} = e^A \otimes e^B$ does not hold.

Recall from the introduction that matrix exponentials of the form e^{-iHt} for Hermitian H and $t \in \mathbb{R}$ are of particular interest in quantum physics.

5. **Unitarity of quantum time evolution.** Prove that for Hermitian $H \in \mathbb{C}^{n\times n}$, $t \in \mathbb{R}_{\geq 0}$, the matrix exponential e^{-iHt} is unitary.

As mentioned in the introduction, we want an algorithm that implements e^{-iHt} using single-and two-qubit gates. In the rest of this exercise, you will learn about a special subset of n-qubit Hamiltonians, the set of *n-qubit Pauli strings*, whose time evolution admits a particularly straightforward implementation.

Pauli matrices and Rotation gates. Recall the Pauli matrices

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \qquad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \qquad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

The gates $R_X(\theta), R_Y(\theta), R_Z(\theta)$, defined by $R_P(\theta) = e^{-i(\theta/2)P}$ for $P \in \{X, Y, Z\}$, are referred to as *rotation gates*.

Rotation gates play an important role in quantum computing, as every 2-qubit unitary can be decomposed into a product of rotation gates and CNOTs (as you showed in the midterm exam). In Qiskit, the operation for applying a quantum gate R_Z(`theta`) to qubit number `i` of a QuantumCircuit object `qc` is

```
qc.rz(theta,i)
```

and analogously for the R_X, R_Y gates by replacing `rz` with `rx`, `ry`, respectively.

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

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

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.

For example, the unitary $e^{-i\theta/2(Z\otimes I\otimes\cdots\otimes I)}$ is equal to $R_Z(\theta) \otimes I \otimes \cdots \otimes I$ (this follows from Exercise 1.3). Thus, this unitary operation can be implemented as a quantum circuit using a single-qubit gate, by applying the $R_Z(\theta)$ gate to the first qubit.

6. (a) Show that
$$e^{-i\theta(Z\otimes Z)}$$
can be implemented using two CNOT gates and one R_Z rotation gate.

   Draw the corresponding circuit.

   Hint : Think about how a CNOT transforms the $Z \otimes Z$ operator.

   (b) Using that $HZH = X$ (where here H is of course a Hadamard gate!), derive a circuit for
$$e^{-i\theta(X\otimes X)}$$
using only Hadamard gates, CNOT gates, and one R_Z gate.

   Draw the corresponding circuit and justify, using identities from class or from the previous questions, that it computes the right matrix exponential.

Exercise 2 *Trotter product formula*

As you have shown in the previous exercise, every unitary corresponding to the exponential of a Pauli string can be efficiently implemented using standard quantum gates.

Hermitians as sums of Pauli strings. A convenient fact is that every Hermitian matrix (equivalently : Hamiltonian) can be decomposed into a real-valued linear combination of Pauli strings, i.e. for every $2^n \times 2^n$-dimensional Hermitian matrix H there exist coefficients $a_P \in \mathbb{R}$ such that we can write

$$H = \sum_P a_P P \ ,$$

where the sum runs over all 4^n possible Pauli strings P.

In this exercise, we will explore how this fact can be used to efficiently implement approximate Hamiltonian evolution.

Exponentials of sums of matrices. For the scalar exponential function, we know of the convenient relation

$$e^{a+b} = e^a \cdot e^b.$$

If this relation held for matrix exponentials as well, we could conclude that e^{-iHt} for $H = \sum_P a_P P$ is exactly equal to $\prod_P e^{-ia_P t P}$. This is an operation we know how to implement efficiently using rotation, Hadamard and CNOT gates (see Exercise 1). However, the relation $e^{A+B} = e^A \cdot e^B$ does *not* hold in general.

1. Find an example for a pair A, B of 2-dimensional matrices such that the relation $e^{A+B} = e^A \cdot e^B$ does not hold.

   Optional : Identify a simple relation between A and B that guarantees $e^{A+B} = e^A \cdot e^B$.

The good news is that we can still *approximate* exponentials of Hermitian matrices by products of exponentials of Pauli strings using the *Trotter product formula*. We begin by reviewing the notion of approximation in the context of quantum circuits, i.e. unitaries.

(*Note :* You have seen/proven some of these definitions and properties in Exercise Sheet 6, Exercise 3, *Approximating QFT*.)

Approximation of Unitaries. For two unitaries U, V we say that V is an approximation of U if U, V are close with respect to the distance D induced by the *operator norm*. The operator norm of a complex matrix A is defined as

$$||A||_{op} := \max_{\|\psi\|_2 = 1} \|A\,|\psi\rangle\|_2 \ ,$$

where $||\cdot||_2$ is the usual L_2 vector norm used in class. The induced distance D is given by

$$D(A, B) := ||A - B||_{op}$$

Recall that every norm, by definition, fulfills the triangle inequality

$$||A + B||_{op} \leq ||A||_{op} + ||B||_{op} \ . \tag{8}$$

An important property of the induced distance D is that it is subadditive for unitary matrices, i.e. for any unitaries U_1, U_2, V_1, V_2, we have that

$$D(U_1U_2, V_1V_2) \leq D(U_1, V_1) + D(U_2, V_2) \ . \tag{9}$$

Trotter Formula. For two matrices A and B, the (first-order) Trotter formula states that

$$D\left(e^{x(A+B)}, e^{xA}e^{xB}\right) = O(x^2||A||_{op} \cdot ||B||_{op}) \tag{10}$$

In particular, setting $x = 1/r$ for some integer r and using the subadditivity of D, one can show that

$$D\left(e^{A+B}, \left(e^{A/r}e^{B/r}\right)^r\right) = O(\frac{1}{r}||A||_{op} \cdot ||B||_{op}) \tag{11}$$

and thus

$$\lim_{r\to\infty} \left(e^{A/r}e^{B/r}\right)^r = e^{A+B} \ .$$

(In case you wonder, the limit of a family of matrices is the matrix obtained by taking the entrywise limit, whenever it is well-defined.)

Thus, as r becomes larger, the approximation of e^{A+B} by $\left(e^{A/r}e^{B/r}\right)^r$ becomes more accurate, and converges to the exact value of e^{A+B}.

Trotterized Hamiltonian Simulation. The idea of Trotterised Hamiltonian simulation is to approximate the unitary e^{-iHt} by $\left(\prod_{j=1}^m e^{-iH_jt/r}\right)^r$, where the H_j are Hermitian matrices such that $H = \sum_{j=1}^m H_j$ and there is an efficient implementation of $e^{-iH_j\delta}$ for any $\delta \in \mathbb{R}$.

An important question here is how we should choose r to ensure a good approximation. We will now walk you through a derivation of the first-order error (i.e. the error term that scales linearly in $1/r$, neglecting higher-order error terms in $1/r$), as a function of the simulation time t, the maximal norm $h_{\max} := \max_j ||H_j||_{op}$, and the number m of Hamiltonian terms.

Hint : Over the following two exercises, you will need to use each of Equations 8,9 and 10 once.

2. **Bounding the error of "splitting-off" a single H_j term, in the time slice δ.** Show that

$$D(e^{-iH\delta}, e^{-iH_1\delta}e^{-i\sum_{j=2}^m H_j\delta}) = O(\delta^2 m h_{\max}^2) \ . \tag{12}$$

Using similar reasoning to the previous question, it is possible to iteratively "split off" each $e^{-iH_j\delta}$ term and, using the triangle inequality for the distance measure D, conclude that

$$D(e^{-iH\delta}, \prod_{j=1}^m e^{-iH_j\delta}) = O(\delta^2 m^2 h_{\max}^2) \ . \tag{13}$$

You may use this equation without proving it.

3. **Bounding the error of combining time slices.** Use Equation (13) for a suitable choice of δ to conclude that :

$$D\Big(e^{-iHt}, \Big(\prod_{j=1}^{m} e^{-iH_j t/r}\Big)^r\Big) = O\Big(\frac{t^2}{r} m^2 h_{\max}^2\Big) \,. \tag{14}$$

In particular, if H is a Hamiltonian with Pauli decomposition $H = \sum_P a_P P$ then Equation (14) implies that the evolution of H can be approximated by the unitary

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

with an approximation error that scales as $O(\frac{t^2}{r} m^2 \max_P\{|a_P|^2\})$, where m is the number of Pauli terms with non-zero coefficients a_P.

Note that in the equation above, the product over P can in principle be taken in any order, since which Pauli is considered the "first", etc., is arbitrary. Due to non-commutation of matrix exponentials the resulting unitaries are not necessarily exactly equal, but the reasoning we carried out above implies that they are approximately equal. In your implementations later on, you are free to choose any order you want.