Enhancement of Aer-based quantum_info

AerDensityMatrix by Shunsuke Sotobayashi

Enhancement of Aer-based quantum_info

Improve performance of quantum_info with optimized runtime of Qiskit-Aer

Background:

- Originally one of QAMP fall 22 assignments:
  - https://github.com/qiskit-advocate/qamp-fall-22/issues/23
- Qiskit-terra already has its quantum_info
- More performance by C++ is required
- Enhanced quantum_info should be implemented in Aer

Recently implemented classes:

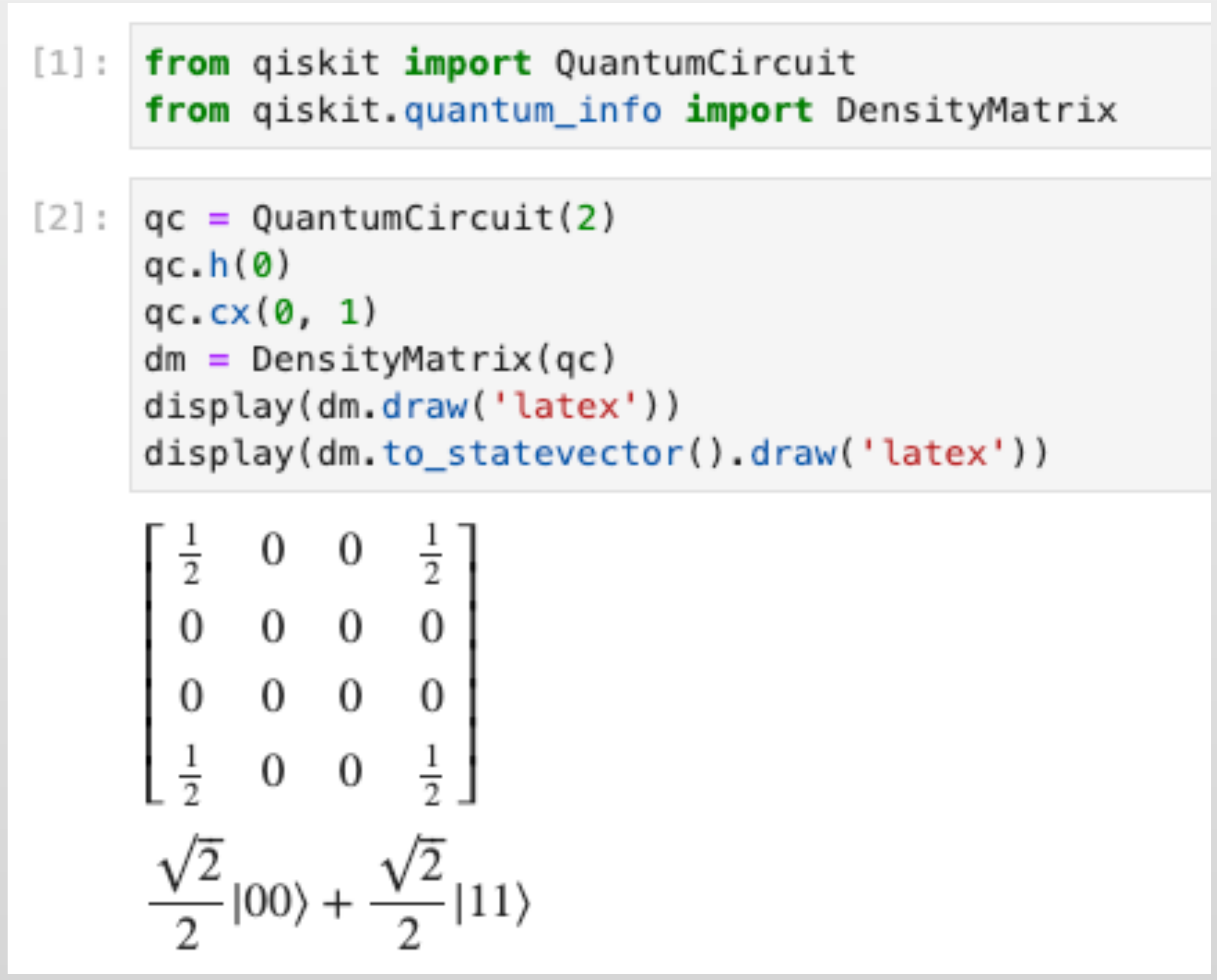
- AerStatevector has been implemented in 0.11.0
- AerDensityMatrix was implemented in 0.12.0

DensityMatrix of Qiskit-Terra

Brief explanation:

- Class in charge of the simulation of density matrices representing pure/mixed states
- Pure Pythonic implementation,so the performance is moderate

Code example:

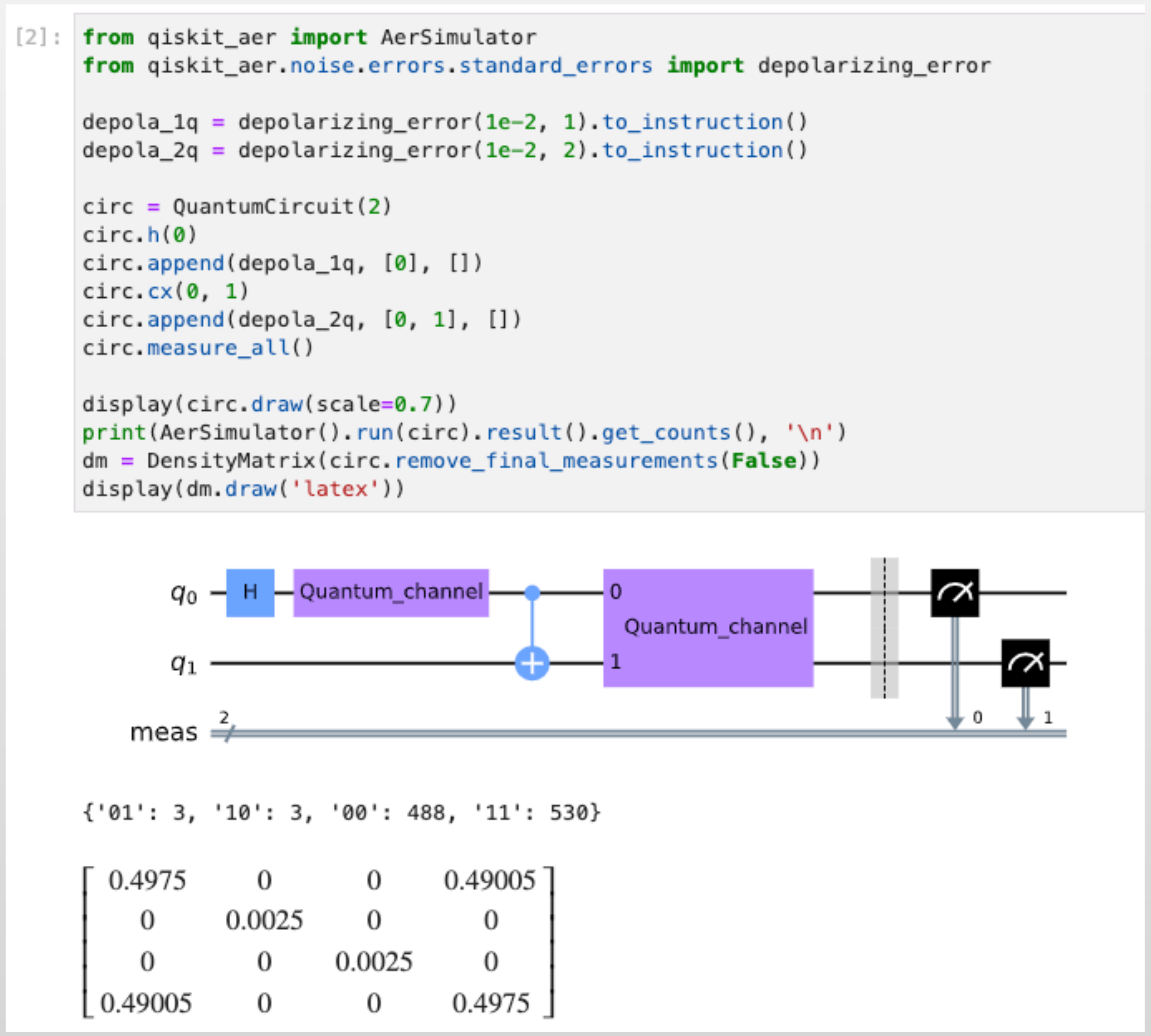


```
[1]: from qiskit import QuantumCircuit
     from qiskit.quantum_info import DensityMatrix
```

```
[2]: qc = QuantumCircuit(2)
     qc.h(0)
     qc.cx(0, 1)
     dm = DensityMatrix(qc)
     display(dm.draw('latex'))
     display(dm.to_statevector().draw('latex'))
```

$$\begin{bmatrix} \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \end{bmatrix}$$

$$\frac{\sqrt{2}}{2}|00\rangle + \frac{\sqrt{2}}{2}|11\rangle$$



```
[2]: from qiskit_aer import AerSimulator
     from qiskit_aer.noise.errors.standard_errors import depolarizing_error

     depola_1q = depolarizing_error(1e-2, 1).to_instruction()
     depola_2q = depolarizing_error(1e-2, 2).to_instruction()

     circ = QuantumCircuit(2)
     circ.h(0)
     circ.append(depola_1q, [0], [])
     circ.cx(0, 1)
     circ.append(depola_2q, [0, 1], [])
     circ.measure_all()

     display(circ.draw(scale=0.7))
     print(AerSimulator().run(circ).result().get_counts(), '\n')
     dm = DensityMatrix(circ.remove_final_measurements(False))
     display(dm.draw('latex'))
```

```
{'01': 3, '10': 3, '00': 488, '11': 530}
```

$$\begin{bmatrix} 0.4975 & 0 & 0 & 0.49005 \\ 0 & 0.0025 & 0 & 0 \\ 0 & 0 & 0.0025 & 0 \\ 0.49005 & 0 & 0 & 0.4975 \end{bmatrix}$$

AerDensityMatrix

Improve performance of quantum_info with optimized runtime of Qiskit-Aer

Features:

- Basically provides APIs compatible with Terra's DensityMatrix
- AerDensityMatrix is about 3x faster than DensityMatrix
  - Full use of C++ under the hood via pybind11
  - Mainly due to Parallelization on C++ side
    - Multi threads
    - SIMD operations
    - GPU if enabled

Usage

Replace:

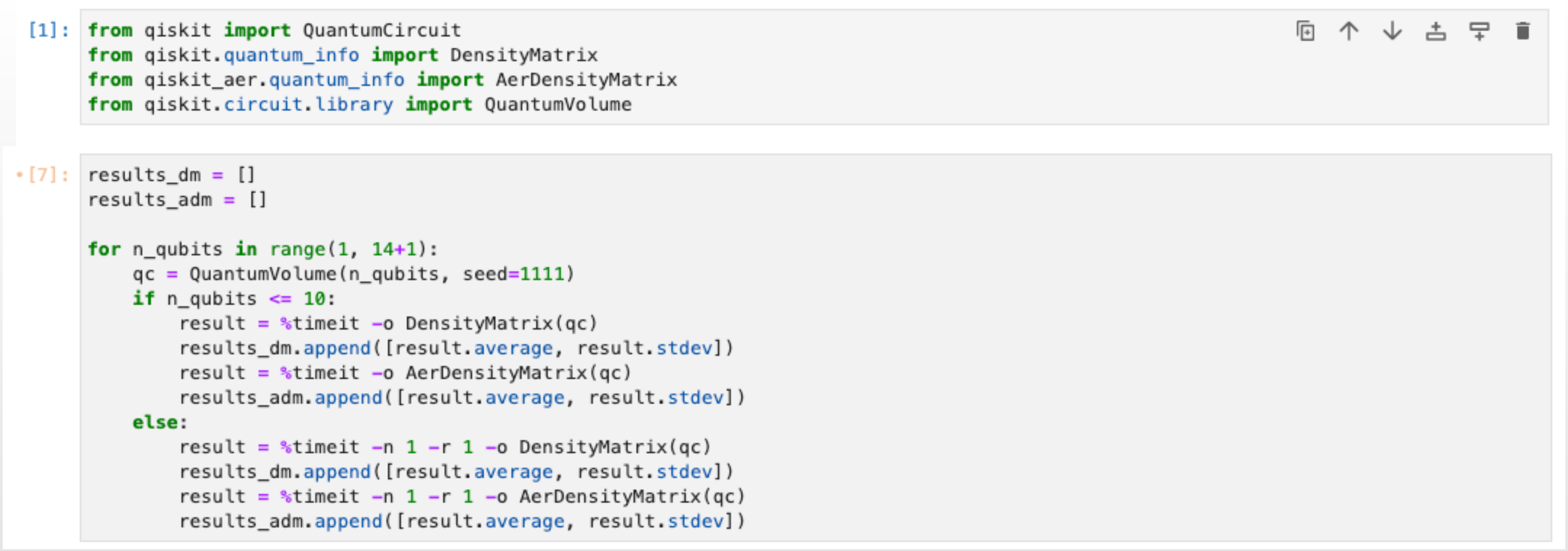
- from qiskit.quantum_info import DensityMatrix

Code example (Terra):

- from qiskit_aer.quantum_info import AerDensityMatrix as DensityMatrix

This is all you need to do!

Performance measurement

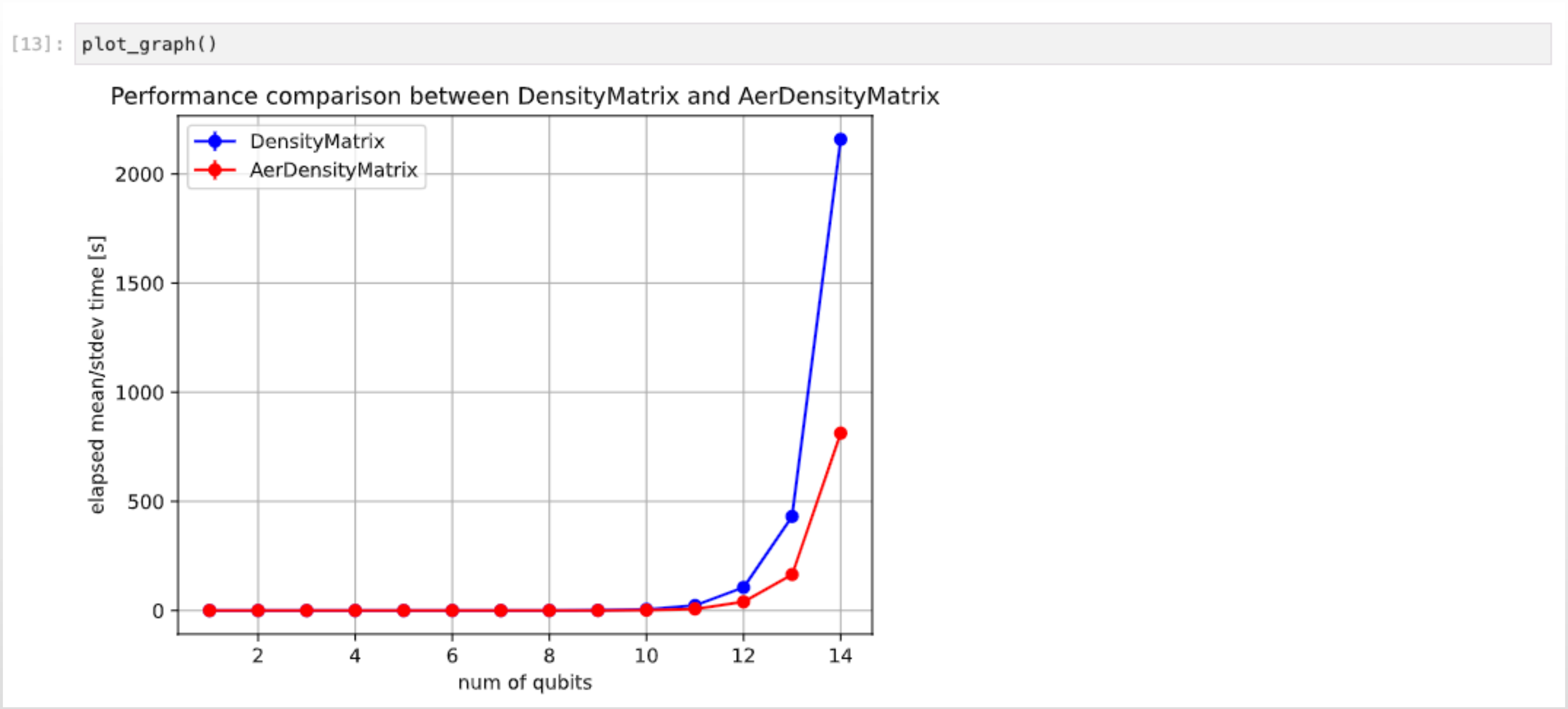


```
[1]: from qiskit import QuantumCircuit
     from qiskit.quantum_info import DensityMatrix
     from qiskit_aer.quantum_info import AerDensityMatrix
     from qiskit.circuit.library import QuantumVolume
```

```
•[7]: results_dm = []
      results_adm = []

      for n_qubits in range(1, 14+1):
          qc = QuantumVolume(n_qubits, seed=1111)
          if n_qubits <= 10:
              result = %timeit -o DensityMatrix(qc)
              results_dm.append([result.average, result.stdev])
              result = %timeit -o AerDensityMatrix(qc)
              results_adm.append([result.average, result.stdev])
          else:
              result = %timeit -n 1 -r 1 -o DensityMatrix(qc)
              results_dm.append([result.average, result.stdev])
              result = %timeit -n 1 -r 1 -o AerDensityMatrix(qc)
              results_adm.append([result.average, result.stdev])
```

- Create evaluation quantum circuits at each number of qubits to measure performance
- This experiments took very long time… (about 1 hour)



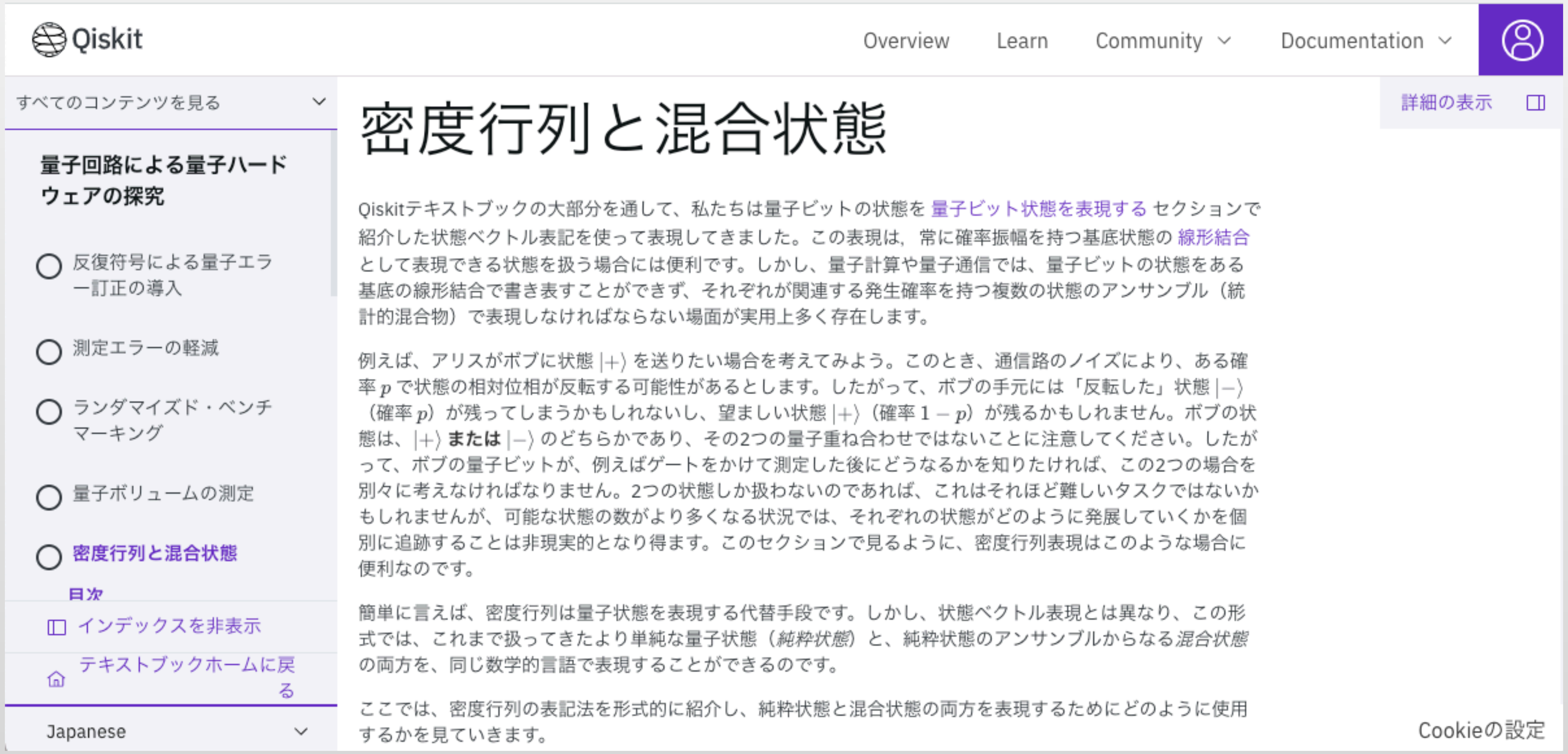
```
[13]: plot_graph()
```

- The result: AerDensityMatrix is about 3x faster than DensityMatrix
- Env: Google Cloud’s n1-highmem-4 instance (Ubuntu 18.04, 4 vCPU, RAM 26GB)

Qiskit Textbook as a by-product

“The Density Matrix and Mixed States” is now available in Japanese:

- I hope that people unfamiliar with density matrices will find it easy to learn and take advantage of AerDensityMatrix



密度行列と混合状態

Qiskitテキストブックの大部分を通して、私たちは量子ビットの状態を 量子ビット状態を表現する セクションで紹介した状態ベクトル表記を使って表現してきました。この表現は，常に確率振幅を持つ基底状態の 線形結合 として表現できる状態を扱う場合には便利です。しかし、量子計算や量子通信では、量子ビットの状態をある基底の線形結合で書き表すことができず、それぞれが関連する発生確率を持つ複数の状態のアンサンブル（統計的混合物）で表現しなければならない場面が実用上多く存在します。

例えば、アリスがボブに状態 $|+\rangle$ を送りたい場合を考えてみよう。このとき、通信路のノイズにより、ある確率 p で状態の相対位相が反転する可能性があるとします。したがって、ボブの手元には「反転した」状態 $|-\rangle$（確率 p）が残ってしまうかもしれないし、望ましい状態 $|+\rangle$（確率 $1-p$）が残るかもしれません。ボブの状態は、$|+\rangle$ **または** $|-\rangle$ のどちらかであり、その2つの量子重ね合わせではないことに注意してください。したがって、ボブの量子ビットが、例えばゲートをかけて測定した後にどうなるかを知りたければ、この2つの場合を別々に考えなければなりません。2つの状態しか扱わないのであれば、これはそれほど難しいタスクではないかもしれませんが、可能な状態の数がより多くなる状況では、それぞれの状態がどのように発展していくかを個別に追跡することは非現実的となり得ます。このセクションで見るように、密度行列表現はこのような場合に便利なのです。

簡単に言えば、密度行列は量子状態を表現する代替手段です。しかし、状態ベクトル表現とは異なり、この形式では、これまで扱ってきたより単純な量子状態（*純粋状態*）と、純粋状態のアンサンブルからなる*混合状態*の両方を、同じ数学的言語で表現することができるのです。

ここでは、密度行列の表記法を形式的に紹介し、純粋状態と混合状態の両方を表現するためにどのように使用するかを見ていきます。