Computer Science and Application 计算机科学与应用, 2022, 12(5), 1227-1236
Published Online May 2022 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2022.125124

Hans汉斯

用cuQuantum框架进行量子计算模拟

邹　铁

雅安开放大学，四川　雅安

收稿日期：2022年4月8日；录用日期：2022年5月4日；发布日期：2022年5月11日

摘　要

量子计算模拟是使用经典计算机和相应框架对量子计算进行模拟，是设计和实现量子算法的有力工具。cuQuantum作为模拟量子计算的框架，为使用GPU进行量子计算模拟提供了重要工具，但在使用该框架时，由于表示法与直观有一定差距，带来了使用上的一些问题，通过研究和学习该框架，以及使用该框架对量子线路的基本操作进行模拟，澄清了框架使用过程中的相关问题，为后续设计和实现量子算法提供基础。

关键词

量子计算，cuQuantum框架，模拟，量子线路

Simulate Quantum Computation Using cuQuantum Framework

Tie Zou

Yaan Open University, Ya'an Sichuan

Received: Apr. 8th, 2022; accepted: May 4th, 2022; published: May 11th, 2022

Abstract

The simulation of quantum computation is a powerful tool for designing and implementing quantum algorithms by using classical computers and corresponding frameworks. The cuQuantum, as a framework for simulating quantum computing, provides an important tool for quantum computing simulation using GPU. However, since there is a certain gap between representation and intuition when using this framework, it brings some problems in the usage. By studying and learning the framework and using the framework to simulate the basic operation of quantum circuits, the related problems in the usage of the framework are clarified, which provides a basis for the subsequent design and implementation of quantum algorithms.

文章引用: 邹铁. 用 cuQuantum 框架进行量子计算模拟[J]. 计算机科学与应用, 2022, 12(5): 1227-1236.
DOI: 10.12677/csa.2022.125124

Keywords

Quantum Computation, cuQuantum, Simulation, Quantum Circuits

1. 引言

量子计算是利用量子力学原理对经典计算模型的一种改进，有望突破强 Church-Turing 论题，实现对复杂计算问题的超多项式级加速[1]。量子计算的提出可以追溯到费曼，但证明量子计算优势的成果当数 Shor 算法[2]——应用此算法可以快速分解大整数，在此前后相继出现了 Simon 算法[3]和 Grover 算法[4]，都表明量子算法比经典算法效率更高。

量子算法要量子计算机才能执行，一般情况下，获取量子计算机过于困难，虽然 IBM 等公司提供了量子计算机的云接口，但开放的接口只是不到 10 个量子比特的量子计算机，对于量子算法特别是量子智能算法的研究无疑是不够的。要想更好设计和实现量子算法，就需要使用模拟的方法，在经典机器上仿真、调试，达到要求后，再使用量子计算机运行，这样设计和实现算法的效率更高。NVIDIA 公司于 2021 年年末推出了 cuQuantum 框架，使用该框架和相应的 GPU 进行量子计算模拟将大大提高效率。

2. 量子计算的基本操作

2.1. 量子比特

在电子计算机中，我们用 0、1 表示计算机的状态，信息被编码为 0、1 所组成的字符串，而相应的 0、1 被称为比特；在量子计算中我们使用叠加态

$$|s\rangle = a|0\rangle + b|1\rangle$$，其中 a、b 为复数，满足 $|a|^2 + |b|^2 = 1$

表示量子计算机的状态，a、b 表示概率幅，其模的平方 $|a|^2$、$|b|^2$ 代表测量后得到状态 $|0\rangle$、$|1\rangle$ 的概率。我们也可将 $|s\rangle$ 用基和坐标形式表示为：

$$|s\rangle = (|0\rangle, |1\rangle)\begin{pmatrix} a \\ b \end{pmatrix}$$

在表示多位量子比特和表示对多位量子比特进行操作时涉及张量积，使用坐标表示法更简洁，我们现在来看看如何表示多位量子比特。

第一种情况，多位量子比特可由单个量子比特组合而成，我们以两位量子比特为例：

$$|s_0 s_1\rangle = |s_0\rangle|s_1\rangle = |s_0\rangle \otimes |s_1\rangle$$

其中

$$|s_0\rangle = (|0\rangle, |1\rangle)\begin{pmatrix} a_0 \\ b_0 \end{pmatrix},\quad |s_1\rangle = (|0\rangle, |1\rangle)\begin{pmatrix} a_1 \\ b_1 \end{pmatrix}$$

可以得出

$$|s_0s_1\rangle=|s_0\rangle|s_1\rangle=|s_0\rangle\otimes|s_1\rangle=\left[(|0\rangle,|1\rangle)\begin{pmatrix}a_0\\b_0\end{pmatrix}\right]\otimes\left[(|0\rangle,|1\rangle)\begin{pmatrix}a_1\\b_1\end{pmatrix}\right]$$

$$=\left[(|0\rangle,|1\rangle)\otimes(|0\rangle,|1\rangle)\right]\left[\begin{pmatrix}a_0\\b_0\end{pmatrix}\otimes\begin{pmatrix}a_1\\b_1\end{pmatrix}\right]=(|00\rangle,|01\rangle,|10\rangle,|11\rangle)\begin{pmatrix}a_0a_1\\a_0b_1\\b_0a_1\\b_0b_1\end{pmatrix}.$$

第二种情况，多位量子比特处于量子纠缠态，这时的多位量子比特不能写为单个量子比特的张量积形式，我们以如下处于纠缠态的量子比特为例：

$$|s_0s_1\rangle=\frac{1}{\sqrt{2}}|00\rangle+\frac{1}{\sqrt{2}}|11\rangle$$

由于这种形式的量子比特无法写为两个独立的量子比特张量积的形式，用基与坐标方式可将其表示为：

$$|s_0s_1\rangle=(|00\rangle,|01\rangle,|10\rangle,|11\rangle)\begin{pmatrix}\frac{1}{\sqrt{2}}\\0\\0\\\frac{1}{\sqrt{2}}\end{pmatrix}.$$

2.2. 量子门

在经典计算中，逻辑门的作用是对比特进行逻辑运算，在量子计算中，我们要对量子比特进行操作就需要有量子门，将量子门作用于量子比特，即可完成相应的量子计算操作，而量子门对量子比特的作用一般都可视为线性变换。

对于单量子比特的量子操作，我们可使用线性代数中的表示法，假设我们有一个幺正算符 $\boldsymbol{U}$，将此算符作用于一个单量子比特，就相当于将此幺正算符对应的矩阵乘上概率幅向量，具体表示如下：

$$\boldsymbol{U}|s\rangle=(|0\rangle,|1\rangle)U\begin{pmatrix}a\\b\end{pmatrix}$$

对于多量子比特构成系统的量子操作，通常情况下可仿照单量子比特量子操作的表示法表示，假设有一个幺正算符 $\boldsymbol{U}$，将其作用于一个双量子比特，可以表示为：

$$\boldsymbol{U}|s_0s_1\rangle=(|00\rangle,|01\rangle,|10\rangle,|11\rangle)U\begin{pmatrix}a_0a_1\\a_0b_1\\b_0a_1\\b_0b_1\end{pmatrix}$$

这种表示法适用于所有多量子比特系统，当多量子比特系统中量子操作相互独立，即可以将多量子操作拆分为几个单量子操作时，或是由多个单量子操作组合为一个多量子操作时，我们可以用张量积的方式表示，我们还是以双量子比特系统为例，在第一个量子比特$|s_1\rangle$上施加了一个 $\boldsymbol{U_1}$ 操作，在第二个量子比特$|s_2\rangle$上施加一个 $\boldsymbol{U_2}$ 操作，即

$$\boldsymbol{U}|s_1s_2\rangle=\left(\boldsymbol{U}_1|s_1\rangle\right)\left(\boldsymbol{U}_2|s_2\rangle\right)=\left(\boldsymbol{U}_1|s_1\rangle\right)\otimes\left(\boldsymbol{U}_2|s_2\rangle\right)=\left[\left(|0\rangle,|1\rangle\right)U_1\begin{pmatrix}a_1\\b_1\end{pmatrix}\right]\otimes\left[\left(|0\rangle,|1\rangle\right)U_2\begin{pmatrix}a_2\\b_2\end{pmatrix}\right]$$

$$=\left[\left(|0\rangle,|1\rangle\right)\otimes\left(|0\rangle,|1\rangle\right)\right]\left(U_1\otimes U_2\right)\left[\begin{pmatrix}a_1\\b_1\end{pmatrix}\otimes\begin{pmatrix}a_2\\b_2\end{pmatrix}\right]$$

$$=\left(|00\rangle,|01\rangle,|10\rangle,|11\rangle\right)\left(U_1\otimes U_2\right)\begin{pmatrix}a_1a_2\\a_1b_2\\b_1a_2\\b_1b_2\end{pmatrix}$$

故上述操作可写为两个操作对应矩阵的张量积表示。这种方式的操作合成可以推广到任意量子比特组合构成的量子线路的情况，这里不再讨论。

常用的量子门及其矩阵表示如表 1 和表 2 所示。

Table 1. Common single qubit quantum gates and their matrix representation [5]
表 1. 常用单比特量子门及其矩阵表示[5]

量子门	矩阵表示
Hadamard 门	$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}$
X 门	$\begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}$
Y 门	$\begin{bmatrix}0 & -i\\i & 0\end{bmatrix}$
Z 门	$\begin{bmatrix}1 & 0\\0 & -1\end{bmatrix}$
Phase 门(S 门)	$\begin{bmatrix}1 & 0\\0 & i\end{bmatrix}$
π/8 门(T 门)	$\begin{bmatrix}1 & 0\\0 & e^{i\pi/4}\end{bmatrix}$

Table 2. Common multi-qubit quantum gates and their matrix representation [5]
表 2. 常用多比特量子门及其矩阵表示[5]

量子门	矩阵表示
CNOT 门	$\begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 0 & 1\\0 & 0 & 1 & 0\end{bmatrix}$
Swap 门	$\begin{bmatrix}1 & 0 & 0 & 0\\0 & 0 & 1 & 0\\0 & 1 & 0 & 0\\0 & 0 & 0 & 1\end{bmatrix}$

Continued

Controlled-Z 门	$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&-1\end{bmatrix}$
Controlled-Phase 门	$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&i\end{bmatrix}$
Toffoli 门	$\begin{bmatrix}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&1\\0&0&0&0&0&0&1&0\end{bmatrix}$
Fredkin 门(Controlled-Swap)	$\begin{bmatrix}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&0&0&1&0\\0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&1\end{bmatrix}$

2.3. 测量

要得到最终结果，我们需要将量子比特变换为比特，从而沟通量子系统和经典系统，这样才能实现量子计算的目的。在量子力学中，我们对处于量子态的粒子进行测量，其量子态就会按照概率幅所确定的概率坍缩为某一种状态，而之后都以这种状态存在。在量子计算中，对量子比特进行测量，其量子比特就将以概率幅模的平方为概率变换为对应的比特。要对测量操作进行模拟，在模拟过程中就需要满足：以概率幅确定的概率来随机决定测量后的状态；测量后的状态一直保持不变，直到下次变换。在量子线路中，我们常常将测量操作作为量子计算的最终步骤。

在模拟测量操作时，我们先要获得量子比特$|s\rangle = a|0\rangle + b|1\rangle$坍缩为某一状态$|m\rangle$的概率($m$ 取 0、1 两值)：

$$p(m) = \left\langle s \left| \frac{1}{2}\left[I + (-1)^m Z \right] \right| s \right\rangle$$

接着用随机数生成器产生一个[0, 1)区间的数 q，比较 q 与 $p(m)$ 的关系：若 $q < p(m)$，则取状态$|m\rangle$；否则，取状态$|1-m\rangle$。测量之后量子比特$|s\rangle = a|0\rangle + b|1\rangle$将坍缩为

$$\frac{\frac{1}{2}\left[I+(-1)^m Z\right]|s\rangle}{\sqrt{p(m)}}$$

3. cuQuantum 框架[6]

从上面的描述可以看出，量子计算主要使用矩阵，而这正是 GPU 处理的强项，使用 GPU 模拟量子计算是较好的选择。NVIDIA 公司于 2021 年年底推出的 cuQuantum，正是基于 GPU 的量子计算模拟框架，它分为 cuStateVec 和 cuTensornet 两个部分，cuStateVec 是处理量子门运算的库，相对比较底层，也较为灵活，我们以下的量子计算模拟主要使用这个库；而 cuTensornet 主要负责张量网络计算，本文不涉及此库的相关内容。

由于 cuQuantum 框架是基于 GPU 进行量子计算模拟的框架，它建立在已有的 CUDA 基础上，其 CPU、内存与 GPU 的数据传输与管理操作都由 CUDA 提供，cuQuantum 仅提供模拟量子计算部分的 API，本文第 4 节将使用 cuStateVec 库的部分 API 进行量子计算模拟。

4. 量子计算模拟

量子计算模拟是指使用经典计算设备对量子计算机进行模拟，主要模拟量子线路和量子操作。现阶段量子算法的描述以量子线路为主，量子线路是由量子门组成，为设计和实现量子算法提供便利，故在 cuQuantum 中所有操作都以量子门形式给出。本文的量子计算模拟环境如表 3 所示。

Table 3. Parameters of the simulation environment in this paper
表 3. 本文模拟环境参数

名称	参数
操作系统	Ubuntu Linux 20.04 64-bit
CPU	Intel Core i7
GPU	Nvidia rtx3070ti 8G
函数库(框架)	cuQuantum 0.1.0+Cuda 7.2

4.1. 量子比特的 cuQuantum 框架表示

首先我们来看看如何用 cuQuantum 框架表示一个量子比特，假设有一个量子比特 $|s\rangle = 2.3|0\rangle + 1.4i|1\rangle$，用框架可表示为：

cuDoubleComplex h_sv[] = {{2.3, 0.0}, {0.0, 1.4}};

这里注意系数，根据定义一个量子比特在各个分量的系数都为复数，在 cuQuantum 中我们用{x, y}来表示复数 x + yi，故系数 2.3 表示为{2.3, 0.0}，系数 1.4i 表示为{0.0, 1.4}。

接下来我们来看如何表示多个量子比特，一般来说，多个量子比特可表示为单量子比特的张量积，如第 2 节中的描述，但在 cuQuantum 中并非以张量积的方式来表示多个量子比特，而是用一个系数向量来表示，并以小端序进行编码，譬如：我们有一个 2 位的量子比特 $|s_0 s_1\rangle = 0.6|00\rangle + |01\rangle + 0.3|10\rangle + 0.5|11\rangle$，用框架表示为：

cuDoubleComplex h_sv[] = {{0.6, 0.0}, {0.3, 0.0}, {1, 0.0}, {0.5, 0.0}};

虽然在框架中，系数可以是任意的复数，但在实际表述时，应当将其归一化，一种常用的做法是仅

对最终结果进行归一化处理。为了方便起见，以下论述中所用示例我们均没有进行归一化处理(cuQuantum 框架没有强行要求归一化)。

4.2. 对单量子比特操作的模拟

在量子计算中最常用的量子门是 Hadamard 门，Pauli 门，Phase 门以及 π/8 门，如第 1 节所述，Pauli 门包含 3 个分别用 X、Y、Z 来表示。(以下描述中，*表示乘，sqrt 表示平方根)

首先我们来看 Hadamard 门的 cuQuantum 框架表示：

```
cuDoubleComplex matrix_H[] = {{1.0*sqrt(2)*0.5, 0.0}, {1.0*sqrt(2)*0.5, 0.0},
                              {1.0*sqrt(2)*0.5, 0.0}, {−1.0*sqrt(2)*0.5, 0.0}};
```

现在我们将此 Hadamard 门应用于一个单量子比特 $|s\rangle = 2.3|0\rangle + 1.4|1\rangle$，用 cuQuantum 框架对此量子比特进行操作，可表示为：

```
custatevecApplyMatrix(handle, d_sv, CUDA_C_64F, 1, matrix_H, CUDA_C_64F,
    CUSTATEVEC_MATRIX_LAYOUT_ROW, 0, {0}, 1, {},
    0, nullptr, CUSTATEVEC_COMPUTE_64F,
    extraWorkspace, extraWorkspaceSizeInBytes);
```

d_sv 是显存中的 h_sv 表示，可以直接用相应函数将 h_sv 复制到 d_sv 中，由于本文不讨论显卡的设备管理问题(如设备空间的分配、GPU 与 CPU 的数据传输等问题)，故具体细节不在这里描述。上述函数使用后，d_sv 数组就是计算结果(上例中 d_sv = {{2.616295, 0.0}, {0.636396, 0.0}}，按照 cuQuantum 的表示法，可以看出最终结果为 $2.616295|0\rangle + 0.636396|1\rangle$)。

其他几个单量子比特操作我们总结如表 4 所示。

Table 4. Operation simulation of single qubit quantum gate
表 4. 单比特量子门的操作模拟

单量子比特操作	cuQuantum 表示
X	{{0.0, 0.0}, {1.0, 0.0}, {1.0, 0.0}, {0.0, 0.0}}
Y	{{0.0, 0.0}, {0.0, −1.0}, {0.0, 1.0}, {0.0, 0.0}}
Z	{{1.0, 0.0}, {0.0, 0.0}, {0.0, 0.0}, {−1.0, 0.0}}
S	{{1.0, 0.0}, {0.0, 0.0}, {0.0, 0.0}, {0.0, 1.0}}
T	{{1.0, 0.0}, {0.0, 0.0}, {0.0, 0.0}, {1.0*sqrt(2)*0.5, −1.0*sqrt(2)*0.5}}

4.3. 对多量子比特操作的模拟

一般情况下，多量子比特是多个量子比特的组合，其表示法可以被看作多个量子比特的张量积，其对应操作可以表示为多个矩阵的张量积，第 1 节中有详细描述，用 cuQuantum 框架中对单量子比特操作模拟的做法，对大多数多量子比特的操作模拟而言都不存在问题，只需要记住多量子比特的 cuQuantum 表示法，下面以对两个量子比特的操作为例：

假设有 $|s_0 s_1\rangle = 0.6|00\rangle + |01\rangle + 0.3|10\rangle + 0.5|11\rangle$，我们要对第一个量子比特位进行 X 操作，对第二个量子比特位进行 Y 操作，则可以计算 X 与 Y 的张量积，将此张量积作用于上述量子比特，即：

$$(X\otimes Y)|s_0s_1\rangle=(|00\rangle\ |01\rangle\ |10\rangle\ |11\rangle)\begin{bmatrix}0&0&0&-i\\0&0&i&0\\0&-i&0&0\\i&0&0&0\end{bmatrix}\begin{pmatrix}0.6\\1\\0.3\\0.5\end{pmatrix}$$
$$=-0.5i|00\rangle+0.3i|01\rangle-i|10\rangle+0.6i|11\rangle$$

在使用 cuQuantum 框架时，要将 X、Y 的位置进行交换，得到

$$Y\otimes X=\begin{bmatrix}0&0&0&-i\\0&0&i&0\\0&-i&0&0\\i&0&0&0\end{bmatrix}$$

再将其作用于$|s_0s_1\rangle$的 cuQuantum 表示法，即此时的量子操作将表示为

cuDoubleComplex matrix[] = {{0.0, 0.0}, {0.0, 0.0}, {0.0, 0.0}, {0.0, −1.0},
{0.0, 0.0}, {0.0, 0.0}, {0.0, −1.0}, {0.0, 0.0},
{0.0, 0.0}, {0.0, 1.0}, {0.0, 0.0}, {0.0, 0.0},
{0.0, 1.0}, {0.0, 0.0}, {0.0, 0.0}, {0.0, 0.0}};

运用 custatevecApplyMatrix 函数后的结果为{{0.0, −0.5}, {0.0, −1.0}, {0.0, 0.3}, {0.0, 0.6}}，这里的结果是 cuQuantum 表示法，它表示 $-0.5i|00\rangle+0.3i|01\rangle-1.0i|10\rangle+0.6i|11\rangle$。

对于 Swap 门，在使用框架时，不需要作改变即可使用，即 Swap 门的变换矩阵可直接表示为

cuDoubleComplex matrix[] = {{1.0, 0.0}, {0.0, 0.0}, {0.0, 0.0}, {0.0, 0.0},
{0.0, 0.0}, {0.0, 0.0}, {1.0, 0.0}, {0.0, 0.0},
{0.0, 0.0}, {1.0, 0.0}, {0.0, 0.0}, {0.0, 0.0},
{0.0, 0.0}, {0.0, 0.0}, {0.0, 0.0}, {1.0, 0.0}};

在量子计算中，受控操作是很重要的一类多量子比特操作，cuQuantum 框架中通过设置控制位的方式，简化受控操作的矩阵表示，这里以 Toffoli 门为例，我们知道 Toffoli 门表示的量子操作是“如果两个控制位同时为 1，则对第三个位取反，否则不改变第三个位”，对一个量子位进行取反的操作用 cuQuantum 框架可表示为

cuDoubleComplex matrix[] = {{0.0, 0.0}, {1.0, 0.0}, {0.0, 0.0}, {1.0, 0.0}};

这时我们把控制位定为 0 和 1，目标位定为 2，运用 custatevecApplyMatrix 函数，这个操作可写为：

custatevecApplyMatrix(handle, d_sv, CUDA_C_64F, 3, matrix, CUDA_C_64F,
CUSTATEVEC_MATRIX_LAYOUT_ROW, 0, {2}, 1, {0, 1},
2, nullptr, CUSTATEVEC_COMPUTE_64F,extraWorkspace, extraWorkspaceSizeInBytes);

d_sv 在函数调用时传入

cuDoubleComplex h_sv[] = {{0.0, 0.0}, {0.0, 0.1}, {0.1, 0.1}, {0.1, 0.2},
{0.2, 0.2}, {0.3, 0.3}, {0.3, 0.4}, {0.4, 0.5}};

传入的参数用标准法表示为

$$(0.1i)|100\rangle+(0.1+0.1i)|010\rangle+(0.1+0.2i)|110\rangle+(0.2+0.2i)|001\rangle$$
$$+(0.3+0.3i)|101\rangle+(0.3+0.4i)|011\rangle+(0.4+0.5i)|111\rangle,$$

运算后的结果为

{{0.0, 0.0}, {0.0, 0.1}, {0.1, 0.1}, {0.4, 0.5}, {0.2, 0.2}, {0.3, 0.3}, {0.3, 0.4}, {0.1, 0.2}}，

可用标准形式写为：

$$(0.1i)|100\rangle+(0.1+0.1i)|010\rangle+(0.4+0.5i)|110\rangle+(0.2+0.2i)|001\rangle$$
$$+(0.3+0.3i)|101\rangle+(0.3+0.4i)|011\rangle+(0.1+0.2i)|111\rangle.$$

4.4. 对测量操作的模拟

在量子计算中，要得到最终结果，就要对量子线路进行测量，在第 1 节中我们从概念上描述了测量操作，现在来看用 cuQuantum 框架如何实现测量操作。

cuQuantum 框架提供了多个关于测量的函数，最常用的是 custatevecMeasureOnZBasis 函数，它使用 Pauli-Z 矩阵为基，可以达到模拟量子计算中测量操作的效果，下面以$|s\rangle=(0.5+0.1i)|0\rangle+(0.4+0.3i)|1\rangle$为例，我们首先准备一个数组 h_sv：

```
cuDoubleComplex h_sv[]= {{0.5, 0.1}, {0.4, 0.3}};
```

接下来产生一个[0, 1)中随机数 randnum：

```
srand(time(0));
const double randnum = rand()/(RAND_MAX+1.0);
```

调用 custatevecMeasureOnZBasis 函数：

```
custatevecMeasureOnZBasis(
handle, d_sv, CUDA_C_64F, nIndexBits, &parity, basisBits, nBasisBits,
randnum, CUSTATEVEC_COLLAPSE_NORMALIZE_AND_ZERO);
```

就完成了对$|s\rangle$的测量，测量结果在 partiy 变量中，测量之后的$|s\rangle$用 d_sv 数组表示。要进一步使用测量后的结果直接使用 d_sv 数组即可，这也就模拟了测量后量子态的变化。表 5 显示了对$|s\rangle$独立测量 5 次，每次测量的结果。(在程序设计时，不能连续测量，若连续测量——调用多次上述的测量函数，其结果在第一次测量后就已经确定，后 4 次其结果都为第 1 次的测量后结果，这也就是测量操作的特点。

Table 5. Measure $|s\rangle=(0.5+0.1i)|0\rangle+(0.4+0.3i)|1\rangle$
表 5. 对$|s\rangle=(0.5+0.1i)|0\rangle+(0.4+0.3i)|1\rangle$进行测量

次数	测量结果	测量后的$\|s\rangle$
第 1 次	1	$(0.800000+0.600000i)\|1\rangle$
第 2 次	0	$(0.980581+0.196116i)\|0\rangle$
第 3 次	1	$(0.800000+0.600000i)\|1\rangle$
第 4 次	0	$(0.980581+0.196116i)\|0\rangle$
第 5 次	0	$(0.980581+0.196116i)\|0\rangle$

5. 结论与展望

本文研究了使用 cuQuantum 框架进行量子计算模拟的方法，运用框架对常用的量子操作进行了模拟。下一步研究主要集中在使用 cuQuantum 框架实现和仿真量子计算智能和量子强化学习算法上。

基金项目

本文系四川开放大学科研课题“基于量子计算智能的组合优化问题研究”(课题编号：KTGCJS2021002Y)阶段性成果。

参考文献

[1] Deutsch, D. (1985) Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer. *Proceedings of the Royal Society of London. Series A*, **400**, 97. https://doi.org/10.1098/rspa.1985.0070

[2] Shor, P.W. (1997) Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Journal on Computing*, **26**, 1484-1509. https://doi.org/10.1137/S0097539795293172

[3] Simon. D.R. (1997) On the Power of Quantum Computation. *SIAM Journal on Computing*, **26**, 1474-1483. https://doi.org/10.1137/S0097539796298637

[4] Grover. L. (1996) A Fast Quantum Mechanical Algorithm for Database Search. *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, Philadelphia, July 1996, 212-219. https://doi.org/10.1145/237814.237866

[5] Nielsen, M.A. and Chuang, I.L. (2015) Quantum Computation and Quantum Information. 10th Anniversary Edition. Cambridge University Press, Cambridge.

[6] NVIDIA Corporation (2022) The Cuquantum Documents. https://docs.nvidia.com/cuda/cuquantum/#