Quantum Challenge 2024 Report

Kyunghee University
Department of Applied Physics
InHee Yun

Problem 1. Quantum State Tomography

1. Quantum States with Few Qubits

Part I.

(a) To determine the state of a single qubit, measurements in different bases are required. First, by measuring in the Pauli Z basis, one can measure the states 0 and 1 with probabilities of a^2 and b^2, respectively. Through the process of quantum tomography, an approximate probability distribution can be obtained. From these probability values, the square roots can be taken, and under the positive and real condition of a and b, each can be uniquely determined. When applying a Hadamard Gate H to an arbitrary single qubit state, the resulting state is $\frac{a+be^{i\theta}}{\sqrt{2}}|0\rangle + \frac{a-be^{i\theta}}{\sqrt{2}}|1\rangle$. This allows for measurements in the Pauli X basis to be performed. From this state, the probability of measuring 0 can be derived as $p(0) = \frac{1}{2}(1 + 2ab\cos(\theta))$. Since the values of a and b have already been determined, the value of $\cos(\theta)$ can be inferred from the probability distribution. However, since the value of θ is not uniquely determined, one more measurement is required. By using the S gate and H gate and measuring, one can derive $p(0) = \frac{1}{2}(1 - 2ab\sin(\theta))$. This allows for measurements in the Pauli Y basis to be performed. This process allows the value of θ to be uniquely determined.

(b) By expanding (a), measurements can be performed in more diverse bases. To reduce errors, the code uses the Z, HZ(=X), HSZ(=Y), HSSZ, and HSSSZ bases to derive the average values of a, b and θ. Through this process, the unitary operation U_{gen} was obtained using the Ry and Rz operators.

Part II.

(a) The same process as in Part I. (a) is followed. In a pure state, when measuring in the $Z \otimes Z$ basis, the measurement probabilities for the states 00, 01, 10, and 11 are a^2, b^2, c^2 and d^2, respectively, allowing the amplitudes to be determined. The signs can be determined by measuring in the $(H \otimes H)(Z \otimes Z)$ basis. This allows for measurements in the Pauli $(X \otimes X)$ basis to be performed. The measurement probabilities are $(a + b + c + d)^2$,

$(a-b+c-d)^2$, $(a+b-c-d)^2$ and $(a-b-c+d)^2$, and from these, the values of a b, c, and d can be uniquely determined based on the measurement probabilities.

(b) We need to perform measurements for all combinations of Pauli bases. For a general complex 2-qubit state where the amplitude of the 00 state is a real number, it can be expressed in terms of positive real number a, b, c, d, θ_1, θ_2 and θ_3. That is, the state $|\psi\rangle$ can be written as:

$$|\psi\rangle = a|00\rangle + be^{i\theta_1}|01\rangle + ce^{i\theta_2}|10\rangle + de^{i\theta_3}|11\rangle$$

First, as done in (a), measurements are taken in the $(Z\otimes Z)$ basis to determine the magnitudes of the amplitudes, and allowing the values of a, b, c, d Next, measurements are performed by combining the Hadamard gate and S gate for various bases. The measurements are taken in the bases $(H\otimes H)(Z\otimes Z)$, $(H\otimes H)(S\otimes S)(Z\otimes Z)$, $(H\otimes H)(S\otimes I)(Z\otimes Z)$, $(H\otimes H)(I\otimes S)(Z\otimes Z)$, $(H\otimes I)(S\otimes I)(Z\otimes Z)$, $(I\otimes H)(I\otimes S)(Z\otimes Z)$, $(H\otimes I)(Z\otimes Z)$, and $(I\otimes H)(Z\otimes Z)$. By solving the system of equations from these measurements, the values of θ_1, θ_2 and θ_3 can be determined.

Part III.

As in Parts I and II, by measuring in all possible combinations of Pauli bases, any arbitrary quantum state can be determined. Each qubit can be measured in the Pauli X, Y, and Z bases by using the Hadamard and S gates. Therefore, with three possible basis choices per qubit, a total of 3^n basis measurements are required for n qubits. This means performing 3^n distinct measurements. Although, in practice, fewer basis measurements can be sufficient to determine an arbitrary qubit state, as experienced in Part II, errors inevitably increase as the number of qubits grows. Thus, to obtain a reliable state, it is advantageous to measure as many cases as possible, requiring measurements in all Pauli bases.

2. Graph States

a) In a graph state, tomography can be performed with fewer bases by leveraging the properties of the stabilizer. The strategy is as follows: since the graph state is an eigenstate of the stabilizer with an eigenvalue of 1, a vertex where Pauli X is applied is measured in the X basis, while its neighboring edges are measured in the Z basis. This setup ensures that the number of outcomes labeled as 1 in the measurement bases will always be even. This is because measuring 1 in the Z basis indicates that the eigenvalue for that qubit is -1, and to satisfy the stabilizer's eigenvalue-1 condition, an even number of qubits must have an eigenvalue of -1. Thus, using this condition, we can iterate over each vertex, applying a Hadamard H to the corresponding qubit and measuring it in the Z basis as an effective X basis measurement. After doing this, we measure all qubits in the Z basis to observe the state and infer possible combinations of neighboring edges. This approach enables tomography to be carried out using only a number of bases equivalent to the number of qubits.

b) Using the strategy described in (a), I implemented a tomography code in Qiskit. When the number of states measured across many qubits far exceeded the number of shots, setting shots to 100 yielded a high error rate and did not produce meaningful results. Upon increasing the shots, however, the score improved, indicating better accuracy in the tomography results.

3. Low Depth Brickwork Circuits

The approach to the problem is as follows: in Low Depth Brickwork Circuits, we take advantage of the property that entanglement is weak. By disregarding each instance of entanglement, we obtain information for each qubit individually, thereby approximating the original state as closely as possible in a disentangled form.

Problem 2. Dicke State Preparation

1. a) Let's define the Hamiltonian H as follows.

$$H = \sum_{i=1}^{n} (\sigma_z)_i = (\sigma_z \otimes I \otimes \cdots \otimes I) + (I \otimes \sigma_z \otimes \cdots \otimes I) + \cdots + (I \otimes I \otimes \cdots \otimes \sigma_z)$$

When applying H to an n-qubit state $|x\rangle$ with no superposition or entanglement, the result is as follows:

$$H|x\rangle = (-1)^{x_1}|x\rangle + (-1)^{x_2}|x\rangle + \cdots (-1)^{x_n}|x\rangle = (n-2k)|x\rangle$$

So, $|x\rangle$ becomes an eigenstate of H with an eigenvalue of n−2k. This allows the extraction of the Hamming weight k directly from the eigenvalue.

b) For an n-qubit state, the range of possible Hamming weights is from 0 to n, giving a total of $n+1$ possible Hamming weight values. Thus, to represent all Hamming weights in binary, $\lceil \log_2(n+1) \rceil$ bits are required.

2. Based on the results calculated in Problem 1 and Algorithm 2 proposed in reference [1], a circuit was constructed that uses Quantum Phase Estimation (QPE) to project the given state onto its Hamming weight. The 'calculate_hamming_weight' function converts the measurement results into binary form and multiplies each state's Hamming weight by its measurement probability to calculate the expected value. This result provides the Hamming weight for the given state.

3. e) As described above, the Hamming weight is calculated based on the example provided in the introduction of the referenced paper. Interpreting case (c), the state is in a superposition where it has a probability of 1/2 for Hamming subspace 0 and a probability of 1/2 for Hamming subspace 2. Thus, the theoretical expected value of the Hamming weight is 1. In the actual Qiskit simulation, noise effects result in a value close to 1, consistent with this expectation.

4. For an n-qubit state, start with a fully uniform superposition input state. This can be easily prepared by applying n Hadamard gates to the initial state. Then, execute the function defined in problem 2, repeating the measurement process until the desired Hamming weight k is measured. After measurement, the state will lie within the subspace corresponding to the desired Hamming weight k. Since the initial state was uniformly superposed, it will also be uniformly distributed within the subspace, resulting in a state equivalent to the Dicke state for the chosen Hamming weight k.

5. a) For an input state uniformly superposed with k = 50, the number of possible states is $\binom{100}{50}$. Since the total number of possible states for 100 qubits is 2^{100}, the probability of measuring the desired state is given by:

$$\frac{\binom{100}{50}}{2^{100}}$$

b) When generating a Dicke state using the method from Problem 4, the probability of measuring the Dicke state is equal to the probability of measuring the desired k state. Let p be the probability of measuring this state at least once. For a number of measurements N, the probability of measuring the desired state at least once is:

$$1-(1-\frac{\binom{100}{50}}{2^{100}})^N \geq p$$

Rearranging this inequality to find the minimum number of measurements N, we get:

$$N \geq \frac{ln(p)}{ln(1-\frac{\binom{100}{50}}{2^{100}})}$$

c) Now, let's calculate the number of gates needed for the algorithm in Problem 2. For n=100, the C register requires $\lceil \log_2(n+1) \rceil = 7$ qubits. The gate requirements are as follows:

- Preparing the initial uniform state requires 100 Hadamard gates.
- At the start of the algorithm, we need 7 Hadamard gates for the C register.
- The controlled-RZ gates require 7×(100+(128−100−1))=8897 $7 \times (100 + (128 - 100 - 1)) = 889$ gates.
- The inverse-QFT requires 7 Hadamard gates.
- Swap gates needed are 3.
- Controlled rotation gates are $\frac{7\times(7-1)}{2} = 21$.

Thus, the total number of gates required is 1027. For N measurements, this total gate count will scale by N, resulting in N x 1027 gates.

Problem 3. Heisenberg Spin Glass

Part 1.

a) Based on the provided code, I used 'numpy' and 'scipy' to construct the Hamiltonian for each Heisenberg model and calculated their eigenvalues. By comparing these eigenvalue distributions, I was able to deduce the number of symmetry classes. The number of symmetry classes is 6.

b) The sets of coupling coefficients J_{ij} for each symmetry class are as follows:

- $J_{01} = -1, J_{02} = -1, J_{03} = -1, J_{12} = -1, J_{13} = -1, J_{23} = -1$
- $J_{01} = 1, J_{02} = -1, J_{03} = -1, J_{12} = -1, J_{13} = -1, J_{23} = -1$
- $J_{01} = 1, J_{02} = 1, J_{03} = -1, J_{12} = -1, J_{13} = -1, J_{23} = -1$
- $J_{01} = 1, J_{02} = 1, J_{03} = 1, J_{12} = -1, J_{13} = -1, J_{23} = -1$
- $J_{01} = -1, J_{02} = -1, J_{03} = 1, J_{12} = 1, J_{13} = -1, J_{23} = -1$
- $J_{01} = 1, J_{02} = -1, J_{03} = 1, J_{12} = 1, J_{13} = -1, J_{23} = -1$

d) The eigenspectrum for each symmetry class is as follows:

- [-6, -6, -6, -6, -6, 2, 2, 2, 2, 2, 2, 2, 2, 2, 6, 6]
- [-4, -4, -4, -4, -4, -4, -4, -4, 0, 4, 4, 4, 4, 4, 4, 8]
- [-6, -6, -6, -2, -2, -2, -2, -2, -2, 2, 2, 2, 6, 6, 6, 6]
- [-8, -8, -8, 0, 0, 0, 0, 0, 0, 0, 4, 4, 4, 4, 4, 4]
- [-6, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, -2, 6, 6, 6, 10]
- [-6.92820323, -5.65685425, -5.65685425, -5.65685425, 0, 0, 0, 0, 0, 0, 0, 0, 5.65685425, 5.65685425, 5.65685425, 6.92820323

e) When writing the code to solve the problems, I ensured that it took into account the sign flip symmetry condition. This condition states that even if the signs of the all coefficients J_{ij} change, two models can still be considered equivalent under certain symmetry transformations. Without considering this symmetry, there would be a total of 11 symmetry classes. However, by accounting for sign flip symmetry, the number of symmetry classes reduces to 6. This reduction implies that there are eigenspectrum that remain invariant under certain sign changes in J_{ij}.

Thus, by incorporating the sign flip symmetry into the code, the models that are equivalent under this transformation are grouped together, simplifying the classification and revealing the inherent symmetries in the eigenspectrum structure.

Part 2.

a) In Part 1, I implemented a function to plot the eigenspectral graphs for each of the 6 symmetry classes as the value of s varies. This function, Problem_a(), generates the correct answer graph when executed from the main program. Each plot shows the eigenvalues for a given symmetry class as s is adjusted, providing a clear visualization of how the eigenspectrum changes with s for each class.

b) I implemented the function in the same manner as in (a), so that when the Problem_b() function is executed, the graph is displayed. This function is designed to plot the relevant eigenspectral data, ensuring that the visualization appears correctly when Problem_b() is run from the main program.

c) The initial state was set to the ground state of the mixer Hamiltonian, and adiabatic evolution over time was performed. By approximating the integral form for small time steps, as suggested in the problem, I was able to construct the unitary operator for time evolution as an exponential of the product of the Hamiltonian matrix and the time increment.

d) To obtain the probability distribution, I first computed the ground state of the target Hamiltonian, H_{sg}. Then, I compared the probability distribution of the adiabatically evolved state to that of this ground state. Using np.allclose(), I checked if each probability value in the distributions was similar up to the third decimal place. If they were similar, the program output "success"; otherwise, it output "failed."

After multiple trials, I observed that around $\tau = 900$ and $N = 900$, the two probability distributions became similar. Increasing these values further resulted in even greater similarity between the distributions, as expected.

The same function was used to solve both (c) and (d), following this approach.

e) Listing each term individually for the Hamiltonians H_{sg} and H_m results in the following:

$$H_{sg} = \sum_{<i,j>} [J_{ij}\sigma_x^{[i]} \otimes \sigma_x^{[j]} + J_{ij}\sigma_y^{[i]} \otimes \sigma_y^{[j]} + J_{ij}\sigma_z^{[i]} \otimes \sigma_z^{[j]}] + \sum_i h_i\, \sigma_z^{[i]}$$

$$H_m = \sum_i \sigma_x^{[i]}$$

In the unitary evolution process, each term in the Hamiltonian acts as a Pauli rotation gate, as described in the problem. Therefore, to implement H_{sg} in each step, we require:

- 6 RXX gates (for XX interactions),
- 6 RYY gates (for YY interactions),
- 6 RZZ gates (for ZZ interactions),
- 4 RZ gates (for single-qubit Z terms).

This results in a total of:

- 4 single-qubit gates (RZ),
- 18 two-qubit gates (6 RXX, 6 RYY, and 6 RZZ).

Similarly, for H_m in each step:

- We need 4 RX gates for each qubit, requiring 4 single-qubit gates in total.

Therefore, the total gate count per step is:

- 8 single-qubit gates (4 RZ from H_{sg} and 4 RX from H_m),
- 18 two-qubit gates (from the RXX, RYY, and RZZ gates in H_{sg}).

This means that each step requires a total of 26 gates.

f) Considering this setup roughly, we first need to prepare the ground state of H_m, which requires 4 Hadamard gates. Then, as calculated in (e), each step requires 8 single-qubit gates and 18 two-qubit gates.

To achieve three-decimal precision, we determined in (d) that N=900 steps are necessary. In an ideal, noise-free environment, this would mean:

- 900 steps total, requiring
- 7200 single-qubit gates (8 gates per step × 900 steps)
- 16200 two-qubit gates (18 gates per step × 900 steps)

Including the initial 4 Hadamard gates, the total number of gates required would be:

- 7204 single-qubit gates,
- 16200 two-qubit gates.

Thus, for just a 4-qubit adiabatic evolution, a total of 23404 gates would be required, which is impractical.

Moreover, since we assumed a noise-free environment, significantly more gates would be needed in reality to counteract errors. Additionally, in actual computation, the Pauli operators in the Hamiltonian do not commute, so the Trotter-Suzuki approximation must be used. Achieving the desired accuracy with this approximation would further increase the gate count substantially. Therefore, implementing this method in a practical, real-world environment is not feasible due to the high gate count and the impact of noise and errors.

Part 3.

a) We generated H_{sg} in the same way as in Part 2, and verified all XY-mixer combinations through intertool combinations. We used the minimize function from scipy to find the minimum value. The method for measuring similarity between the two probability distributions involved comparing each probability distribution using Euclidean distance. As a result, we found that applying the XY-mixer to the qubit combinations (0, 1), (1, 2), and (1, 3) minimized the value when the rotation angles (in radians) of the XY-mixer were 1.32519923, 1.2158871, and 0.67324855, respectively. At this time, the initial state is $|1110\rangle$.

b) I implemented the XY-mixer rotation angles and qubit combinations obtained in part (a) directly into the circuit. Using Qiskit's statevector_simulator, we obtained the state of the circuit and compared it to the ground state of H_{sg} using Euclidean distance. Similar to part (a), the distance between the two probability distributions was very low, allowing us to achieve the desired result.a)

c) Upon checking the state, we found an imaginary coefficient in the $|1101\rangle$ state. Although applying an Rz gate to shift the phase by π could remove this, doing so would also affect other qubits, preventing us from achieving the desired result. Since we can disregard the global phase, we only need to consider the relative phase difference. Therefore, by rotating the qubits in the $|1101\rangle$ state with a value of 1 by $-\frac{\pi}{4}$ using Rz and rotating those with a value of 0 by $\frac{\pi}{4}$, we were able to obtain the desired result.

d) We generated the circuit using the same process as in part (c). Following the instructions in the problem, we applied the Hadamard gate and Rz gate to this state to perform measurements in the X, Y, and Z bases, respectively. Since Qiskit stores bitstrings in reverse order, we adjusted the outcome accordingly. Based on the probability values obtained for each state, we calculated the expectation value of the Hamiltonian using the coefficients previously provided in the problem. As a result, we were able to obtain a ground state energy close to -19.5.