Christian B. Mendl, Irene López Gutiérrez, Keefe Huang due: 22 November 2021, 08:00 on Moodle

Tutorial 4 (Quantum circuit simulation)

In the lecture you learned about the tensor product of vector spaces to define multiple qubit spaces. Here we discuss how to compute and work with quantum gates applied to these qubits.

Given two vector spaces V and W and two operators A and B acting respectively on them, we define $A \otimes B$ implicitly via

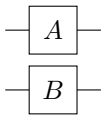
$$(A \otimes B)(|v\rangle \otimes |w\rangle) = (A\,|v\rangle) \otimes (B\,|w\rangle) \quad \text{for all } |v\rangle \in V \text{ and } |w\rangle \in W.$$

Using matrix notation, this leads to the *Kronecker product* of two matrices $A \in \mathbb{C}^{m\times n}$ and $B \in \mathbb{C}^{p\times q}$:

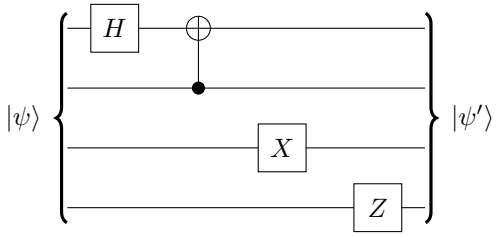
$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{pmatrix} \in \mathbb{C}^{mp\times nq},$$

with a_{ij} the entries of A, and the terms $a_{ij}B$ denoting $p \times q$ submatrices. The corresponding circuit diagram is

$A \otimes B =$



Now consider the following quantum circuit:

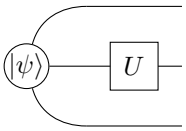


(a) What is the overall matrix representation of the first operation on $|\psi\rangle$?

(b) Assemble this matrix using Python/NumPy.

(c) If $|\psi\rangle$ was a 20 qubit state, what would be the dimension of the gates acting on it? Is it possible to store a *dense* matrix of such dimension on your laptop/PC? Discuss.

If a gate only acts non-trivially on a subset of the qubits, it can be implemented without forming the overall matrix. Specifically for a single-qubit gate U acting on the i-th qubit (counting from zero), we can first reshape the input state $|\psi\rangle$ into a $2^i \times 2 \times 2^{N-i-1}$ tensor and then apply U to the second dimension.

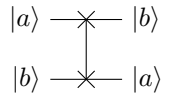


(d) Write a function that applies H to an arbitrary qubit of an input state $|\psi\rangle$ using a "matrix-free" approach.

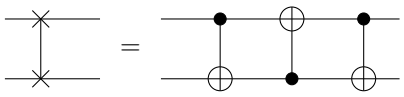
(e) Implement analogous functions for the X, Z and CNOT gates. Test your implementation via the above circuit with a random input state.

Exercise 4.1 (Basic quantum circuits)

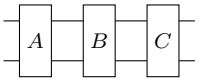
(a) Find the matrix representation (with respect to the computational basis states $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$) of the *swap-gate* $|a, b\rangle \mapsto |b, a\rangle$, which is written in circuit form as



Also show that the swap operation is equivalent to the following sequence of three CNOT gates:

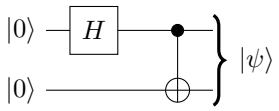


Hint: You can either work directly with basis states, e.g. $|a, b\rangle \overset{\text{CNOT}}{\mapsto} |a, a \oplus b\rangle$, or use matrix representations. In the latter case, note that a sequence of gates like



(with A, B, C unitary 4×4 matrices) corresponds to the matrix product CBA since the circuit is read from left to right, but the input vector in the matrix representation is multiplied from the right.

(b) Compute the output $|\psi\rangle$ of the following "entanglement circuit" applied to the input $|00\rangle$:



with $H = \frac{1}{\sqrt{2}} \left(\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix} \right)$ denoting the Hadamard gate.

(c) Build the CNOT gate from the controlled-Z gate and two Hadamard gates, and verify your construction.

Exercise 4.2 (IBM Q / Qiskit and Qaintum)
IBM Quantum (`https://quantum-computing.ibm.com/`) is a quantum cloud service and software platform, which allows users to run experiments even on real quantum computing hardware. For this exercise you should create a personal account and familiarize yourself with the service.

IBM Quantum offers a graphical *Circuit Composer* and *Qiskit Notebooks* as interface, which are Jupyter Python notebooks using the Qiskit open-source framework (`https://qiskit.org`). Both can be conveniently accessed online via a web browser; alternatively, you can also install Qiskit locally via `pip install qiskit`. An introduction to Qiskit is available at `https://qiskit.org/documentation/getting_started.html`.

(a) Use the Circuit Composer to construct the quantum circuit from exercise 4.1(b). You can view the corresponding `OPENQASM` code on the right side of the circuit interface. Compare the computed entries of the "Statevector" on the bottom left with the state $|\psi\rangle$ from exercise 4.1(b).

Please hand in a screenshot of the webpage.

(b) Construct the circuit again using Qiskit (including measurements), and execute the circuit via Aer's qasm_simulator (1024 shots).

Please hand in your code and the generated output.

(c) (Voluntary) Our group is working on a quantum computing software framework and circuit simulator written in Julia, see `https://github.com/Qaintum`. The main repository of the simulator is Qaintessent.jl, for which you will find the documentation under "docs". From your Julia command line you can install it via `using Pkg; Pkg.add("Qaintessent")`. Use this package to construct once again the circuit from 4.1(b) and measure both qubits with respect to the standard basis.

In the upcoming worksheets, programming assignments will be tested automatically by submitting your code through GitLab. Part (c) of this exercise is intended as test run for both you and us. Sign up and log in to your LRZ GitLab account (`https://gitlab.lrz.de/`) and follow these steps:

1. Edit Profile → Change Job Title to group number (set to an integer), i.e., for group 15, fill in '15'.
2. Go to "`https://gitlab.lrz.de/iqc`" and request access.
3. Wait until your request has been processed.
4. Follow the instructions in the ReadMe file.