Christian B. Mendl, Irene López Gutiérrez, Keefe Huang

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

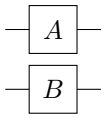
$$(A \otimes B)(|v\rangle \otimes |w\rangle) = (A\,|v\rangle) \otimes (B\,|w\rangle) \quad \text{for all } |v\rangle \in V \text{ and } |w\rangle \in W.$$

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

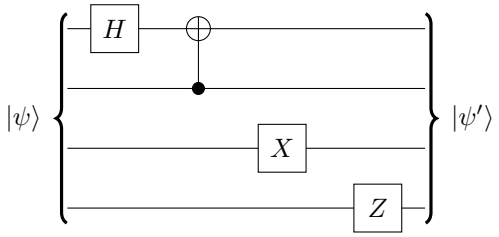
$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{pmatrix} \in \mathbb{C}^{mp\times nq},$$

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:

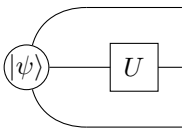


(a) What is the overall matrix representation of the first operation on $|\psi\rangle$?

(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.

Solution

(a) There is a Hadamard gate acting on the top qubit, and identities on the remaining three. Therefore the first operation is

$$H \otimes I_2 \otimes I_2 \otimes I_2 = H \otimes I_8 \in \mathbb{C}^{16\times 16},$$

with I_n denoting the $n \times n$ identity matrix.

(b) The NumPy function `kron` implements the Kronecker product:

```
import numpy as np
H = 1/np.sqrt(2) * np.array([[1, 1], [1,-1]])
H1 = np.kron(H, np.eye(8))
```

(c) It would be a $2^{20} \times 2^{20}$ matrix. Since a double precision floating-point value uses 8 bytes of memory, a matrix of this size would require $2^{20} \cdot 2^{20} \cdot 8$ bytes $= 2^{43}$ bytes $\approx 8.8 \times 10^{12}$ bytes $= 8.8$ TB of memory. No conventional PC or laptop has that much RAM. For a general gate with complex entries, even twice as much memory is needed.

(d) See Jupyter notebook.

(e) See Jupyter notebook.