佐藤 拓也

近年，数値計算はより複雑で，より大規模なモデルを計算対象とする傾向にあります．これに伴い，計算時間もより長くかかるようになり，既存のコンピュータでは現実的な時間で計算が終わらない状況になりつつあります．そこで，超高速計算が可能な量子コンピュータが着目されています．

本章では量子コンピュータを用いた数値計算の技法について解説します．数値計算では線型方程式（未知量$x$に対する$Ax = b$の方程式）が解ければ，大抵の問題に応用ができます．そこで，線型方程式を解く代表的なアルゴリズムとしてHHL（Harrow-Hassidim-Lloyd）アルゴリズムを紹介します．これは量子コンピュータが得意な，固有値を使った高速解法です．また，HHLアルゴリズム内で使われている基礎的なアルゴリズムとして，量子フーリエ変換と量子位相推定も紹介します．これらのアルゴリズムは，短時間での暗号解読や大規模なデータからの高速検索などに応用できます．

# 9-1 量子コンピュータと量子ビット

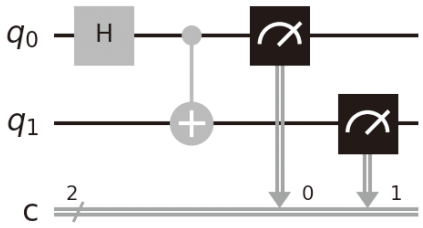


図1 量子ビット$q_0$と$q_1$がそれぞれ確率1/2で観測されるようにした量子回路

## ● 概要

量子コンピュータは，従来のコンピュータとは全く異なる原理で動作します．量子力学の重ね合わせの原理を用いることができ，量子効果を利用していて，理論的には高速計算が可能です．この計算で鍵となるのが，量子コンピュータのビットである量子ビットです．本稿では量子コンピュータを使った計算の基礎として，量子ビットの計算を解説します．

## ● 仕組み

従来のコンピュータのビット（以降，古典ビット）では，1つのビットは0か1のどちらかの状態しか取ることができません．これに対して，量子ビットでは，量子力学の重ね合わせの原理によって，1つのビットが0または1の重ね合わせ状態を取ることができます．両者の違いは，ビットの数が大きくなるほど，極めて大きな差となって表れます．

例えば，古典ビットでは，4ビットで0000 ～ 1111の16種類の状態を表現できますが，そのうちどれか1つしか同時には利用できません．一方，量子ビットでは，4ビットで16種類の状態を同時に利用できます．このため，高速計算が可能となるのです．

0の状態の量子ビットを$|0\rangle$，1の状態を$|1\rangle$と表現し，任意の量子ビット$|\varphi\rangle$は式（1）のように表します．

リスト1 2量子ビットのもつれ状態を観測

```
import numpy as np
from qiskit import QuantumCircuit, transpile
from qiskit.providers.aer import QasmSimulator
from qiskit.visualization import plot_histogram
# Use Aer's qasm_simulator
simulator = QasmSimulator()
# Create a Quantum Circuit acting on the q register
circuit = QuantumCircuit(2, 2)
# Add a H gate on qubit 0
circuit.h(0)
# Add a CX (CNOT) gate on control qubit 0 and
                                        target qubit 1
circuit.cx(0, 1)
# Map the quantum measurement to the classical bits
circuit.measure([0,1], [0,1])
# compile the circuit down to low
                                level QASM instructions
# supported by the backend (not needed for simple
                                            circuits)
compiled_circuit = transpile(circuit, simulator)
# Execute the circuit on the qasm simulator
job = simulator.run(compiled_circuit, shots=1000)
# Grab results from the job
result = job.result()
# Returns counts
counts = result.get_counts(circuit)
print("nTotal count for 00 and 11 are:",counts)
# Draw the circuit
circuit.draw()
```

Qiskitを使う準備

（a）ソースコード

```
(venv)C: Test>python test.py
Total count for 00 and 11 are: {'11': 520, '00': 480}
```

（b）コンソールの出力

$$|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle \cdots\cdots (1)$$

このとき，観測したビットが$|0\rangle$になる確率が$|\alpha|^2$，$|1\rangle$になる確率が$|\beta|^2$で，$|\alpha|^2 + |\beta|^2 = 1$です．便利な表現として次のように表すこともあります[注1]．

$$|\varphi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{j\varphi}\sin\left(\frac{\theta}{2}\right)|1\rangle$$

注1：$\theta$はブロッホ球上の$Z$軸からの傾きを示す．ブロッホ球についてはコラム参照．