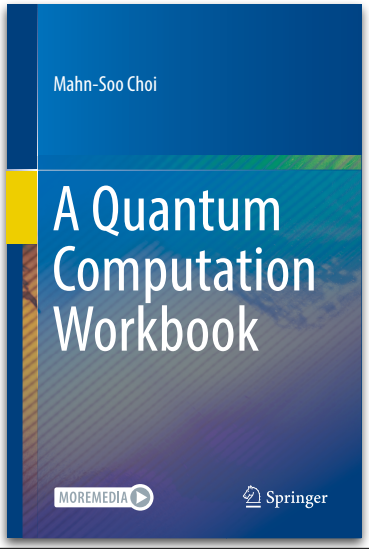
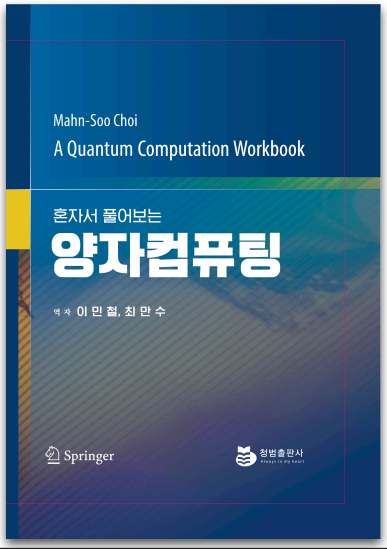
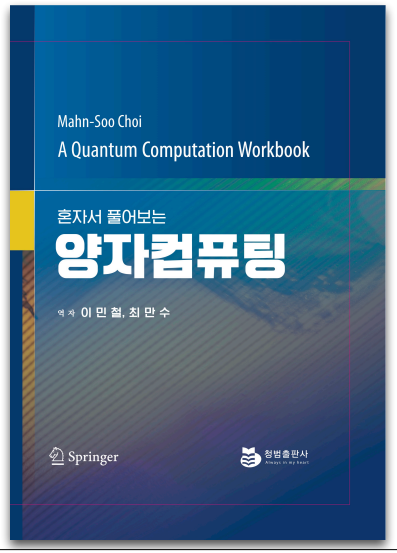
QUANTUM PLAYBOOK

A COLLECTION OF EXAMPLES & DEMOS

최만수 (고려대 물리학과)





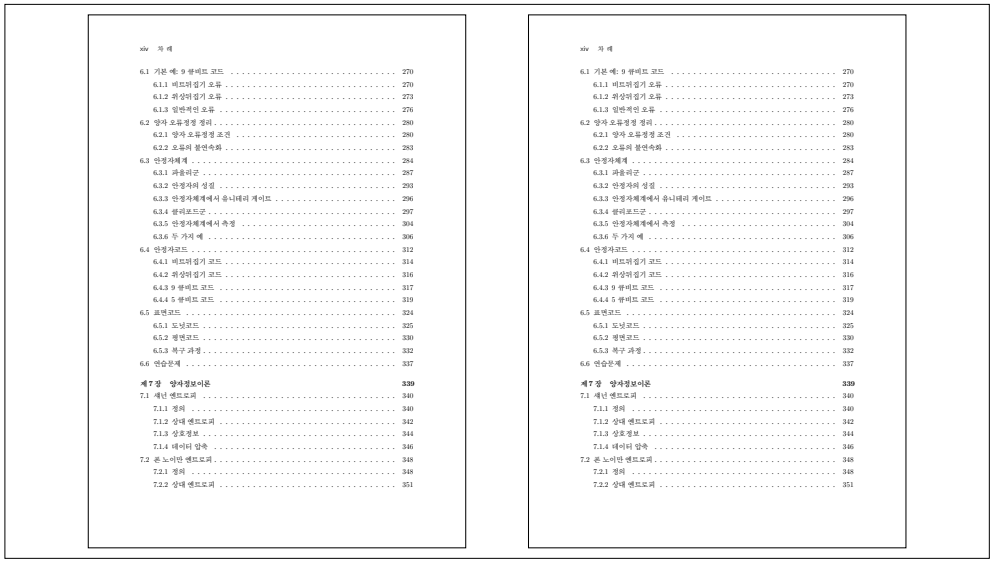


차 례

머리말 vii
저자소개 ix
차 례 xi
제 1 장 양자역학의 기본 가정 1
1.1 양자상태 . . . 2
1.1.1 순수상태 . . . 2
1.1.2 혼합상태 . . . 8
1.2 양자상태의 시간진행 . . . 17
1.2.1 유니태리 동역학 . . . 17
1.2.2 양자집합 동역학 . . . 21
1.3 양자상태 측정하기 . . . 21
1.3.1 투영측정 . . . 22
1.3.2 일반화된 측정 . . . 27
1.4 연습문제 . . . 29
제 2 장 양자계산 둘러보기 33
2.1 단일 큐비트 게이트 . . . 34
2.1.1 파울리 게이트 . . . 34
2.1.2 하다마드 게이트 . . . 39
2.1.3 회전 . . . 42
2.2 두 큐비트 게이트 . . . 46
2.2.1 CNOT, CZ 및 SWAP . . . 47
2.2.2 제어 유니태리 게이트 . . . 58

2.2.3 일반 유니태리 게이트 . . . 64
2.3 다중제어 유니태리 게이트 . . . 72
2.3.1 그레이 코드 방법 . . . 73
2.3.2 다중제어 NOT 게이트 . . . 75
2.4 보편적 양자계산 . . . 81
2.5 측정의 계산모형 . . . 87
2.6 연습문제 . . . 90
제 3 장 양자컴퓨터 구현하기 95
3.1 양자비트 . . . 96
3.2 동적 방식 . . . 99
3.2.1 단일 큐비트 게이트 구현하기 . . . 99
3.2.2 CNOT 구현하기 . . . 105
3.3 기하학적/위상학적 방식 . . . 109
3.3.1 간단한 모형 . . . 110
3.3.2 기하학적 위상 . . . 116
3.4 측정기반 방식 . . . 119
3.4.1 단일 큐비트 회전 . . . 123
3.4.2 CNOT 게이트 . . . 126
3.4.3 그래프 상태 . . . 129
3.5 연습문제 . . . 132
제 4 장 양자 알고리즘 134
4.1 양자원격이동 . . . 135
4.1.1 원형의 비국소성 . . . 136
4.1.2 양자원격이동 구현하기 . . . 140
4.2 양자 관측 알고리즘 . . . 143
4.2.1 양자 오라클 . . . 144
4.2.2 도이치-조사 알고리즘 . . . 150
4.2.3 번스타인-바지라니 알고리즘 . . . 152
4.2.4 사이먼 알고리즘 . . . 153
4.3 양자 푸리에 변환 . . . 158
4.3.1 정의 및 물리적 의미 . . . 158
4.3.2 양자역학적으로 구현하기 . . . 160

4.3.3 준고전적으로 구현하기 . . . 164
4.4 양자위상추정 . . . 168
4.4.1 정의 . . . 168
4.4.2 구현하기 . . . 170
4.4.3 정확도 . . . 172
4.4.4 존 노이만 측정 시늉내기 . . . 173
4.5 응용하기 . . . 175
4.5.1 주기찾기 알고리즘 . . . 175
4.5.2 좌수찾기 알고리즘 . . . 183
4.5.3 양자 인수분해 알고리즘 . . . 185
4.5.4 양자 검색 알고리즘 . . . 186
4.6 연습문제 . . . 196
제 5 장 양자얽음과 경계점 198
5.1 경계점은 어떻게 발생하는가? . . . 199
5.2 양자연산 . . . 206
5.2.1 크라우스 표현 . . . 207
5.2.2 최 동형사상 . . . 216
5.2.3 유니태리 표현 . . . 220
5.2.4 간단한 예들 . . . 221
5.3 양자연산과 측정 . . . 226
5.4 양자 으뜸 방정식 . . . 226
5.4.1 유도하기 . . . 229
5.4.2 간단한 예들 . . . 230
5.4.3 풀이 방법 . . . 233
5.4.4 간단한 예들을 풀어보기 . . . 240
5.5 양자상태 사이의 거리 재기 . . . 245
5.5.1 노름과 거리 . . . 246
5.5.2 힐베르트-슈미트 노름과 대각합 노름 . . . 247
5.5.3 힐베르트-슈미트 거리와 대각합 거리 . . . 253
5.5.4 충실도 . . . 257
5.6 연습문제 . . . 263
제 6 장 양자 오류정정 코드 269

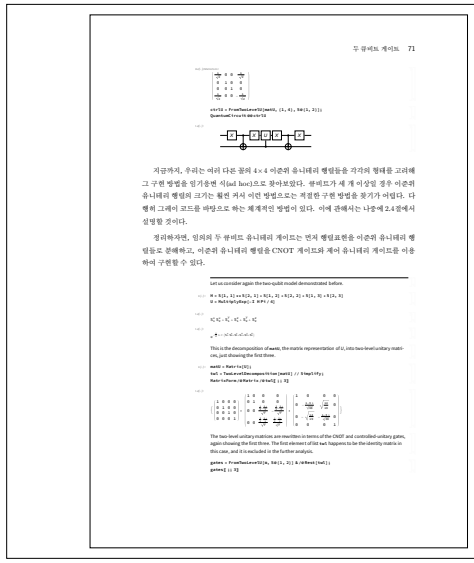


6.1 기본 예: 9 큐비트 코드 270
6.1.1 비트뒤집기 오류 270
6.1.2 위상뒤집기 오류 273
6.1.3 일반적인 오류 276
6.2 양자 오류정정 정리 280
6.2.1 양자 오류정정 조건 280
6.2.2 오류의 불연속화 283
6.3 안정자체계 284
6.3.1 파울리군 287
6.3.2 안정자의 성질 293
6.3.3 안정자체계에서 유니테리 게이트 296
6.3.4 클리포드군 297
6.3.5 안정자체계에서 측정 304
6.3.6 두 가지 예 306
6.4 안정자코드 312
6.4.1 비트뒤집기 코드 314
6.4.2 위상뒤집기 코드 316
6.4.3 9 큐비트 코드 317
6.4.4 5 큐비트 코드 319
6.5 표면코드 324
6.5.1 도넛코드 325
6.5.2 평면코드 330
6.5.3 복구 과정 332
6.6 연습문제 337

제 7 장 양자정보이론 339
7.1 섀넌 엔트로피 340
7.1.1 정의 340
7.1.2 상대 엔트로피 342
7.1.3 상호정보 344
7.1.4 데이터 압축 346
7.2 폰 노이만 엔트로피 348
7.2.1 정의 348
7.2.2 상대 엔트로피 351

6.1 기본 예: 9 큐비트 코드 270
6.1.1 비트뒤집기 오류 270
6.1.2 위상뒤집기 오류 273
6.1.3 일반적인 오류 276
6.2 양자 오류정정 정리 280
6.2.1 양자 오류정정 조건 280
6.2.2 오류의 불연속화 283
6.3 안정자체계 284
6.3.1 파울리군 287
6.3.2 안정자의 성질 293
6.3.3 안정자체계에서 유니테리 게이트 296
6.3.4 클리포드군 297
6.3.5 안정자체계에서 측정 304
6.3.6 두 가지 예 306
6.4 안정자코드 312
6.4.1 비트뒤집기 코드 314
6.4.2 위상뒤집기 코드 316
6.4.3 9 큐비트 코드 317
6.4.4 5 큐비트 코드 319
6.5 표면코드 324
6.5.1 도넛코드 325
6.5.2 평면코드 330
6.5.3 복구 과정 332
6.6 연습문제 337

제 7 장 양자정보이론 339
7.1 섀넌 엔트로피 340
7.1.1 정의 340
7.1.2 상대 엔트로피 342
7.1.3 상호정보 344
7.1.4 데이터 압축 346
7.2 폰 노이만 엔트로피 348
7.2.1 정의 348
7.2.2 상대 엔트로피 351




```
ctrl12 = FromTwoLevel[mat0, {1, 4}, Id[2, 2]];
QuantumCircuitGate[ctrl2]
```

지금까지, 우리는 여러 다른 꼴의 4×4 이준위 유니테리 행렬들을 각각의 형태를 고려해 그 구현 방법을 임기응변 식(ad hoc)으로 찾아보았다. 큐비트가 세 개 이상일 경우 이준위 유니테리 행렬의 크기는 훨씬 커서 이런 방법으로는 적절한 구현 방법을 찾기가 어렵다. 다행히 그래이 코드를 바탕으로 하는 체계적인 방법이 있다. 이에 관해서는 나중에 2.4절에서 설명할 것이다.

정리하자면, 임의의 두 큐비트 유니테리 게이트는 먼저 행렬표현을 이준위 유니테리 행렬들로 분해하고, 이준위 유니테리 행렬을 CNOT 게이트와 제어 유니테리 게이트들 이용하여 구현할 수 있다.

Let us consider again the two-qubit model demonstrated before.

```
M = R[1, 2] ** R[2, 3] ** R[1, 2] ** R[2, 3] ** R[1, 3] ** R[2, 3]
U = MultiplyDagg[.I MPM / 4]
```

$$U_1^\dagger U_2^\dagger \cdots U_7^\dagger = U_7^\dagger \cdots U_1^\dagger$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

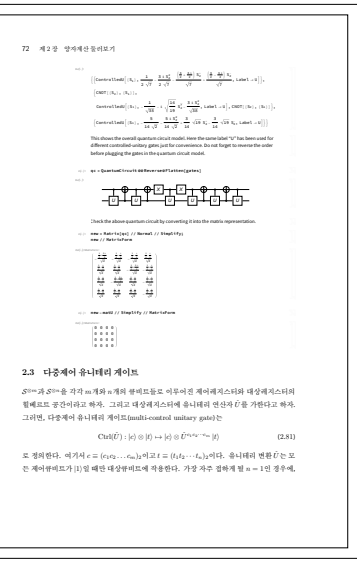
This is the decomposition of **mat0**, the matrix representation of U, into two-level unitary matrices, just showing the first three.

```
mat0 = Matrix[U];
tw1 = TwoLevelDecompositon[mat0] // Simplify;
Matrix/@Matrix/@tw1[[ 1 ;; 3]]
```

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

The two-level unitary matrices are rewritten in terms of the CNOT and controlled-unitary gates, again showing the first three. The first element of list **tw1** happens to be the identity matrix in this case, and it is included in the further analysis.

```
gates = FromTwoLevel[m, Id[2, 2]] & /@Rest[tw1];
gates[[ 1 ;; 3]]
```



This shows the overall quantum circuit model. Here the same label "U" has been used for different controlled-unitary gates just for convenience. Do not forget to reverse the order before plugging the gates in the quantum circuit model.

```
qc = QuantumCircuitOperator[gates]
```

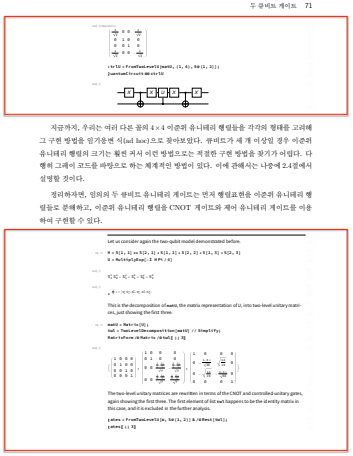
Check the above quantum circuit by converting it into the matrix representation.

```
new = Matrix[qc] // Normal // Simplify;
new // MatrixForm
```

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

```
new - mat0 // Simplify // MatrixForm
```

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$




```
ctrl12 = FromTwoLevel[mat0, {1, 4}, Id[2, 2]];
[QuantumCircuitGate/@ctrl2
```

지금까지, 우리는 여러 다른 꼴의 4×4 이준위 유니테리 행렬들을 각각의 형태를 고려해 그 구현 방법을 임기응변 식(ad hoc)으로 찾아보았다. 큐비트가 세 개 이상일 경우 이준위 유니테리 행렬의 크기는 훨씬 커서 이런 방법으로는 적절한 구현 방법을 찾기가 어렵다. 다행히 그래이 코드를 바탕으로 하는 체계적인 방법이 있다. 이에 관해서는 나중에 2.4절에서 설명할 것이다.

정리하자면, 임의의 두 큐비트 유니테리 게이트는 먼저 행렬표현을 이준위 유니테리 행렬들로 분해하고, 이준위 유니테리 행렬을 CNOT 게이트와 제어 유니테리 게이트들 이용하여 구현할 수 있다.

Let us consider again the two-qubit model demonstrated before.

```
M = R[1, 2] ** R[2, 3] ** R[1, 2] ** R[2, 3] ** R[1, 3] ** R[2, 3]
U = MultiplyDagg[.I MPM / 4]
```

$$U_1^\dagger U_2^\dagger \cdots U_7^\dagger = U_7^\dagger \cdots U_1^\dagger$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

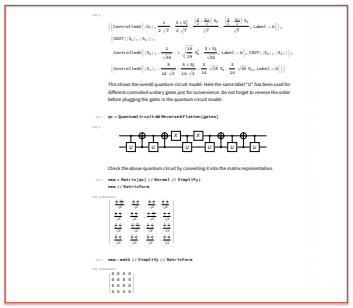
This is the decomposition of **mat0**, the matrix representation of U, into two-level unitary matrices, just showing the first three.

```
mat0 = Matrix[U];
tw1 = TwoLevelDecompositon[mat0] // Simplify;
Matrix/@Matrix/@tw1[[ 1 ;; 3]]
```

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

The two-level unitary matrices are rewritten in terms of the CNOT and controlled-unitary gates, again showing the first three. The first element of list **tw1** happens to be the identity matrix in this case, and it is included in the further analysis.

```
gates = FromTwoLevel[m, Id[2, 2]] & /@Rest[tw1];
gates[[ 1 ;; 3]]
```



This shows the overall quantum circuit model. Here the same label "U" has been used for different controlled-unitary gates just for convenience. Do not forget to reverse the order before plugging the gates in the quantum circuit model.

```
qc = QuantumCircuitOperator[gates]
```

Check the above quantum circuit by converting it into the matrix representation.

```
new = Matrix[qc] // Normal // Simplify;
new // MatrixForm
```

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

```
new - mat0 // Simplify // MatrixForm
```

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

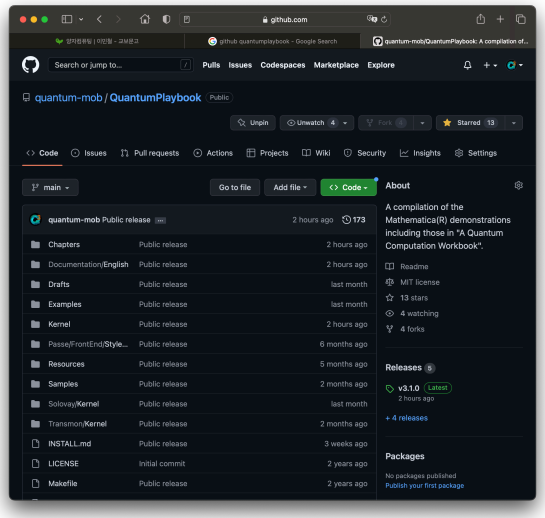
2.3 다중제어 유니테리 게이트

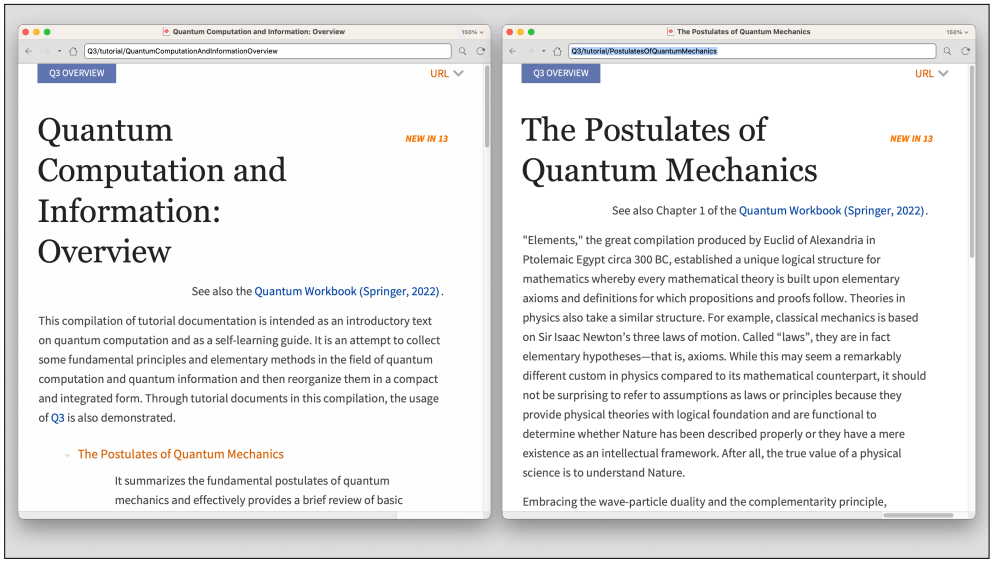
$\mathcal{S}^{\mathbb{C}m}$과 $\mathcal{S}^{\mathbb{C}n}$을 각각 m개와 n개의 큐비트들로 이루어진 제어레지스터와 대상레지스터의 힐베르트 공간이라고 하자. 그리고 대상레지스터에 유니테리 연산자 $\hat{U}$를 가한다고 하자. 그러면, 다중제어 유니테리 게이트(multi-control unitary gate)는

$$\mathrm{Ctrl}(\hat{U}) : |c\rangle \otimes |t\rangle \mapsto |c\rangle \otimes \hat{U}^{c_1 c_2 \cdots c_m} |t\rangle \tag{2.81}$$

로 정의한다. 여기서 $c = (c_1 c_2 \ldots c_m)_2$이고 $t = (t_1 t_2 \cdots t_n)_2$이다. 유니테리 변환 $\hat{U}$는 모든 제어큐비트가 $|1\rangle$일 때만 대상큐비트에 작용한다. 가장 자주 접하게 될 $n = 1$인 경우에,

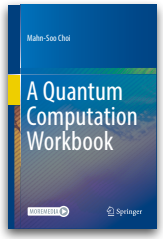
Quantum Playbook





https://github.com/quantum-mob/Q3

Q3 Application



A Quantum Computation Workbook

https://github.com/quantum-mob/QuantumPlaybook

감사합니다!