Lecture 8: Quantum computing

Hendrik Weimer

Institute for Theoretical Physics, Leibniz University Hannover

Quantum Physics with Python, 20 June 2016

Outline of the course

1. Introduction to Python
2. SciPy/NumPy packages
3. Plotting and fitting
4. QuTiP: states and operators
5. Ground state problems
6. Time evolution and quantum quenches
7. Quantum master equations
8. **Quantum computing**
9. Grover's algorithm and quantum machine learning
10. Student presentations

What is a quantum computer?

- Binary representation of a quantum state of N quantum bits ("qubits")

$$|\psi\rangle = \sum_{i\in\{0,1\},j\in\{0,1\},\ldots} c_{ij\ldots}|ij\ldots\rangle$$

- Time evolution under unitary dynamics

$$|\psi(t)\rangle = U|\psi(0)\rangle$$

- Goal: construct U such that $|\psi(t)\rangle$ contains the result of a computational task with high probability

Elementary quantum gates

- As in classical computing, we can decompose the time evolution operator into elementary operations (quantum gates)

$$U = \prod_{i}^{N} U_i$$

- Implement $U_i = \exp(-iH_i t)$ using appropriate physical process (atoms, photons, superconductors, ...)
- Universal set: all unitary matrices U can be expressed using a limited set of elementary gates
- Common choice: z rotation gate, Hadamard gate, Controlled-NOT gates

$$R_z(\phi) = \exp(i\phi\sigma_z/2) = \begin{pmatrix} e^{i\phi/2} & \\ & e^{-i\phi/2} \end{pmatrix} \quad U_H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

$$U_{CNOT} = |0\rangle\langle 0|_A \otimes 1_B + |1\rangle\langle 1|_A \otimes \sigma_B^x = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & & 1 \\ & & 1 & \end{pmatrix}$$

Quantum gates in QuTiP

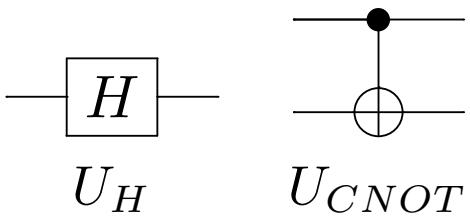
```
from qutip import rz, snot as hadamard, cnot
print(hadamard())
print(cnot())
```

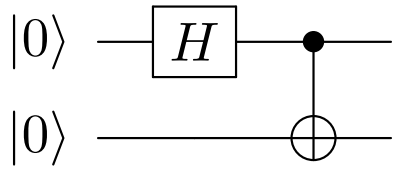
Output:

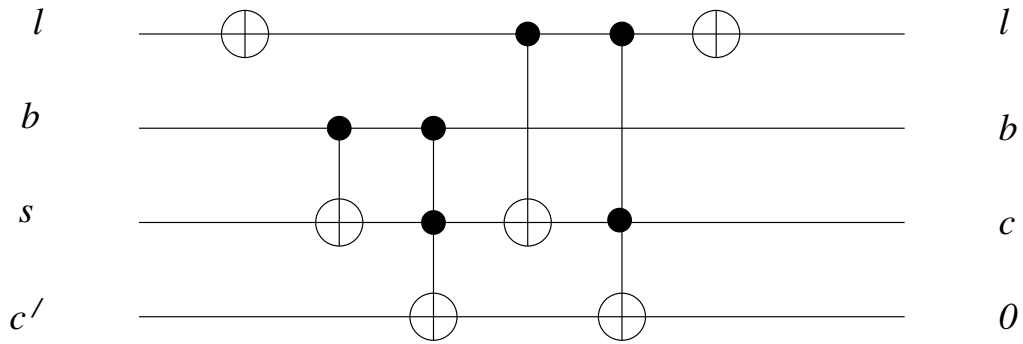
```
Quantum object: dims = [[2], [2]], shape = [2, 2], type = oper,
Qobj data =
[[ 0.70710678  0.70710678]
 [ 0.70710678 -0.70710678]]
Quantum object: dims = [[2, 2], [2, 2]], shape = [4, 4], type =
Qobj data =
[[ 1.  0.  0.  0.]
 [ 0.  1.  0.  0.]
 [ 0.  0.  0.  1.]
 [ 0.  0.  1.  0.]]
```

Multi-qubit systems

- Quantum circuit diagrams







[Beckman et al., PRA **54**, 1034 (1996)]

Quantum circuits in QuTiP

Function parameters:

1. Optional parameter of the gates such as a rotation angle
2. Number of qubits in the entire circuit
3. Qubit that the gate acts upon
4. Optional parameter if gate acts on multiple qubits

```
print(rz(np.pi, 2, 0))
```

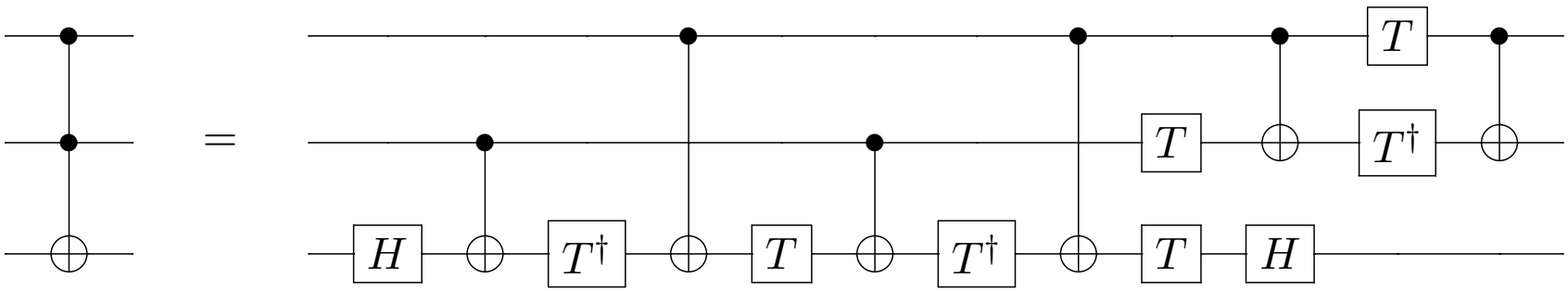
Output:

```
Quantum object: dims = [[2, 2], [2, 2]], shape = [4, 4], type =
Qobj data =
[[ 0.-1.j  0.+0.j  0.+0.j  0.+0.j]
 [ 0.+0.j  0.-1.j  0.+0.j  0.+0.j]
 [ 0.+0.j  0.+0.j  0.+1.j  0.+0.j]
 [ 0.+0.j  0.+0.j  0.+0.j  0.+1.j]]
```

The Toffoli gate

The Toffoli gate is a universal gate for classical reversible computation, i.e., all possible classical computations can be implements (cf. NAND gate for non-reversible compuatation)

$$U_T = \begin{pmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & & & & & \\ & & & 1 & & & & \\ & & & & 1 & & & \\ & & & & & & & 1 \\ & & & & & & 1 & \\ & & & & & 1 & & \end{pmatrix}$$



$(T = R_z(\pi/4))$

The Deutsch-Josza algorithm

- Are quantum computers superior to classical computers?
- Deutsch-Josza algorithm (1992): determine whether an unknown binary function f ("oracle") is either constant (always 0 or 1) or balanced (0 or 1 in exactly half of the input options)
- Classical computer: more than one half of the input options must be tested in the worst case
- n qubits: $2^{n-1}+1$ oracle calls are needed
- Quantum circuit