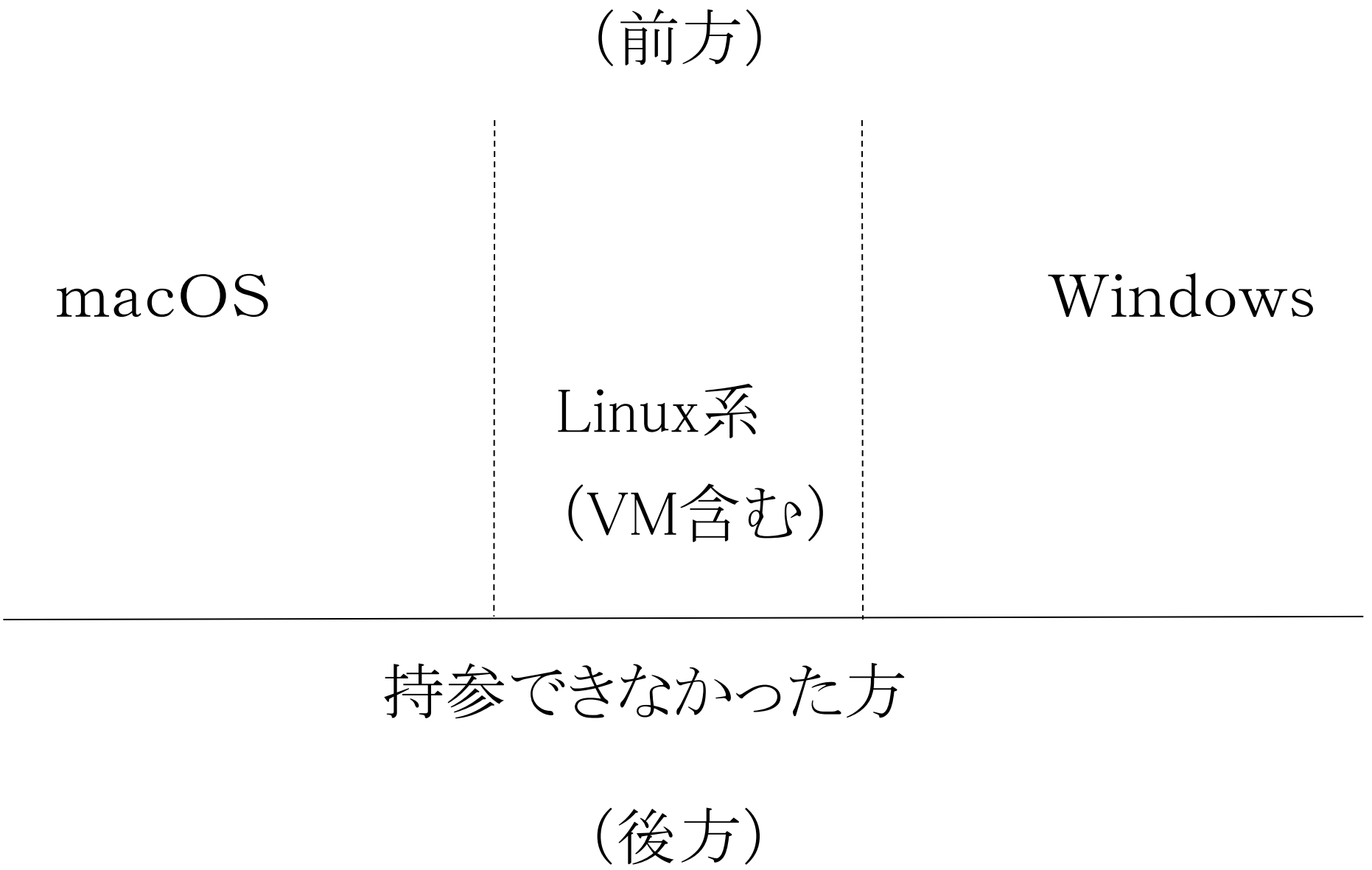
SymPy を使った量子プログラミング

（Pythonハンズオン）

OpenQL勉強会
Feb.19, Apr.23 2018

@k_yamaz

座席 （PCの種類により、席の位置をなんとなく指定しています）



本日のコンテンツ

- SymPy の紹介

  スライドを使って勉強します。

- 基礎的な使い方（量子プログラミングの初歩）

  jupyter notebook を使って勉強します。

- 例題1、例題2、例題3

  Groverアルゴリズムを題材に、プログラミングを通して
  試してみましょう。

- QuantPyの紹介

  jupyter notebook を使って試してみましょう。

- Homeworkと付録は、自習とします。

SymPyとは

・Mathematica のように、数式を数式のまま扱える処理系をめざして開発。

・完全に Python 上で動作。

・非常に軽量で、かつ依存するソフトウェアが少ないライブラリ。

・「無償」提供。

・変数を抽象的なシンボルのまま、代入や計算が行えるのが特徴。

通常の Python で使われる「プログラム上の変数」を扱うのではなく、
数学や理科の教科書にでてくるような x や y という「数学上の変数」をそのまま扱う。

・SymPy のパッケージには、数学計算に便利な基礎的な利用用途とは別に、
様々な応用計算が標準で含まれています。

・実は、その一つに、本題である
「量子コンピューターの計算」を行うための機能が含まれています。

SymPyをインストール

まずは、SymPyをインストールします。
Pythonの環境さえ整っていれば、pip でインストールできます。

```
$ pip install sympy
```

量子計算に必要なモジュールは、SymPy の中でも極一部です。
物理計算のなかの量子計算である sympy.physics.quantum にあります。

```
>>> from sympy.physics.quantum import *
# ↑基本機能のインポート
>>> from sympy.physics.quantum.qubit import Qubit,QubitBra
# ↑ブラケットを使えるようにします
>>> from sympy.physics.quantum.gate import X,Y,Z,H,S,T,CNOT,SWAP
# ↑基本的な量子ゲート（省略形）を使えるようにします
>>> from sympy.physics.quantum.gate import IdentityGate as _I
# ↑恒等変換は省略されないため _I と省略できるようにします
```

（狭義の）量子プログラミングの手順

1. 計算に必要な量子ビット（量子レジスタ）を準備して、
   その値を初期化する

2. 量子計算をユニタリ行列（ゲート演算子）で記述する

3. ユニタリ行列を量子ビットに作用する

4. 測定する

（例題1）
Toffoli（CCX）, CCCX, CCCCX ...

n-qubit における制御NOT系のゲート操作（演算子）は、
量子ゲート操作において重要な役割を果たします。

$$\begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 & 0 \end{pmatrix}$$

$Z = HXH$

$Y = SXS^{\dagger}$

より

CCZ, CCCZ, …

CCY, CCCY, …

も作成可能です。

（例題2）
重ね合わせ状態の中の、ある状態のみマーキングする

重ね合わせ状態の中の、ある状態 S_t のみに、
−1掛ける操作（ここでは、マーキングを呼びます）を考えます。

・3量子ビットのマーキング操作を試してみましょう

（例題3）
重ね合わせ状態の中に、マーキングした状態があるかを見る

Grover のアルゴリズム：

量子状態が与えられていて、
その量子状態に、特定の量子状態が存在していれば、
その量子状態を抽出するアルゴリズムです。

振幅増幅手法 [Amplitude Amplification]

(1) 対象の状態をマーキングする　：M(s)
(2) 平均値周りの反転操作を行う　：D
(3) DM(s) | φ ＞ の操作を、sqrt(n) 回行う

・3量子ビットで状態 | 7＞ を探します

QuantPyの紹介

QuantPy は、OpenQL 発の Python ライブラリです。
現在は、sympy, numpy, qiskit に依存しています。pip でインストールができます。

```
$ pip install quantpy
```

QuantPy は、フロントエンドを SymPy を使っています。

「量子ビットの準備」「量子操作（ゲート演算子）の記述」は SymPy で行います。

QuantPy は、「量子ビットにゲート演算子を作用」するところで、SymPyの qapply() を置き換えて動作します。

QuantPy では、qapply() を実行する Executor を選択的に指定して、計算を行えます。

```
>>> from sympy.physics.quantum import *
>>> from quantpy.sympy.qapply import qapply
```

QuantPyを用いた量子プログラミングの手順

1. 計算に必要な量子ビット(量子レジスタ)を準備して、
   その値を初期化する

2. 量子計算をユニタリ行列(ゲート演算子)で記述する

SymPy を使う

3. ユニタリ行列を量子ビットに作用する

4. 測定する

QuantPyにあるqapply()
に置き換えて計算する

QuantPy の qappy()

```
# Executor を準備する
 ibmq_simulator = IBMQExecutor(backend='local_qasm_simulator')
 classical_simulator = ClassicalSimulationExecutor()

qapply ( circuit, executor = ▲ ▲ ▲ ▲ )
```

Executor（今後の開発で機能アップや種類を増やす予定です！）

IBMExecutor：　IBM Q を利用できます。
内部で QISKit を呼び出し、QISKit の機能で IBM Q も利用可能に。

ClassicalSimulationExecutor：numpy を使った計算ができます。

GPU対応やCython対応も絶賛開発中

（Homework）Toffoli 再び

Toffoli ゲートは、SymPy では、準備された演算子クラス(CGateS)があります。
量子コンピューター（実機）では、標準のゲート演算子だけで表さなければなりません。

課題1） Toffoli ゲートを、基本的な量子ゲートだけで表してください。

→ Quantum circuits of T-depth one (arXiv: 1210.0974v2)

CCCX ゲートは、補助ビット（ancilla bit）を使うと、Toffoli ゲートと CNOT ゲートを組み合わせて表せます。

課題2） CCCX ゲートを、Toffoli ゲートと CNOT ゲートで表してください。

→ Ancilla-Driven Universal Quantum Computation (arXiv: 0911.3783)？