QOSF tasks

Matthew Kendall

***Abstract*—This report outlines the results of tasks 1-4 for the QOSF screening tasks for cohort 10.**

I. Task one

A. state vector simulation

The statevector simulator for quantum circuits was implemented using matrix multiplication. A quantum register is initialized in the $|0\rangle$ state, represented as a vector. Quantum gates, including the Pauli-X, Hadamard, and CNOT gates, are modeled as matrices and applied sequentially to individual qubits. Each gate's effect on the full quantum system is computed by combining it with identity matrices for the unaffected qubits using the Kronecker product.

The CNOT gate, which operates on two qubits, modifies the target qubit based on the state of the control qubit. The statevector is updated at each step of the circuit through matrix multiplication with the appropriate gate matrices. Performance was assessed by measuring the runtime of the simulation as the number of qubits increased. The runtime scales exponentially with the number of qubits due to the exponential growth in the statevector's size, making simulations impractical for more than 15 qubits without substantial computational resources.

B. Tensor simulation

The code utilizes tensors to simulate quantum circuits, where the quantum register and gate operations are represented as multi-dimensional arrays (tensors). The quantum register is initialized as a tensor of shape $(2, 2, \ldots, 2)$, corresponding to the number of qubits, with the first element set to 1, representing the $|000\cdots0\rangle$ state. All other elements are initialized to zero.

Quantum gates are applied to the register using tensor contractions. Each gate, such as the Pauli-X, Hadamard, and CNOT, is represented as a matrix or higher-order tensor. For example, the Pauli-X and Hadamard gates are 2×2 matrices, while the CNOT gate is represented as a $2\times2\times2\times2$ tensor. The function `apply_gate` performs a tensor contraction between the gate tensor and the quantum register using `np.tensordot`, which allows the gate to act on specific qubit indices.

The tensor contraction is performed over the axes corresponding to the qubits that the gate acts on, while preserving the overall structure of the quantum state. After the contraction, the resulting tensor is transposed using a custom transposition function to reorder the axes and maintain consistency in the qubit indexing. This approach ensures that the quantum operations are efficiently applied to the multi-qubit register, with the tensor structure providing a flexible representation of the quantum state throughout the simulation.

C. results and extension

This simulations showed that the tensor simulation was significantly more efficient, with the runtime exponentially increasing after around 23 qubits whereas for the statevector simulation this was around 13.

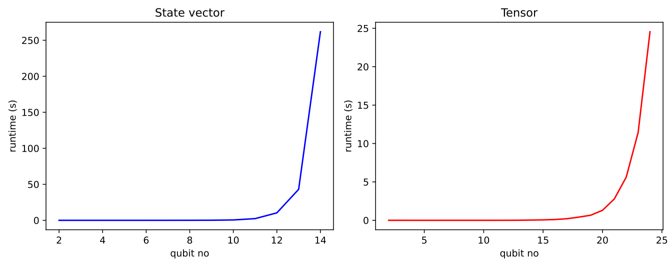

Fig. 1. Comparison of runtime of statevector and tensor simulations

D. Sampling from the final state:

To sample from the final states in the statevector or tensor representation, we first compute the probability distribution over the computational basis states. For a given quantum state $|\Psi\rangle$, each basis state $|i\rangle$ has an associated probability $P(i) = |\langle i|\Psi\rangle|^2$, which corresponds to the square of the amplitude of $|i\rangle$ in the statevector. In the case of a tensor representation, the state amplitudes are found by flattening the tensor into a vector and then calculating the norm of each element.

Once the probabilities are computed, sampling can be done by generating random numbers and selecting basis states according to the computed distribution. This process mimics the behavior of quantum measurements in which the system collapses to one of the basis states with a probability proportional to the square of its amplitude.

E. Computing expectation values:

To compute the exact expectation value of an observable $\langle\Psi|\mathrm{Op}|\Psi\rangle$, we use the statevector or tensor representation of the quantum state. The expectation value is calculated by first applying the operator Op to the state $|\Psi\rangle$, yielding $\mathrm{Op}|\Psi\rangle$. In the tensor framework, this involves contracting the operator tensor with the state tensor. The resulting vector (or tensor) is then conjugated and dotted with the original state vector $\langle\Psi|$, which produces a scalar value representing the expectation $\langle\Psi|\text{Op}|\Psi\rangle$.

In practice, for tensor-based representations, the operator is also represented as a tensor, and the tensor contraction tools (such as `np.tensordot`) can be employed to efficiently compute the expectation value. This allows for the exact evaluation of physical observables in quantum simulations.

II. Task two

A. *Pauli Noise Model*

In this task, noise was modeled using Pauli operators (X, Y, Z) with probabilistic application after each gate. A function was developed to apply random Pauli noise after single-qubit and two-qubit gates in a quantum circuit. The noise model introduces random Pauli operators with probabilities p_1 (for single-qubit gates) and p_2 (for two-qubit gates). For each gate in the circuit, the function generates a random number and applies the noise operator accordingly.

B. *Gate Decomposition to Basis Gates*

Quantum computers support a limited set of gates, known as the gate basis. In this implementation, we transformed a general quantum circuit into the gate basis {CX, ID, RZ, SX, X}. Gates not part of this basis were decomposed. For example, the Hadamard gate (H) was decomposed into a combination of $RZ(\frac{\pi}{2})$, SX, and another $RZ(\frac{\pi}{2})$. Similarly, the Y and Z gates were expressed using rotations and Pauli operations.

C. *Quantum Fourier Transform (QFT) and Quantum Addition*

The Draper adder algorithm was implemented to add two numbers using quantum gates. The algorithm leverages the Quantum Fourier Transform (QFT), which was built from scratch. The QFT applies Hadamard gates and controlled phase rotations to qubits in the circuit. After encoding the first number into the qubits, the second number is added by applying phase shifts. Finally, the inverse QFT is applied to bring the qubits back to the computational basis.

D. *Effects of Noise on Quantum Addition*

The circuit used for quantum addition was subjected to different noise levels by combining the Pauli noise model and gate decomposition techniques. The impact of noise on the result was analyzed by varying the noise probabilities and comparing the accuracy of the sum. An expected decrease in the accuracy of the adder was observed.