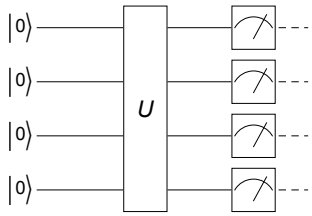
Chapter 2

Quantum Computation: Overview

- April 20, 2021 (v1.13)

In the simplest physical terms, quantum computation is an implementation of an arbitary unitary operation on a finite collection of *quantum bits* or *qubits* for short—a two-level quantum system. It is typically depicted in a *quantum circuit diagram* as following:



(2.1)

Each qubit is associated with a line which indicates the time evolution of the state specified on the left. The time flows from the left to the right. The *quantum logic gate operations* (or *gates* for short) on single or multiple qubits are denoted by a rectangular box often with labels indicating the types of the gates. Measurements are denoted by square boxes with needles. After a measurement, the time-evolution is represented by dashed lines to remind that the information is classical—no superposition.

The input state is prepared in one of the states in the logical basis, typically $|0\rangle \otimes \cdots \otimes |0\rangle$. After the unitary operation, the resulting state is measured in the logical basis, and the readouts are supposed to be the result of computation.

In order for a quantum computer to be programmable, it is required to implement a given unitary operator $\hat{U}$ in a combination of other more elementary unitary operators

$$\hat{U} = \hat{U}_1 \hat{U}_2 \cdots \hat{U}_L \,, \tag{2.2}$$

where each $\hat{U}_j$ is chosen from a small fixed set of elementary gate operations. The latter operations are called the *elementary quantum logic gates* for the quantum computer. In this chapter, we will examine widely used choices for elementary gates, and illustrate how a set of elementary gates achieve the arbitrary unitary operation, the so-called *universal quantum computation*.

Throughout the chapter, we denote by $\mathcal{S}$ the Hilbert space associated with a single qubit. The Hilbert space of an n-qubit system is given by $\mathcal{S}^{\otimes n}$, a tensor-product space of multiple $\mathcal{S}$. Each element $|x\rangle$ in the logical basis of $\mathcal{S}^{\otimes n}$ will be labeled by an integer $x = 0, 1, \cdots, 2^n - 1$, which should be understood to enumerate the tensor product form

$$|x\rangle \equiv |x_1 x_2 \cdots x_n\rangle := |x_1\rangle \otimes |x_2\rangle \otimes \cdots \otimes |x_n\rangle \tag{2.3}$$

in terms of the binary digits x_j $(j = 1, 2, \cdots, n)$ of x, that is, $x \equiv (x_1 x_2 \ldots x_n)_2$.

2.1 Single-Qubit Gates

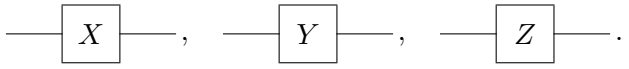
Unitary operators on the two-dimensional vector space $\mathcal{S}$ associated with a singel qubit form the *unitary group* U(2). In the standard basis, they are represented by a 2×2 unitary matrices. We first take a look at some special examples and discuss the general properties of the single-qubit unitary operations.

2.1.1 Pauli Gates

The Pauli gate opertaions (or Pauli operators for short) are defined by the corresponding Pauli matrices

$$\hat{X} \doteq \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \hat{Y} \doteq \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad \hat{Z} \doteq \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}. \tag{2.4}$$

They form the most elementary single-qubit gate operations and are frequently used in many quantum algorithms. In this book, the Pauli gates will be sometimes denoted by $\hat{X}$, $\hat{Y}$, and $\hat{Z}$, and othertimes by $\hat{S}^x$, $\hat{S}^y$, and $\hat{S}^z$, depending on the context. In the quantum circuit model, they are typically depicted by the circuit elements



(2.5)

Pauli $\hat{X}$ maps the logical basis states as

$$\hat{X} : |0\rangle \mapsto |1\rangle\,, \; |1\rangle \mapsto |0\rangle\,, \tag{2.6}$$

and is similar to the classical logic gate NOT. It is also customary to write Pauli $\hat{X}$ in the bra-ket notation as

$$\hat{X} = |1\rangle\langle 0| + |0\rangle\langle 1|\,. \tag{2.7}$$

It is important to remember that like any other quantum gate operations, it can take a linear superposition as input and transform the two logical basis states "simultaneously",

$$\hat{X}(|0\rangle\, c_0 + |1\rangle\, c_1) = |1\rangle\, c_0 + |0\rangle\, c_1\,, \tag{2.8}$$

which is not possible with the classical counterpart NOT.

The Pauli X gate is represented by `S[...,1]`.

```
In[•]:= S[1]
Out[•]= S^x
```

It corresponds to the Pauli X matrix.

```
In[•]:= Matrix[S[1]] // MatrixForm
Out[•]//MatrixForm=
        ( 0  1 )
        ( 1  0 )
```

In the quantum circuit model, it is denoted by the following quantum circuit element.

```
In[•]:= QuissoCircuit[S[1]]
Out[•]=  ─┤ X ├─
```

Operating on the logical basis states, it flips the states and is similar to the classical logical gate NOT.

```
In[•]:= bs = Basis[S];
        out = S[1] ** bs;
        Thread[bs → out] // LogicalForm // TableForm
Out[•]//TableForm=
        |0_S⟩ → |1_S⟩
        |1_S⟩ → |0_S⟩
```

Operating on a superposition state, it flips the state "simultaneously".

```
In[•]:= in = Ket[] × c[0] + Ket[S → 1] × c[1];
        in // LogicalForm
        out = S[1] ** in;
        out // LogicalForm
Out[•]= c_0 |0_S⟩ + c_1 |1_S⟩
Out[•]= c_1 |0_S⟩ + c_0 |1_S⟩
```

Operating on the logical basis states, Pauli $\hat{Z}$ only changes the relative phase of $|1\rangle$,

$$\hat{Z} : |0\rangle \mapsto |0\rangle\,,\ |1\rangle \mapsto -|1\rangle\,, \tag{2.9}$$

and hence in the bra-ket notation, it reads as

$$\hat{Z} = |0\rangle\langle 0| - |1\rangle\langle 1|\,. \tag{2.10}$$

The phase change is meaningless on classical bits, but it makes a significant difference on a superposition as illustrated in the following example

$$\hat{Z}(|0\rangle\, c_0 + |1\rangle\, c_1) = |0\rangle\, c_0 - |1\rangle\, c_1\,. \tag{2.11}$$

The Pauli Z gate is represented by `S[...,3]`.

```
In[•]:= S[3]
Out[•]= S^z
```

It corresponds to the Pauli Z matrix.

```
In[•]:= Matrix[S[3]] // MatrixForm
Out[•]//MatrixForm=
( 1   0 )
( 0  -1 )
```

In the quantum circuit model, it is denoted by the following quantum circuit element.

```
In[•]:= QuissoCircuit[S[3]]
Out[•]=  ─┤ Z ├─
```

Operating on the logical basis states, it "flips the phase", that is, it changes the phase factor to `-1` of the logical basis state $|1\rangle$.

```
In[•]:= bs = Basis[S];
        out = S[3] ** bs;
        Thread[bs → out] // LogicalForm // TableForm
Out[•]//TableForm=
|0_S⟩ → |0_S⟩
|1_S⟩ → - |1_S⟩
```

Here is an example how the Pauli Z gate acts on a superposition state.

```
In[•]:= in = Ket[] × c[0] + Ket[S → 1] × c[1];
        in // LogicalForm
        out = S[3] ** in;
        out // LogicalForm
Out[•]= c_0 |0_S⟩ + c_1 |1_S⟩
Out[•]= c_0 |0_S⟩ - c_1 |1_S⟩
```

Pauli $\hat{Y}$ combines the bit-flip feature of $\hat{X}$ and the phase-flip feature of $\hat{Z}$ to get

$$|0\rangle \mapsto i\,|1\rangle\ , \quad |1\rangle \mapsto -i\,|0\rangle\ . \tag{2.12}$$

This can also be seen in the operator identity, $\hat{Y} = i\hat{X}\hat{Z}$. In the bra-ket notation, it reads as

$$\hat{Y} = i\,|1\rangle\langle 0| - i\,|0\rangle\langle 1|\ . \tag{2.13}$$

The Pauli Y gate is represented by `S[...,2]`.

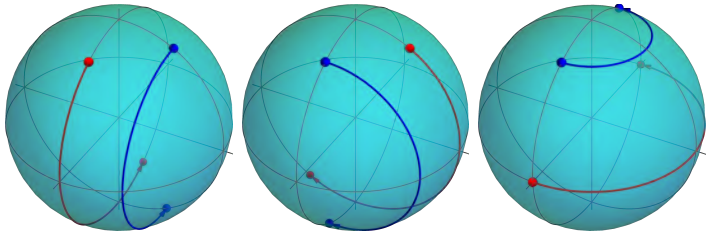


Figure 2.1: Illustration of the actions of Pauli gates as rotaions by angle π. From the left the actions of Pauli $\hat{X}$, $\hat{Y}$, $\hat{Z}$.

```
In[•]:= S[2]
Out[•]= S^y
```

It corresponds to the Pauli Y matrix.

```
In[•]:= Matrix[S[2]] // MatrixForm
Out[•]//MatrixForm=
( 0   -i )
( i    0 )
```

In the quantum circuit model, it is denoted by the following quantum circuit element.

```
In[•]:= QuissoCircuit[S[2]]
Out[•]=  —[ Y ]—
```

Pauli Y is a combination of the bit-flip (Pauli X) and phase-flip (Pauli Z) operation.

```
In[•]:= qc = QuissoCircuit[S[1], S[3]]
        op = QuissoExpression[qc]
Out[•]=  —[ X ]—[ Z ]—

Out[•]= i S^y
```

This shows more explicitly how Pauli Y "flips" both the bit and phase of the logical basis states.

```
In[•]:= bs = Basis[S];
        out = S[2] ** bs;
        Thread[bs → out] // LogicalForm // TableForm
Out[•]//TableForm=
|0_S⟩ → i |1_S⟩
|1_S⟩ → -i |0_S⟩
```

Here is an example how the Pauli Y gate acts on a superposition state.

```
In[•]:= in = Ket[] × c[0] + Ket[S → 1] × c[1];
        in // LogicalForm
        out = S[2] ** in;
        out // LogicalForm
Out[•]= c_0 |0_S⟩ + c_1 |1_S⟩

Out[•]= -i c_1 |0_S⟩ + i c_0 |1_S⟩
```

The Pauli gates can also be regarded as rotations by π around the x-, y-, and z-axis, respectively, in the Bloch sphere as illustrated in Fig. 2.1 and demonstrated in the following:

The Pauli gates also correspond, up to a global phase factor ($-i$), to rotations by the corresponding axes by angle π. Here `QuissoRotation[ϕ,S[...,μ]]` gives the rotation operator around the μ-axis by angle ϕ.

```
In[•]:= QuissoRotation[Pi, S[1]]
Out[•]= - i S^x

In[•]:= QuissoRotation[Pi, S[2]]
Out[•]= - i S^y

In[•]:= QuissoRotation[Pi, S[3]]
Out[•]= - i S^z
```

Here we are mainly focusing on their roles as unitary operators. However, the Pauli operators play another important role as an orthogonal *basis vectors* of the vector space of all linear operators on a two-dimensional vector space—see Section 2.1.3 and Appendix B.1.

2.1.2 Hadamard Gate

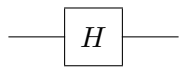
The Hadamard gate is one of the most frequently used elementary gates in many quantum algorithms. The Hadamard gate $\hat{H}$ is defined by the mapping:

$$|0\rangle \mapsto \frac{|0\rangle + |1\rangle}{\sqrt{2}}, \quad |1\rangle \mapsto \frac{|0\rangle - |1\rangle}{\sqrt{2}}. \tag{2.14}$$

That is, it constructs linear superpoistions of the two logical basis states. It is this feature that makes the Hadamard gate so useful, and is exploited in a wide range of quantum algorithms. In the logical basis, it is represented by the 2×2 Hadamard matrix

$$\hat{H} \doteq \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{2.15}$$

In the quantum circuit model, the Hadamard gate is depicted by an element with label "H"



(2.16)

Note that the output states in (2.14) corresponds to the eigenstates of the Pauli X operator. One can thus regard the Hadamard gate as a basis transformation from the logical basis to the eigenbasis of the Pauli X gate.

The Hadamard gate is represented by S[..., 6] .

```
In[•]:= S[1, 6]
Out[•]= S_1^H
```

Let us consider all the logical basis states.

```
In[•]:= bs = Basis[S[1]];
        bs // LogicalForm
Out[•]= {|0_S1⟩, |1_S1⟩}
```

Operating the Hadamard gate on them gives the two superposition states.

```
In[•]:= out = S[1, 6] ** bs;
        out // LogicalForm
Out[•]= {|0_S1⟩/√2 + |1_S1⟩/√2, |0_S1⟩/√2 - |1_S1⟩/√2}
```

In the quantum circuit model, it is denoted by the following circuit element.

```
In[•]:= QuissoCircuit[S[1, 6]]
Out[•]=  —[H]—
```

It is also ingsightful to note that it can be regarded (up to a global phase factor) as a rotation by angle π around the axis $(1, 0, 1)$ on the Bloch sphere. This can be seen from the following:

$$\hat{H} = \frac{1}{\sqrt{2}}\left(\hat{X} + \hat{Z}\right) = i \exp\left[-i\frac{\pi}{2}(\hat{X} + \hat{Z})\right] \tag{2.17}$$

This is illustrated in Fig. 2.2.

Geometrically, the Hadamard gate can be regarded as a rotation around the axis `(1,0,1)` in the Pauli space by angle π.

```
In[•]:= op = I QuissoRotation[π, S, {1, 0, 1}] // Garner
        mat = Matrix[op];
        mat // MatrixForm
Out[•]= S^x/√2 + S^z/√2
Out[•]//MatrixForm=
        ⎛ 1/√2   1/√2 ⎞
        ⎝ 1/√2  -1/√2 ⎠
```

An obvious but very useful feature is that it makes a linear superposition of *all* states in the logical basis: Consider a system of n qubits. When applied to each qubit in $|0\rangle$, it generates a linear superposition of all states in the logical basis

$$\hat{H}^{\otimes n}|0\rangle^{\otimes n} = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \otimes \cdots \otimes \frac{|0\rangle + |1\rangle}{\sqrt{2}} = \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle \ , \tag{2.18}$$

where

$$|x\rangle := |x_1\rangle \otimes |x_2\rangle \otimes \cdots \otimes |x_n\rangle \tag{2.19}$$

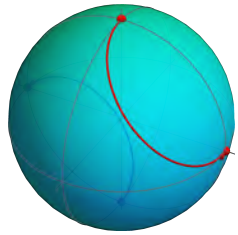


Figure 2.2: Illustration of the action of the Hadamard gate on the Bloch sphere. The Hadamard gate correponds to a rotation around the axis $(1, 0, 1)$ in the xz-plane by angle π.

for an integer x represented by $x = (x_1 x_2 \dots x_n)_2$ in the binary digits. More generaly, for an arbitrary state $|y\rangle$ in the logical basis,

$$\hat{H}^{\otimes n} |y\rangle = \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle \, (-1)^{x \cdot y} \,, \tag{2.20}$$

where we have used a short-hand notation

$$x \cdot y := x_1 y_1 + \cdots + x_n y_n \pmod 2. \tag{2.21}$$

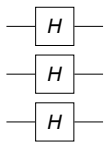
Suppose the Hadamard gates are applied to three qubits.

```
In[•]:= op = HoldForm@Multiply[S[1, 6], S[2, 6], S[3, 6]]
Out[•]= S_1^H S_2^H S_3^H
```

This shows the overall operation in the quantum circuit model.

```
In[•]:= qc = QuissoCircuit[S[{1, 2, 3}, 6], Null]
```

Out[•]=



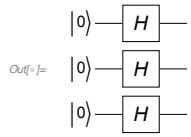
Operating the Hadamard gate on each qubit produces a superposition state consisting all logical basis states.

```
In[•]:= out = ReleaseHold[op] ** Ket[];
        out // LogicalForm
```

$$\text{Out}[\bullet]= \frac{|0_{s_1} 0_{s_2} 0_{s_3}\rangle}{2\sqrt{2}} + \frac{|0_{s_1} 0_{s_2} 1_{s_3}\rangle}{2\sqrt{2}} + \frac{|0_{s_1} 1_{s_2} 0_{s_3}\rangle}{2\sqrt{2}} + \frac{|0_{s_1} 1_{s_2} 1_{s_3}\rangle}{2\sqrt{2}} + \frac{|1_{s_1} 0_{s_2} 0_{s_3}\rangle}{2\sqrt{2}} + \frac{|1_{s_1} 0_{s_2} 1_{s_3}\rangle}{2\sqrt{2}} + \frac{|1_{s_1} 1_{s_2} 0_{s_3}\rangle}{2\sqrt{2}} + \frac{|1_{s_1} 1_{s_2} 1_{s_3}\rangle}{2\sqrt{2}}$$

This show the same result in the quantum circuit model.

```
In[•]:= qc = QuissoCircuit[LogicalForm[Ket[], S@{1, 2, 3}], S[{1, 2, 3}, 6]]
        QuissoExpression[qc] // LogicalForm
```



$$\text{Out}[\bullet]= \frac{\left|0_{S_1}0_{S_2}0_{S_3}\right\rangle}{2\sqrt{2}} + \frac{\left|0_{S_1}0_{S_2}1_{S_3}\right\rangle}{2\sqrt{2}} + \frac{\left|0_{S_1}1_{S_2}0_{S_3}\right\rangle}{2\sqrt{2}} + \frac{\left|0_{S_1}1_{S_2}1_{S_3}\right\rangle}{2\sqrt{2}} + \frac{\left|1_{S_1}0_{S_2}0_{S_3}\right\rangle}{2\sqrt{2}} + \frac{\left|1_{S_1}0_{S_2}1_{S_3}\right\rangle}{2\sqrt{2}} + \frac{\left|1_{S_1}1_{S_2}0_{S_3}\right\rangle}{2\sqrt{2}} + \frac{\left|1_{S_1}1_{S_2}1_{S_3}\right\rangle}{2\sqrt{2}}$$

Let us compare it with an explicit construction. To do it first prepare the indices of the logical basis states in binary digits.

```
In[•]:= nn = Range[0, 2^3 - 1];
        bit = IntegerDigits[nn, 2, 3]
Out[•]= {{0, 0, 0}, {0, 0, 1}, {0, 1, 0},
         {0, 1, 1}, {1, 0, 0}, {1, 0, 1}, {1, 1, 0}, {1, 1, 1}}
```

This gives an explicit construction (unnormalized) of the superposition of all local basis states.

```
In[•]:= vec = Total[Ket[S@{1, 2, 3} → #] & /@ bit];
        vec // LogicalForm
```

$$\text{Out}[\bullet]= \left|0_{S_1}0_{S_2}0_{S_3}\right\rangle + \left|0_{S_1}0_{S_2}1_{S_3}\right\rangle + \left|0_{S_1}1_{S_2}0_{S_3}\right\rangle + \left|0_{S_1}1_{S_2}1_{S_3}\right\rangle + \left|1_{S_1}0_{S_2}0_{S_3}\right\rangle + \left|1_{S_1}0_{S_2}1_{S_3}\right\rangle + \left|1_{S_1}1_{S_2}0_{S_3}\right\rangle + \left|1_{S_1}1_{S_2}1_{S_3}\right\rangle$$

On other elements of the logical basis, the sign of each term is determined by the bitwise dot product of the its bit-string with that of the input state.

```
In[•]:= in = Ket[S[{1, 2, 3}] → {1, 0, 1}];
        in = LogicalForm[in, S@{1, 2, 3}]
```

$$\text{Out}[\bullet]= \left|1_{S_1}0_{S_2}1_{S_3}\right\rangle$$

```
In[•]:= out = ReleaseHold[op] ** in;
        out // LogicalForm
```

$$\text{Out}[\bullet]= \frac{\left|0_{S_1}0_{S_2}0_{S_3}\right\rangle}{2\sqrt{2}} - \frac{\left|0_{S_1}0_{S_2}1_{S_3}\right\rangle}{2\sqrt{2}} + \frac{\left|0_{S_1}1_{S_2}0_{S_3}\right\rangle}{2\sqrt{2}} - \frac{\left|0_{S_1}1_{S_2}1_{S_3}\right\rangle}{2\sqrt{2}} - \frac{\left|1_{S_1}0_{S_2}0_{S_3}\right\rangle}{2\sqrt{2}} + \frac{\left|1_{S_1}0_{S_2}1_{S_3}\right\rangle}{2\sqrt{2}} - \frac{\left|1_{S_1}1_{S_2}0_{S_3}\right\rangle}{2\sqrt{2}} + \frac{\left|1_{S_1}1_{S_2}1_{S_3}\right\rangle}{2\sqrt{2}}$$

2.1.3 Rotations

Any unitary operator $\hat{U}$ can always be written in the form $\hat{U} = \exp(-i\hat{H})$ with a Hermitian operator $\hat{H}$. On a two-dimensional vector space $\mathcal{S}$ associated with a qubit, any Hermitian operator $\hat{H}$ can be expanded in terms of the Pauli operators $\hat{S}^\mu$ as

$$\hat{H} = \phi_0 + \hat{S}^x B_x + \hat{S}^y B_y + \hat{S}^z B_z \tag{2.22}$$

where ϕ_0, B_x, B_y, B_z are real parameters. Regarding $\boldsymbol{B} := (B_x, B_y, B_z)$ as a three-dimensional vector, we consider the unit vector $\boldsymbol{n}$ pointing to the same direction

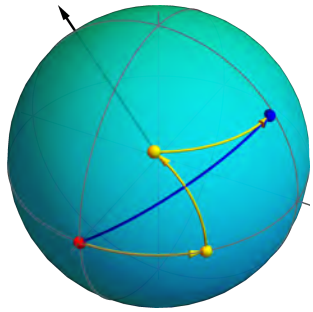


Figure 2.3: Visualization of transformations of states under the single-qubit operations. Up to a global phase factor, a single-qubit unitary operation is a rotation on the Bloch sphere. The rotation around the axis indicated by the black arrow is depicted by the blue arrow. The same rotation can be achieved by combining three rotations around y- or z-axis depicted by the yellow arrows.

as $\boldsymbol{B}$ and another real parameter $\phi := 2|\boldsymbol{B}|$, where the factor 2 is just for later convenience. In terms of these new parameterization, $\hat{H}$ reads as

$$\hat{H} = \phi_0 + \hat{\boldsymbol{S}} \cdot \boldsymbol{n}\, \phi/2\,, \tag{2.23}$$

where $\hat{\boldsymbol{S}} := (\hat{S}^x, \hat{S}^y, \hat{S}^z)$. In short, any unitary operator on $\mathcal{S}$ has the form $\hat{U} = e^{-i\phi_0}\hat{U}_{\boldsymbol{n}}(\phi)$ with

$$\hat{U}_{\boldsymbol{n}}(\phi) := \exp(-i\hat{\boldsymbol{S}} \cdot \boldsymbol{n}\, \phi/2). \tag{2.24}$$

Here $e^{-i\phi_0}$ changes the global phase factor and is physically irrelevant. More important and interesting is the part $\hat{U}_{\boldsymbol{n}}(\phi)$, which as we will see below, describes a "rotation" around the axis $\boldsymbol{n}$ by the angle ϕ. The rotations here are on the Bloch sphere—see Fig. 2.3 for an illustration—corresponding to the two-dimensional vector space $\mathcal{S}$, not in the real three-dimensional world. We will further denote the rotations around the μ-axis—$\boldsymbol{n}$ parallel to the μ-axis—of the Bloch sphere by $\hat{U}_\mu(\phi)$.

To see that the unitary operation $\hat{U}_{\boldsymbol{n}}(\phi)$ in (2.24) corresponds to a rotation, recall that the Pauli operators $\hat{S}^\mu$ are the spin angular momentum operators of spin 1/2. That is, they are the generators of rotations and satisfy the commutation relations

$$[\hat{S}^\mu, \hat{S}^\nu] = 2i \sum_\lambda \hat{S}^\lambda \epsilon_{\lambda\mu\nu}. \tag{2.25}$$

The connection of the unitary operator $\hat{U}_\lambda(\phi)$ to rotation is seen more explicitly in the equivalent relation

$$\hat{U}_\lambda(\phi)\hat{S}^\nu\hat{U}_\lambda^\dagger(\phi) = \sum_\mu \hat{S}^\mu [R_\lambda(\phi)]_{\mu\nu}\,, \tag{2.26}$$

where $R_\lambda(\phi)$ is the 3×3 orthogonal matrix describing the rotation of three-dimensional coordinates around the λ-axis by angle ϕ.

The Pauli operators are generators of the rotational transformations in a two-dimensional complex vector space, and hence satisfy the fundamental commutation relations of angular momentum operators (up to a normalization factor).

```
In[•]:= op = S[All]
Out[•]= {S^x, S^y, S^z}

In[•]:= in = Outer[HoldForm@*Commutator, op, op];
        out = ReleaseHold[in];
        Thread[Flatten[in] → Flatten[out]] // TableForm
Out[•]//TableForm=
        Commutator[S^x, S^x] → 0
        Commutator[S^x, S^y] → 2 i S^z
        Commutator[S^x, S^z] → -2 i S^y
        Commutator[S^y, S^x] → -2 i S^z
        Commutator[S^y, S^y] → 0
        Commutator[S^y, S^z] → 2 i S^x
        Commutator[S^z, S^x] → 2 i S^y
        Commutator[S^z, S^y] → -2 i S^x
        Commutator[S^z, S^z] → 0
```

In $\mathbb{R}^3$, any 3×3 rotation matrix can be decomposed into three factors

$$R = R_z(\alpha) R_y(\beta) R_z(\gamma)\,, \tag{2.27}$$

where α, β, γ are the so-called *Euler angles* and such a combination of rotations is called the *Euler rotation*. In the same manner, any unitary operator on $\mathcal{S}$ can also be written as

$$\hat{U} = e^{-i\phi_0} \hat{U}_z(\alpha) \hat{U}_y(\beta) \hat{U}_z(\gamma)\,, \tag{2.28}$$

that is, a combination of elementary "rotations" around the y- and z-axis and an additional overall phase shift. The unitary operator $\hat{U}(\alpha, \beta, \gamma) := \hat{U}_z(\alpha) \hat{U}_y(\beta) \hat{U}_z(\gamma)$ is called the Euler rotation in the two-dimensional vector space $\mathcal{S}$. Figure 2.3 illustrates an Euler rotation $\hat{U}(\pi/3, -\pi/3, \pi/4)$. It transforms $|v\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$ (red dot in Fig. 2.3) is transformed to $|w\rangle = \hat{U}\,|v\rangle$ (blue dot in Fig. 2.3). The transformation is displayed by the blue arrow. The yellow dots and arrows depicts the transformations under $\hat{U}_z(\alpha)$, $\hat{U}_y(\beta)$, and $\hat{U}_z(\gamma)$, which combine to reproduce $\hat{U}$.

Consider a unitary operator represented by the following matrix.

```
In[•]:= mat = {
            {3 - I Sqrt[3], I - Sqrt[3]},
            {I + Sqrt[3], 3 + I Sqrt[3]}
          } / 4;
        mat // MatrixForm
Out[•]//MatrixForm=
        ⎛ 1/4 (3 - i √3)   1/4 (i - √3) ⎞
        ⎝ 1/4 (i + √3)     1/4 (3 + i √3) ⎠
```

This gives its expression in terms of the Pauli operators.

```
In[•]:= op = QuissoExpression[mat, S]
Out[•]= 3/4 + i S^x/4 - 1/4 i √3 S^y - 1/4 i √3 S^z

In[•]:= Dagger[op] ** op
Out[•]= 1
```

This gives the Euler angles of the unitary operator.

```
In[•]:= angs = TheEulerAngles[mat]
Out[•]= {π/3, π/3, 0}
```

Indeed, the Euler angles reproduces the original unitary operator.

```
In[•]:= new = QuissoEulerRotation[angs, S]
        op - new
Out[•]= 3/4 + i S^x/4 - 1/4 i √3 S^y - 1/4 i √3 S^z
Out[•]= 0
```

2.2 Two-Qubit Gates

Let us next consider quantum logic gate operations acting on two qubits. Such operations are represented by 4×4 unitary matrices. We will see that any two-qubit gate operations can be decomposed into controlled-U gates. A controlled-U gate acts a unintary operator on one qubit depending on the logical state of the other qubit. A controlled-U gate on two qubits can be further decomposed into factors including only CNOT gate and single-qubit rotation gates. In this sense, the CNOT gate alone is sufficient for any two-qubit gate.

The controlled-U and CNOT gate have various interesting properties that make them useful in the implementation of quantum algorithms. In this section, we will first examine the basic properties of the CNOT gate, in particular, how it is used to generate an entanglement between two qubits. We then discuss the properties of the controlled-U gate and how to implement a controlled-U gate in terms of the CNOT gate and the single-qubit rotations. Finally, we discuss how an arbitrary two-qubit unitary operation can be decomposed into controlled-U gates.

2.2.1 CNOT, CZ, and SWAP

The CNOT or controlled-NOT gate is a quantum logic gate on two qubits that maps the logical basis states as

$$|c\rangle \otimes |t\rangle \mapsto |c\rangle \otimes |c \oplus t\rangle \ ; \quad c, t \in \{0, 1\} \ , \tag{2.29}$$

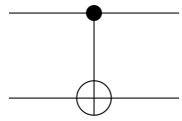
where the first qubit is typically called the *control qubit* (c) and the second qubit the *target qubit* (t). It has the following matrix representation in the logical basis

$$\mathrm{CNOT} \doteq \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 0 & 1 \\ & & 1 & 0 \end{bmatrix}, \tag{2.30}$$

and expressed in terms of the Pauli operators on the control and target qubit as

$$\mathrm{CNOT} = \frac{1}{2}\left(1 + \hat{S}_c^z + \hat{S}_t^z - \hat{S}_c^z \hat{S}_t^x\right). \tag{2.31}$$

In a quantum circuit model, it is represented as the following circuit element:



, (2.32)

where the smaller filled circle indicates the dependence on the state of the control qubit and the circled-plus sign denotes the conditional NOT action on the target qubit.

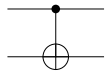
CNOT[*control*, *target*] denotes the CNOT gate in the quantum circuit model.

```
In[•]:= op = CNOT[S[1], S[2]]
Out[•]= CNOT[{S1}, {S2}]
```

This displays the quantum circuit model of the CNOT gate.

```
In[•]:= qc = QuissoCircuit[op]
```

Out[•]=



This is the explicit expression of the CNOT gate in terms of the Pauli operators.

```
In[•]:= op = QuissoExpression[qc]
```

$$\text{Out}[\bullet]= \frac{1}{2} - \frac{1}{2} S_1^z S_2^x + \frac{S_1^z}{2} + \frac{S_2^x}{2}$$

This is the matrix representation of the CNOT gate in the logical basis.

```
In[•]:= Matrix[op] // MatrixForm
```

Out[•]//MatrixForm=

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

This shows how the CNOT gate operates on the logical basis states.

```
In[•]:= in = Basis[S@{1, 2}];
        out = op ** in;
        Thread[in → out] // LogicalForm // TableForm
Out[•]//TableForm=
        |0_{s_1} 0_{s_2}⟩ → |0_{s_1} 0_{s_2}⟩
        |0_{s_1} 1_{s_2}⟩ → |0_{s_1} 1_{s_2}⟩
        |1_{s_1} 0_{s_2}⟩ → |1_{s_1} 1_{s_2}⟩
        |1_{s_1} 1_{s_2}⟩ → |1_{s_1} 0_{s_2}⟩
```

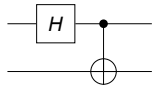
A simple yet important feature of CNOT is to copy the logical state of the control qubit to the target bit provided that the target bit is initially set to $|0\rangle$; or the reversed state when the target qubit is in $|1\rangle$. A vital implication is that CNOT generates an entangled state when the control qubit is in a superposition:

$$(|0\rangle\, c_0 + |1\rangle\, c_1) \otimes |0\rangle \mapsto |0\rangle \otimes |0\rangle\, c_0 + |1\rangle \otimes |1\rangle\, c_1. \tag{2.33}$$

Applying a single qubit rotation such the Hadamard gate on the control qubit prior to CNOT, one can thus generate entangled states from logical states. In this sense, such a circuit is called *quantum entangler circuit*.

As an example of the application of the CNOT gate, this shows an entangler quantum circuit.

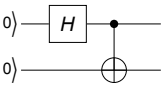
```
In[•]:= entangler = QuissoCircuit[S[1, 6], CNOT[S[1], S[2]]]
Out[•]=
```



For example, when the input state is $|0\rangle \otimes |0\rangle$, the outcome of the circuit is given by one of the so-called Bell states.

This demonstrates the generation of an entangled state from a product state.

```
In[•]:= new = QuissoCircuit[LogicalForm[Ket[S@{1, 2} → {0, 0}], S@{1, 2}], entangler]
        vec = QuissoExpression[new];
        vec // LogicalForm
Out[•]=
```



$$\text{Out}[\bullet]= \frac{|0_{s_1} 0_{s_2}\rangle}{\sqrt{2}} + \frac{|1_{s_1} 1_{s_2}\rangle}{\sqrt{2}}$$

In general, the product states in the logical basis are transformed to the Bell states as you can see in the demonstration below.

This lists the mapping between the standard tensor-product basis states and the Bell states.

```
In[•]:= bs = Basis@S@{1, 2};
        op = QuissoExpression[entangler];
        out = op ** bs;
        table = Thread[bs → out];
        table // LogicalForm // TableForm
```

Out[•]//TableForm=

$$\left|0_{S_1}0_{S_2}\right\rangle \to \frac{\left|0_{S_1}0_{S_2}\right\rangle}{\sqrt{2}} + \frac{\left|1_{S_1}1_{S_2}\right\rangle}{\sqrt{2}}$$
$$\left|0_{S_1}1_{S_2}\right\rangle \to \frac{\left|0_{S_1}1_{S_2}\right\rangle}{\sqrt{2}} + \frac{\left|1_{S_1}0_{S_2}\right\rangle}{\sqrt{2}}$$
$$\left|1_{S_1}0_{S_2}\right\rangle \to \frac{\left|0_{S_1}0_{S_2}\right\rangle}{\sqrt{2}} - \frac{\left|1_{S_1}1_{S_2}\right\rangle}{\sqrt{2}}$$
$$\left|1_{S_1}1_{S_2}\right\rangle \to \frac{\left|0_{S_1}1_{S_2}\right\rangle}{\sqrt{2}} - \frac{\left|1_{S_1}0_{S_2}\right\rangle}{\sqrt{2}}$$

Quantum entanglement is a valuable resource in quantum information processing and quantum communications. The most popular example is quantum teleportation to be discussed in Section 4.1. One can further generalize the above procedure to generate an maximally entangled states between systems larger than one qubit, as demonstrated in the following example:

Example 6 Consider two systems, A and B, *each* of which consisting of two qubits. Construct a quantum circuit model which generates a maximally entangled state

$$\sum_{j=0}^{3} |j\rangle_A \otimes |j\rangle_B = |00\rangle \otimes |00\rangle + |01\rangle \otimes |01\rangle + |10\rangle \otimes |10\rangle + |11\rangle \otimes |11\rangle \quad (2.34)$$

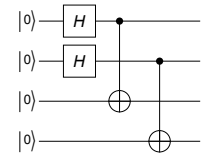
starting from the product state $|00\rangle \otimes |00\rangle$. Can you generalize it to any pair of n-qubit systems?

Hint: Recall the property of the Hadamard gate in (2.18).

Solution: To generate a maximally entangled state between two two-qubit registers.

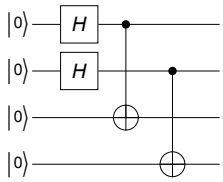
```
In[•]:= qc = QuissoCircuit[LogicalForm[Ket[], S@{1, 2, 3, 4}],
          S[{1, 2}, 6], CNOT[S[1], S[3]], CNOT[S[2], S[4]],
          PlotRangePadding → 0, ImagePadding → {{36, 36}, {5, 5}}]
```

Out[•]=




```
In[•]:= qc = QuissoCircuit[LogicalForm[Ket[], S@{1, 2, 3, 4}],
          S[{1, 2}, 6], CNOT[S[1], S[3]], CNOT[S[2], S[4]]]
```

Out[•]=



```
In[•]:= out = QuissoExpression[qc];
        out // LogicalForm
```

$$\text{Out[•]=}\quad \frac{1}{2}\left|0_{S_1}0_{S_2}0_{S_3}0_{S_4}\right\rangle + \frac{1}{2}\left|0_{S_1}1_{S_2}0_{S_3}1_{S_4}\right\rangle + \frac{1}{2}\left|1_{S_1}0_{S_2}1_{S_3}0_{S_4}\right\rangle + \frac{1}{2}\left|1_{S_1}1_{S_2}1_{S_3}1_{S_4}\right\rangle$$

This example illustrate that the entanglement depends on the partition of the system. Indeed, the above system is a product state for the partition (1,3) and (2,4) qubits.

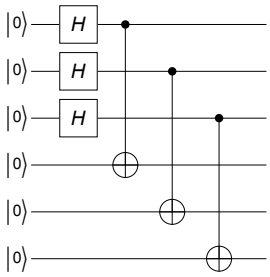
```
In[•]:= QuissoFactor[out]
```

$$\text{Out[•]=}\quad \frac{1}{2}\left(\left|0_{S_1}0_{S_3}\right\rangle + \left|1_{S_1}1_{S_3}\right\rangle\right) \otimes \left(\left|0_{S_2}0_{S_4}\right\rangle + \left|1_{S_2}1_{S_4}\right\rangle\right)$$

The above construction can be generalized for a pair of n-qubit systems.

```
In[•]:= n = 3;
        qc = QuissoCircuit[LogicalForm[Ket[], S@Range[2 n]],
          S[Range[n], 6], Sequence @@ Table[CNOT[S[j], S[n + j]], {j, 1, n}]]
```

Out[•]=




```
In[•]:= out = QuissoExpression[qc];
        out // LogicalForm
```

$$\text{Out[•]=}\quad \frac{\left|0_{S_1}0_{S_2}0_{S_3}0_{S_4}0_{S_5}0_{S_6}\right\rangle}{2\sqrt{2}} + \frac{\left|0_{S_1}0_{S_2}1_{S_3}0_{S_4}0_{S_5}1_{S_6}\right\rangle}{2\sqrt{2}} + \frac{\left|0_{S_1}1_{S_2}0_{S_3}0_{S_4}1_{S_5}0_{S_6}\right\rangle}{2\sqrt{2}} + \frac{\left|0_{S_1}1_{S_2}1_{S_3}0_{S_4}1_{S_5}1_{S_6}\right\rangle}{2\sqrt{2}} +$$
$$\frac{\left|1_{S_1}0_{S_2}0_{S_3}1_{S_4}0_{S_5}0_{S_6}\right\rangle}{2\sqrt{2}} + \frac{\left|1_{S_1}0_{S_2}1_{S_3}1_{S_4}0_{S_5}1_{S_6}\right\rangle}{2\sqrt{2}} + \frac{\left|1_{S_1}1_{S_2}0_{S_3}1_{S_4}1_{S_5}0_{S_6}\right\rangle}{2\sqrt{2}} + \frac{\left|1_{S_1}1_{S_2}1_{S_3}1_{S_4}1_{S_5}1_{S_6}\right\rangle}{2\sqrt{2}}$$

An interesting variant of CNOT gates is the so-called CZ or controlled-Z gate: It is a quantum logic gate on two qubits that maps the logical basis states as

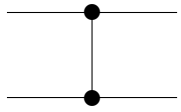
$$\mathrm{CZ} : |c\rangle \otimes |t\rangle \mapsto |c\rangle \otimes |t\rangle\,(-1)^{ct}\,. \tag{2.35}$$

The matrix representation of the CZ gate in the logical basis is given by

$$\mathrm{CZ} \doteq \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & -1 \end{bmatrix}. \tag{2.36}$$

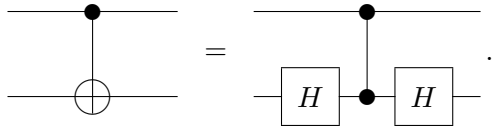
As it is symmetric for the two qubits, the distinction of the control and taget qubit is meaningless. Accordingly, in the quantum circuit model, the gate is depicted by

the following quantum circuit element



. (2.37)

The filled circles on both qubit lines–rather than a square box on either qubit—indicates that the bit values of the both qubits remain unchanged. Noting the identity $\hat{H}\hat{Z}\hat{H} = \hat{X}$, one can regard that the CZ gate is equal to the CNOT gate up to the Hadamard gate on the target qubit. The relation between the CNOT and CZ gate is expressed in the following quantum circuit model



. (2.38)

Depending on the Hamiltonian of particular physical system, the direct realization of the CNOT gate may be significantly difficult while the CZ gate is relatively easier to realize. In such a case, the identity (2.38) offers a straightforward workaround for the physical implementation of the CNOT gate.

The CZ (or controlled-Z) gate is a variant of the CNOT gate.

```
In[•]:= op = CZ[S[1], S[2]]
Out[•]= CZ[S1, S2]
```

This shows how it transforms the logical basis states.

```
In[•]:= bs = Basis@S@{1, 2};
        out = op ** bs;
        Thread[bs → out] // LogicalForm // TableForm
Out[•]//TableForm=
        |0_S1 0_S2⟩ → |0_S1 0_S2⟩
        |0_S1 1_S2⟩ → |0_S1 1_S2⟩
        |1_S1 0_S2⟩ → |1_S1 0_S2⟩
        |1_S1 1_S2⟩ → - |1_S1 1_S2⟩
```

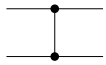
Here is the matrix representation of the CZ gate.

```
In[•]:= mat = Matrix[Elaborate@op];
        mat // MatrixForm
Out[•]//MatrixForm=
        ⎛1  0  0  0 ⎞
        ⎜0  1  0  0 ⎟
        ⎜0  0  1  0 ⎟
        ⎝0  0  0  -1⎠
```

This is the quantum circuit model of the CZ gate.

```
In[•]:= cz = QuissoCircuit[CZ[S[1], S[2]]]
```

Out[•]=



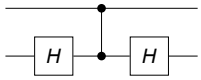
Note the following identity.

```
In[•]:= expr = HoldForm[S[1, 6] ** S[1, 3] ** S[1, 6] == S[1, 1]]
        ReleaseHold[expr] // Elaborate
Out[•]= S_1^H ** S_1^z ** S_1^H == S_1^x
Out[•]= True
```

It leads to the following relation between the CNOT gate and CZ gate.

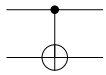
```
In[•]:= new = QuissoCircuit[S[2, 6], cz, S[2, 6]]
```

Out[•]=




```
In[•]:= cnot = QuissoCircuit[CNOT[S[1], S[2]]]
```

Out[•]=



```
In[•]:= Elaborate[new - cnot]
Out[•]= 0
```

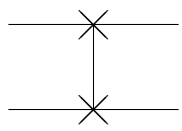
Another interesting two-qubit gate is the SWAP gate: The SWAP gate "swaps" the states of the two qubits, and maps the logical basis states as

$$\text{SWAP} : |x_1\rangle \otimes |x_2\rangle \mapsto |x_2\rangle \otimes |x_1\rangle \ . \tag{2.39}$$

As the states $|0\rangle \otimes |0\rangle$ and $|1\rangle \otimes |1\rangle$ are not altered by the operation, the matrix representation is given by

$$\text{SWAP} \doteq \begin{bmatrix} 1 & & & \\ & 0 & 1 & \\ & 1 & 0 & \\ & & & 1 \end{bmatrix} . \tag{2.40}$$

In the qunatum circuit model, it is depicted as



(2.41)

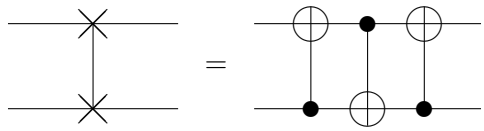
The SWAP gate can be implemented using the CNOT gate. To see this, first note that the simulaneous exchange of second and forth columns and rows of the matrix in (2.40) leads to

$$\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 0 & 1 \\ & & 1 & 0 \end{bmatrix} , \tag{2.42}$$

which is nothing but the matrix representation of the CNOT gate. On the other hand, the exchange of the second and forth columns and rows is described by the transformation matrix

$$\begin{bmatrix} 1 & & & \\ & 0 & & 1 \\ & & 1 & \\ & 1 & & 0 \end{bmatrix}, \tag{2.43}$$

which flips the bit values of the first qubit only when the second qubit is set to $|1\rangle$, and hence it corresponds to the CNOT gate with the second and first qubit as the control and target qubit, respectively. In short, the SWAP gate can be achieved by a combination of the CNOT gates as following



(2.44)

The SWAP gate exchanges the states of two qubits.

```
In[•]:= op = SWAP[S[1], S[2]]
Out[•]= SWAP[S1, S2]
```

```
In[•]:= bs = Basis@S@{1, 2};
        new = op ** bs;
        Thread[bs → new] // LogicalForm // TableForm
```

Out[•]//TableForm=

$$\begin{aligned} \left|0_{S_1} 0_{S_2}\right\rangle &\to \left|0_{S_1} 0_{S_2}\right\rangle \\ \left|0_{S_1} 1_{S_2}\right\rangle &\to \left|1_{S_1} 0_{S_2}\right\rangle \\ \left|1_{S_1} 0_{S_2}\right\rangle &\to \left|0_{S_1} 1_{S_2}\right\rangle \\ \left|1_{S_1} 1_{S_2}\right\rangle &\to \left|1_{S_1} 1_{S_2}\right\rangle \end{aligned}$$

This is the matrix representation of the SWAP gate in the logical basis.

```
In[•]:= Matrix[Elaborate@op] // MatrixForm
```

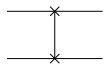
Out[•]//MatrixForm=

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

In the quantum circuit model, the SWAP gate is represented as following.

```
In[•]:= qc = QuissoCircuit[SWAP[S[1], S[2]]]
```

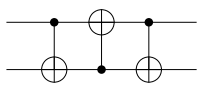
Out[•]=



The SWAP gate can be implemented by means of the CNOT gate.

```
In[•]:= new = QuissoCircuit[CNOT[S[1], S[2]], CNOT[S[2], S[1]], CNOT[S[1], S[2]]]
```

Out[•]=



```
In[•]:= Elaborate[qc - new]
Out[•]= 0
```

Interestingly, the SWAP gate itself is not universal, but the $\sqrt{\text{SWAP}}$ gate—the gate when squared equals to SWAP—is. That is, any quantum gate on a multi-qubit system can be implemented using $\sqrt{\text{SWAP}}$ and single-qubit rotations—see also Section 2.4 and Section 3.2.2. Indeed, one can combine the $\sqrt{\text{SWAP}}$ gate with single-qubit rotations to construct the CZ gate (Loss & DiVincenzo, 1998). As discussed above, the CZ gate just requires two more Hadamard gates to implement the CNOT gate and hence is universal.

Let us construct the CZ gate with the $\sqrt{\text{SWAP}}$ gate. This is the matrix representation of the the $\sqrt{\text{SWAP}}$ gate.

```
In[•]:= mat = MatrixPower[Matrix@Elaborate@SWAP[S[1], S[2]], 1 / 2];
        mat // MatrixForm
```

Out[•]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2}+\frac{i}{2} & \frac{1}{2}-\frac{i}{2} & 0 \\ 0 & \frac{1}{2}-\frac{i}{2} & \frac{1}{2}+\frac{i}{2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

This is an explicit operator expression of the $\sqrt{\text{SWAP}}$ gate in terms of the Pauli operators.

```
In[•]:= sqrtSWAP = {QuissoExpression[mat, S@{1, 2}], , "Label" → "√U"}
```

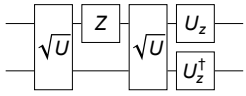
Out[•]=

$$\left\{\left(\frac{3}{4}+\frac{i}{4}\right)+\left(\frac{1}{4}-\frac{i}{4}\right)S_1^x S_2^x+\left(\frac{1}{4}-\frac{i}{4}\right)S_1^y S_2^y+\left(\frac{1}{4}-\frac{i}{4}\right)S_1^z S_2^z,\ \text{Null},\ \text{Label}\to\sqrt{U}\right\}$$

This is a quantum circuit model to construct the CZ gate from the $\sqrt{\text{SWAP}}$ gate and single-qubit gates. In this diagram, $\sqrt{U}$ denotes the $\sqrt{\text{SWAP}}$ gate and U_z the rotation around the z-axis by angle $\pi/2$.

```
In[•]:= qc = QuissoCircuit[sqrtSWAP, S[1, 3], sqrtSWAP,
          {Rotation[Pi / 2, S[1, 3]], Rotation[-Pi / 2, S[2, 3], "Label" → "U_z^†"]}]
```

Out[•]=



To check if it indeeds implement the CZ gate, take a look at the matrix representation of the quantum circuit model.

```
In[•]:= Matrix[qc] // MatrixForm
```

Out[•]//MatrixForm=

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

2.2.2 Controlled-U Gate

Consider two qubits, again called as the control and target qubit. Let $\hat{U}$ be a unitary operator on the target qubit. The controlled-U gate is a unitary operator on the two-qubit Hilbert space defined analogously to CNOT by

$$\mathrm{Ctrl}(\hat{U}) : |c\rangle \otimes |t\rangle \mapsto |c\rangle \otimes \hat{U}^c |t\rangle \ , \tag{2.45}$$

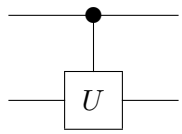
or equivalently, by

$$\mathrm{Ctrl}(\hat{U}) := |0\rangle\langle 0| \otimes \hat{I} + |1\rangle\langle 1| \otimes \hat{U} \, . \tag{2.46}$$

If the conrol qubit is in $|0\rangle$, then it does nothing. On the other hand, if the control qubit is in $|1\rangle$, the unitary operator $\hat{U}$ acts on the target qubit. In analogy with the CNOT gate, the matrix representation of the controlled-U gate is thus given by

$$\mathrm{Ctrl}(\hat{U}) \doteq \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & U_{11} & U_{12} \\ & & U_{21} & U_{22} \end{bmatrix} , \tag{2.47}$$

where U is the matrix representation of $\hat{U}$. In the quantum circuit model, a controlled-U gate is depicted as



(2.48)

The filled circle connected to the quantum circuit element on the target qubit indicates the conditional operation of the element.

Consider a single-qubit rotation acting on the qubit `S[2,None]`. It is a rotation around the y-axis by angle ϕ.

In[•]:=
```
U = Rotation[ϕ, S[2, 2], "Label" → "U"];
U // Elaborate // Matrix // MatrixForm
```

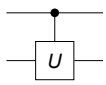
Out[•]//MatrixForm=

$$\begin{pmatrix} \mathrm{Cos}\left[\frac{\phi}{2}\right] & -\mathrm{Sin}\left[\frac{\phi}{2}\right] \\ \mathrm{Sin}\left[\frac{\phi}{2}\right] & \mathrm{Cos}\left[\frac{\phi}{2}\right] \end{pmatrix}$$

This shows the controlled-U gate.

In[•]:=
```
qc = QuissoCircuit[ControlledU[S[1], U]]
```

Out[•]=



This is the explicit expression of the controlled-U gate operation in terms of the Pauli operators.

```
In[•]:= op = Elaborate[qc]
```

$$\text{Out[•]= } \text{Cos}\left[\frac{\phi}{4}\right]^2 + S_1^z \,\text{Sin}\left[\frac{\phi}{4}\right]^2 + \frac{1}{2}\, i\, S_1^z\, S_2^y\, \text{Sin}\left[\frac{\phi}{2}\right] - \frac{1}{2}\, i\, S_2^y\, \text{Sin}\left[\frac{\phi}{2}\right]$$

The controlled-U gate maps the logical basis states as following.

```
In[•]:= bs = Basis[S@{1, 2}];
        bs // LogicalForm
        out = op ** bs;
        out // LogicalForm
```

$$\text{Out[•]= } \left\{\left|0_{S_1} 0_{S_2}\right\rangle,\ \left|0_{S_1} 1_{S_2}\right\rangle,\ \left|1_{S_1} 0_{S_2}\right\rangle,\ \left|1_{S_1} 1_{S_2}\right\rangle\right\}$$

$$\text{Out[•]= } \left\{\left|0_{S_1} 0_{S_2}\right\rangle,\ \left|0_{S_1} 1_{S_2}\right\rangle,\ \text{Cos}\left[\frac{\phi}{2}\right]\ \left|1_{S_1} 0_{S_2}\right\rangle + \left|1_{S_1} 1_{S_2}\right\rangle \text{Sin}\left[\frac{\phi}{2}\right],\right.$$
$$\left.\text{Cos}\left[\frac{\phi}{2}\right]\ \left|1_{S_1} 1_{S_2}\right\rangle - \left|1_{S_1} 0_{S_2}\right\rangle \text{Sin}\left[\frac{\phi}{2}\right]\right\}$$

To make the mapping clearer, this tabulates the above result.

```
In[•]:= new = QuissoFactor[#, S[1]] & /@ out;
        Thread[bs → new] // LogicalForm // TableForm
```

Out[•]//TableForm=

$$\left|0_{S_1} 0_{S_2}\right\rangle \to \left|0_{S_1}\right\rangle \otimes \left|0_{S_2}\right\rangle$$
$$\left|0_{S_1} 1_{S_2}\right\rangle \to \left|0_{S_1}\right\rangle \otimes \left|1_{S_2}\right\rangle$$
$$\left|1_{S_1} 0_{S_2}\right\rangle \to \left|1_{S_1}\right\rangle \otimes \left(\text{Cos}\left[\frac{\phi}{2}\right]\ \left|0_{S_2}\right\rangle + \left|1_{S_2}\right\rangle \text{Sin}\left[\frac{\phi}{2}\right]\right)$$
$$\left|1_{S_1} 1_{S_2}\right\rangle \to \left|1_{S_1}\right\rangle \otimes \left(\text{Cos}\left[\frac{\phi}{2}\right]\ \left|1_{S_2}\right\rangle - \left|0_{S_2}\right\rangle \text{Sin}\left[\frac{\phi}{2}\right]\right)$$

Let us take a look at the mapping more closely. When the first qubit is set to `Ket[0]`, it does nothing.

```
In[•]:= Let[Complex, c]
        vec = Ket[] × c[0] + Ket[S[2] → 1] × c[2];
        LogicalForm[vec, S@{1, 2}]
```

$$\text{Out[•]= } c_0 \left|0_{S_1} 0_{S_2}\right\rangle + c_2 \left|0_{S_1} 1_{S_2}\right\rangle$$

```
In[•]:= new = op ** vec;
        LogicalForm[new, S@{1, 2}]
```

$$\text{Out[•]= } c_0 \left|0_{S_1} 0_{S_2}\right\rangle + c_2 \left|0_{S_1} 1_{S_2}\right\rangle$$

When the control qubit -- the first qubit in this case -- is set to `Ket[1]`, it operates the unitary operator on the second qubit.

```
In[•]:= vec = Ket[S[1] → 1] ** (Ket[] × c[0] + Ket[S[2] → 1] × c[2]);
        LogicalForm[vec, S@{1, 2}]
```

$$\text{Out[•]= } c_0 \left|1_{S_1} 0_{S_2}\right\rangle + c_2 \left|1_{S_1} 1_{S_2}\right\rangle$$

```
In[•]:= new = op ** vec;
        LogicalForm[new, S@{1, 2}]
```

$$\text{Out[•]= } \left|1_{S_1} 1_{S_2}\right\rangle \left(c_2 \,\text{Cos}\left[\frac{\phi}{2}\right] + c_0 \,\text{Sin}\left[\frac{\phi}{2}\right]\right) + \left|1_{S_1} 0_{S_2}\right\rangle \left(c_0 \,\text{Cos}\left[\frac{\phi}{2}\right] - c_2 \,\text{Sin}\left[\frac{\phi}{2}\right]\right)$$

When the control qubit is in a superposition, the resulting state is an entangled state in general.

```
In[•]:= vec = Ket[] + Ket[S[1] → 1];
        LogicalForm[vec, S@{1, 2}]
```

$$\text{Out[•]= } \left|0_{S_1} 0_{S_2}\right\rangle + \left|1_{S_1} 0_{S_2}\right\rangle$$

```
In[•]:= new = op ** vec;
        LogicalForm[new, S@{1, 2}]
Out[•]= |0_S1 0_S2⟩ + Cos[ϕ/2] |1_S1 0_S2⟩ + |1_S1 1_S2⟩ Sin[ϕ/2]
```

An important aspect of the controlled-U operator is that it induces relative phase shifts on the *control* qubits when the target is prepared in an eigenstate of $\hat{U}$. At a first glace, it may sound counter intuitive as the definition in (2.45) seems to indicate that it changes the target qubit depending on the state of the control qubit and leaves the latter intact. This is another feature distinguishing quantum gates from the classical counterparts. Let us take a closer look to see how it works. Prepare the control register in a superposition $|\psi\rangle = |0\rangle + |1\rangle$ and the target register in a eigenstate $|u\rangle$ of $\hat{U}$ with eigenvalue $e^{i\phi}$. The controlled-U gate transforms the state to

$$\begin{aligned} |\psi\rangle \otimes |u\rangle \to |0\rangle \otimes |u\rangle + |1\rangle \otimes \hat{U}\,|u\rangle \\ = |0\rangle \otimes |u\rangle + |1\rangle \otimes |u\rangle\, e^{i\phi} = \left(|0\rangle + |1\rangle\, e^{i\phi}\right) \otimes |u\rangle\,. \end{aligned} \tag{2.49}$$

This feature is extended to multi-qubit controlled-U gates and plays a crucial role in many quantum algorithms. In particular, the quantum phase estimation algorithm (Section 4.4) is a direct consequence of this feature.

When the target qubit is set to an eigenstate of the unitary operator, it does not change but the control qubit acquires the phase factor given by the eigenvalue of the target state.

```
In[•]:= vec = (Ket[] + Ket[S[1] → 1]) ** (Ket[] - I Ket[S[2] → 1]);
        LogicalForm[QuissoFactor@vec, S@{1, 2}]
Out[•]= (|0_S1⟩ + |1_S1⟩) ⊗ (|0_S2⟩ - i |1_S2⟩)

In[•]:= new = op ** vec // TrigToExp;
        LogicalForm[QuissoFactor@new, S@{1, 2}]
Out[•]= (|0_S1⟩ + e^(iϕ/2) |1_S1⟩) ⊗ (|0_S2⟩ - i |1_S2⟩)
```

Example 7 Suppose that a qubit is known to be in one of the two eigenstates of the unitary operator

$$\hat{U} = \hat{\sigma}^0 \cos(\phi/2) - i\hat{\sigma}^x \sin(\phi/2) \tag{2.50}$$

with the angle ϕ *known*. Construct a quantum circuit model to figure out the unknown state using an additional qubit.

Hint: Use a controlled-U gate to acquire the one-bit information. This is a simplified version of the quantum phase estimation procedure (see Section 4.4), but it can be worked out without resorting to it.

Solution: The eigenstates of `U` are the same as those of `Pauli[1]`.

```
In[•]:= U = Rotation[ϕ, S[1, 1]] // Elaborate
```

$$\text{Out[•]=}\quad \text{Cos}\left[\frac{\phi}{2}\right] - \text{i}\, S_1^x \,\text{Sin}\left[\frac{\phi}{2}\right]$$

```
In[•]:= vec = S[1, 6] ** Ket[];
        vec // LogicalForm
```

$$\text{Out[•]=}\quad \frac{\left|0_{S_1}\right\rangle}{\sqrt{2}} + \frac{\left|1_{S_1}\right\rangle}{\sqrt{2}}$$

```
In[•]:= U ** vec // LogicalForm // TrigToExp // Simplify
```

$$\text{Out[•]=}\quad \frac{e^{-\frac{i\phi}{2}}\left(\left|0_{S_1}\right\rangle + \left|1_{S_1}\right\rangle\right)}{\sqrt{2}}$$

```
In[•]:= vec = S[1, 6] ** Ket[S[1] → 1];
        vec // LogicalForm
```

$$\text{Out[•]=}\quad \frac{\left|0_{S_1}\right\rangle}{\sqrt{2}} - \frac{\left|1_{S_1}\right\rangle}{\sqrt{2}}$$

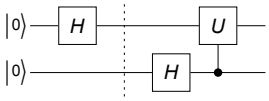
```
In[•]:= U ** vec // LogicalForm // TrigToExp // Simplify
```

$$\text{Out[•]=}\quad \frac{e^{\frac{i\phi}{2}}\left(\left|0_{S_1}\right\rangle - \left|1_{S_1}\right\rangle\right)}{\sqrt{2}}$$

The ancillary qubit takes a relative phase shift depending on in which eigenstate the native qubit is.

```
In[•]:= Let[Real, ϕ];
        qc = QuissoCircuit[LogicalForm[Ket[], S@{1, 2}], S[1, 6], "Separator",
          S[2, 6], ControlledU[S[2], Rotation[ϕ, S[1, 1]], "Label" → "U"]]
```

Out[•]=



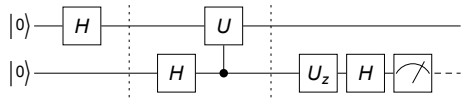

```
In[•]:= out = QuissoExpression[qc] // TrigToExp;
        LogicalForm[QuissoFactor@out, S@{1, 2}]
```

$$\text{Out[•]=}\quad \frac{1}{2}\, e^{-\frac{i\phi}{2}}\left(\left|0_{S_1}\right\rangle + \left|1_{S_1}\right\rangle\right) \otimes \left(e^{\frac{i\phi}{2}}\left|0_{S_2}\right\rangle + \left|1_{S_2}\right\rangle\right)$$

Make a basis change to detect it.

```
In[•]:= qc = QuissoCircuit[LogicalForm[Ket[], S@{1, 2}], S[1, 6], "Separator",
          S[2, 6], ControlledU[S[2], Rotation[ϕ, S[1, 1]], "Label" → "U"],
          "Separator", Rotation[ϕ / 2, S[2, 3]], S[2, 6], Measurement[S[2]]]
```

Out[•]=




```
In[•]:= out = QuissoExpression[qc] // TrigToExp;
        LogicalForm[QuissoFactor@out, S@{1, 2}]
```

$$\text{Out[•]=}\quad \frac{e^{-\frac{i\phi}{4}}\left(\left|0_{S_1}0_{S_2}\right\rangle + \left|1_{S_1}0_{S_2}\right\rangle\right)}{\sqrt{2}}$$

Example 8 Suppose that a two-qubit system is known to be in one of the four eigenstates of the unitary operator

$$\hat{U} = e^{i\phi} \left(|0\rangle \langle 0| + i\, |1\rangle \langle 1| - |2\rangle \langle 2| - i\, |3\rangle \langle 3| \right). \tag{2.51}$$

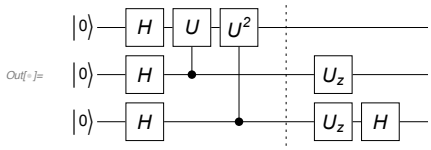
Construct a quantum circuit model to figure out the unknown state using two additional qubits.

Hint: Use the property (2.18) of the Hadamard gate and those of the controlled-U gates, where a unitary operator acts on a two-qubit system controlled by a single qubit.

Solution: This constructs the desired quantum circuit model.

```
In[•]:= qc = QuissoCircuit[LogicalForm[Ket[], S@{1, 2, 3}], S[{1, 2, 3}, 6],
           ControlledU[S[2], Rotation[ϕ, S[1, 1]], "Label" → "U"],
           ControlledU[S[3], Rotation[2 ϕ, S[1, 1]], "Label" → "U²"],
           "Separator", {Rotation[0, S[2, 3]], Rotation[Pi / 2, S[3, 3]]}, S[3, 6]]
```

Out[•]=




```
In[•]:= out = QuissoExpression[qc] // TrigToExp // Simplify;
        LogicalForm[QuissoFactor@out /. ϕ → Pi / 2, S@{1, 2}]
```

Out[•]=

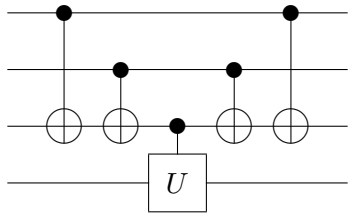
$$\frac{\left(\frac{1}{4}+\frac{i}{4}\right) e^{-\frac{3\,i\,\pi}{4}} \left(\left|0_{S_1}\right\rangle + \left|1_{S_1}\right\rangle \right) \otimes \left(e^{\frac{i\,\pi}{4}} \left|0_{S_2}\right\rangle + \left|1_{S_2}\right\rangle \right) \otimes \left(2 \left|0_{S_3}\right\rangle \right)}{\sqrt{2}}$$

Combining the CNOT gate and the controlled-U gate, one can achieve a variety of conditional gate operations: For example, consider a system consisting of n control qubits and one target qubit, and suppose that you want to operate a unitary gate $\hat{U}$ on the target qubit when one and only one of the control qubits is set to $|1\rangle$,

$$|c\rangle \otimes |t\rangle \mapsto |c\rangle \otimes \hat{U}^{c_1 \oplus \cdots \oplus c_n} |t\rangle \ . \tag{2.52}$$

First operate the CNOT gates with the first $(n-1)$ qubits in the control register consecutively as the control qubit and the last qubit in the control register as the target qubit. It transforms the nth qubit to $|c_1 \oplus \cdots \oplus c_n\rangle$—Problem 5. Then, by applying the controlled-U gate controlled by the nth qubit, the desired operation is implemented. To return the control qubits back to the original state, operate the CNOT gates in the reverse order. Overall, the following quantum circuit model

implements the conditional operation



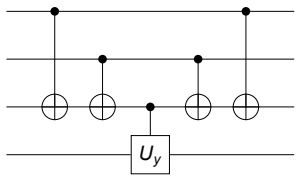
(2.53)

for the case of $n = 3$. This method is used for the implementation of multi-qubit controlled-U gate based on the Gray code—see Section 2.3.

This shows a quantum circuit model conditionally operating the logic gate U on the target qubit.

```
In[•]:= $n = 3;
cc = Table[CNOT[S[j], S[$n]], {j, 1, $n - 1}];
op = Rotation[ϕ, S[$n + 1, 2]];
cU = ControlledU[S[$n], op];
qc = QuissoCircuit[Sequence @@ Flatten@{cc, cU, Reverse@cc}]
```

Out[•]=



This shows how the above quantum circuit model maps the logical basis states. It affects the target qubit only when $c_1 \oplus c_2 \oplus c_3 = 1$.

```
In[•]:= ss = S[Range[$n], None];
        bs = Basis[S@Range[$n + 1]];
        out = Elaborate[qc] ** bs;
        new = QuissoFactor[#, ss] & /@ out;
        Thread[bs → new] // LogicalForm // TableForm
```

Out[•]//TableForm=

$$
\begin{aligned}
&|0_{S_1}0_{S_2}0_{S_3}0_{S_4}\rangle \to |0_{S_1}0_{S_2}0_{S_3}\rangle \otimes |0_{S_4}\rangle\\
&|0_{S_1}0_{S_2}0_{S_3}1_{S_4}\rangle \to |0_{S_1}0_{S_2}0_{S_3}\rangle \otimes |1_{S_4}\rangle\\
&|0_{S_1}0_{S_2}1_{S_3}0_{S_4}\rangle \to |0_{S_1}0_{S_2}1_{S_3}\rangle \otimes \left(\mathrm{Cos}\left[\tfrac{\phi}{2}\right]\ |0_{S_4}\rangle + |1_{S_4}\rangle\ \mathrm{Sin}\left[\tfrac{\phi}{2}\right]\right)\\
&|0_{S_1}0_{S_2}1_{S_3}1_{S_4}\rangle \to |0_{S_1}0_{S_2}1_{S_3}\rangle \otimes \left(\mathrm{Cos}\left[\tfrac{\phi}{2}\right]\ |1_{S_4}\rangle - |0_{S_4}\rangle\ \mathrm{Sin}\left[\tfrac{\phi}{2}\right]\right)\\
&|0_{S_1}1_{S_2}0_{S_3}0_{S_4}\rangle \to |0_{S_1}1_{S_2}0_{S_3}\rangle \otimes \left(\mathrm{Cos}\left[\tfrac{\phi}{2}\right]\ |0_{S_4}\rangle + |1_{S_4}\rangle\ \mathrm{Sin}\left[\tfrac{\phi}{2}\right]\right)\\
&|0_{S_1}1_{S_2}0_{S_3}1_{S_4}\rangle \to |0_{S_1}1_{S_2}0_{S_3}\rangle \otimes \left(\mathrm{Cos}\left[\tfrac{\phi}{2}\right]\ |1_{S_4}\rangle - |0_{S_4}\rangle\ \mathrm{Sin}\left[\tfrac{\phi}{2}\right]\right)\\
&|0_{S_1}1_{S_2}1_{S_3}0_{S_4}\rangle \to |0_{S_1}1_{S_2}1_{S_3}\rangle \otimes |0_{S_4}\rangle\\
&|0_{S_1}1_{S_2}1_{S_3}1_{S_4}\rangle \to |0_{S_1}1_{S_2}1_{S_3}\rangle \otimes |1_{S_4}\rangle\\
&|1_{S_1}0_{S_2}0_{S_3}0_{S_4}\rangle \to |1_{S_1}0_{S_2}0_{S_3}\rangle \otimes \left(\mathrm{Cos}\left[\tfrac{\phi}{2}\right]\ |0_{S_4}\rangle + |1_{S_4}\rangle\ \mathrm{Sin}\left[\tfrac{\phi}{2}\right]\right)\\
&|1_{S_1}0_{S_2}0_{S_3}1_{S_4}\rangle \to |1_{S_1}0_{S_2}0_{S_3}\rangle \otimes \left(\mathrm{Cos}\left[\tfrac{\phi}{2}\right]\ |1_{S_4}\rangle - |0_{S_4}\rangle\ \mathrm{Sin}\left[\tfrac{\phi}{2}\right]\right)\\
&|1_{S_1}0_{S_2}1_{S_3}0_{S_4}\rangle \to |1_{S_1}0_{S_2}1_{S_3}\rangle \otimes |0_{S_4}\rangle\\
&|1_{S_1}0_{S_2}1_{S_3}1_{S_4}\rangle \to |1_{S_1}0_{S_2}1_{S_3}\rangle \otimes |1_{S_4}\rangle\\
&|1_{S_1}1_{S_2}0_{S_3}0_{S_4}\rangle \to |1_{S_1}1_{S_2}0_{S_3}\rangle \otimes |0_{S_4}\rangle\\
&|1_{S_1}1_{S_2}0_{S_3}1_{S_4}\rangle \to |1_{S_1}1_{S_2}0_{S_3}\rangle \otimes |1_{S_4}\rangle\\
&|1_{S_1}1_{S_2}1_{S_3}0_{S_4}\rangle \to |1_{S_1}1_{S_2}1_{S_3}\rangle \otimes \left(\mathrm{Cos}\left[\tfrac{\phi}{2}\right]\ |0_{S_4}\rangle + |1_{S_4}\rangle\ \mathrm{Sin}\left[\tfrac{\phi}{2}\right]\right)\\
&|1_{S_1}1_{S_2}1_{S_3}1_{S_4}\rangle \to |1_{S_1}1_{S_2}1_{S_3}\rangle \otimes \left(\mathrm{Cos}\left[\tfrac{\phi}{2}\right]\ |1_{S_4}\rangle - |0_{S_4}\rangle\ \mathrm{Sin}\left[\tfrac{\phi}{2}\right]\right)
\end{aligned}
$$

How can you implement a controlled-U gate? The operation involves only two qubits, and in principle, it should be possible to implement any specific controlled-U gate. However, as it will get clearer in Chapter 3, the requirements for physical implementation of two-qubit gates is far more difficult to fulfill on realistic systems than single-qubit gates. Fortunately, any controlled-U gate can be implemented using only CNOT gate and single-qubit gates This is one of the basic steps in an establishment of the universal quantum computation.

Let $\hat{U}$ a unitary gate on the second (target) qubit controlled by the first (control) qubit. Suppose that $\hat{U}$ has Euler angles α, β, and γ and an additional phase factor $e^{i\varphi}$ [see (2.28)], $\hat{U} = e^{i\varphi}\hat{U}_z(\alpha)\hat{U}_y(\beta)\hat{U}_z(\gamma)$. Then one can find three unitary operators $\hat{A}$, $\hat{B}$, and $\hat{C}$ such that

$$\hat{U} = e^{i\varphi}\hat{A}\hat{X}\hat{B}\hat{X}\hat{C}\,, \quad \hat{A}\hat{B}\hat{C} = \hat{I}\,, \tag{2.54}$$

where $\hat{X}$ is the Pauli X operator. More explicitly, one common choice is

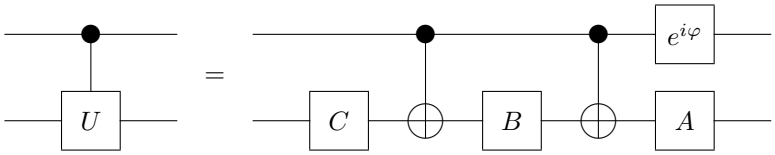
$$\hat{A} = \hat{U}_z(\alpha)\hat{U}_y(\beta/2)\,, \tag{2.55a}$$

$$\hat{B} = \hat{U}_y(-\beta/2)\hat{U}_z(-(\alpha+\gamma)/2)\,, \tag{2.55b}$$

$$\hat{C} = \hat{U}_z(-(\alpha-\gamma)/2). \tag{2.55c}$$

Since $\hat{U}_{y/z}(\phi) = \hat{X}\hat{U}_{y/z}(-\phi)\hat{X}$ for any ϕ, the above choice satisfies the desired properties in (2.54). The properties imply that the controlled-U gate can be

implemented as in the following quantum circuit model



, (2.56)

where the last gate on the first qubit is the *relative* phase shift by φ, $|0\rangle\langle 0| + |1\rangle\langle 1| e^{i\varphi}$. Indeed, when the control qubit is in $|0\rangle$, the two CNOT gates in the middle do nothing, and the combined operator $\hat{A}\hat{B}\hat{C}$ on the target qubit ends up with the identity operator. With the control qubit in $|1\rangle$, on the other hand, the two CNOT gates are operational and the overall operator on the target qubit becomes $\hat{A}\hat{X}\hat{B}\hat{X}\hat{C} = e^{-i\varphi}\hat{U}$, where the phase factor is cancel by the opposite phase from the control qubit.

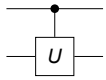
Consider a controlled-U gate.

```
In[•]:= matU = RandomUnitary[2];
        matU // MatrixForm
```

```
Out[•]//MatrixForm=
        ( -0.969594 - 0.22122 i    0.0934503 - 0.0470765 i )
        ( -0.0484934 - 0.0927229 i   -0.916795 - 0.385407 i )
```

```
In[•]:= opU = QuissoExpression[matU, S[2]];
        qc1 = QuissoCircuit[ControlledU[S[1], opU, "Label" → "U"]]
```

Out[•]=



For the decomposition, first find the Euler angles of the unitary operator.

```
In[•]:= detU = Det[matU];
        vphi = Arg[detU] / 2;
        {a, b, c} = TheEulerAngles[matU / Sqrt[detU]]
```

```
Out[•]= {4.15394, 0.0538308, 2.00771}
```

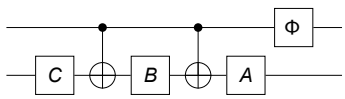
From the Euler angles, choose the component operators.

```
opA = EulerRotation[{a, b / 2, 0}, S[2], "Label" → "A"];
opB = EulerRotation[{0, -b / 2, -(a + c) / 2}, S[2], "Label" → "B"];
opC = EulerRotation[{0, 0, -(a - c) / 2}, S[2], "Label" → "C"];
opD = Phase[vphi, S[1]];
```

Finally, construct the equivalent quantum circuit model.

```
In[•]:= qc2 = QuissoCircuit[opC, CNOT[S[1], S[2]], opB, CNOT[S[1], S[2]], opA, opD]
```

Out[•]=



Check the result.

```
In[•]:= QuissoExpression[qc1] - QuissoExpression[qc2] // Chop
Out[•]= 0
```

2.2.3 General Unitary Gate

Here we decompose an arbitray two-qubit unitary gate into factors of controlled-U gates only. This is a remarkable advantage when one tries to build a quantum computer as you can just focus on how to realize the CNOT gate and single-qubit gates. Moreover, the same idea eventually leads to the proof of the universal quantum computation on a larger system, which will be discussed in Section 2.4.

Consider an aribtrary two-qubit unitary operator $\hat{U}$. In the logical basis, it is represented by a unitary matrix

$$U = \begin{bmatrix} U_{11} & U_{12} & U_{13} & U_{14} \\ U_{21} & U_{22} & U_{23} & U_{24} \\ U_{31} & U_{32} & U_{33} & U_{34} \\ U_{41} & U_{42} & U_{43} & U_{44} \end{bmatrix}. \tag{2.57}$$

To break the unitary operation $\hat{U}$ down into more elementary quantum logic gates, we make use of *two-level unitary transformations*. A two-level unitary transformation is a unitary operation the matrix representation of which acts only on the two columns and rows of other matrices or column or row vectors. The discriptive word "two-level" should not confused with "two-qubit". A two-level transformation acts on multiple qubits, but it just transforms only two rows or columns at a time in the representation. For example, consider a two-level unitary trasformation of the form

$$T_1 = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \tilde{U}_{13}^* & \tilde{U}_{14} \\ & & \tilde{U}_{14}^* & -\tilde{U}_{13} \end{bmatrix}, \tag{2.58}$$

where $\tilde{U}_{ij} \propto U_{ij}$ with normalization factor (unspecified) such that $T_1^\dagger T_1 = T_1 T_1^\dagger = I$. When it multiplies U from the right, it does not change the first two columns of U. It only alters the last two columns; hence the name two-level transformation. The elements in the lower-right subblock of T_1 have been chosen so that it the first element of the last column is canceled:

$$UT_1 = \begin{bmatrix} U_{11} & U_{12} & U'_{13} & 0 \\ U_{21} & U_{22} & U'_{23} & U'_{24} \\ U_{31} & U_{32} & U'_{33} & U'_{34} \\ U_{41} & U_{42} & U'_{43} & U'_{44} \end{bmatrix}. \tag{2.59}$$

Now take another two-level unitary transformation, this time, of the form

$$T_2 = \begin{bmatrix} 1 & & & \\ & \tilde{U}^*_{12} & \tilde{U}'_{13} & \\ & \tilde{U}'^*_{13} & -\tilde{U}_{12} & \\ & & & 1 \end{bmatrix}. \tag{2.60}$$

It removes U'_{13} while keeeping the first and last column as they are:

$$UT_1T_2 = \begin{bmatrix} U_{11} & U''_{12} & 0 & 0 \\ U_{21} & U''_{22} & U''_{23} & U'_{24} \\ U_{31} & U''_{32} & U''_{33} & U'_{34} \\ U_{41} & U''_{42} & U''_{43} & U'_{44} \end{bmatrix}. \tag{2.61}$$

Similarly, we go further with the twol-level unitary matrix

$$T_2 = \begin{bmatrix} \tilde{U}^*_{12} & \tilde{U}'_{13} & & \\ \tilde{U}'^*_{13} & -\tilde{U}_{12} & & \\ & & 1 & \\ & & & 1 \end{bmatrix}. \tag{2.62}$$

to remove the element U''_{12} and get

$$UT_1T_2T_3 = \begin{bmatrix} U'''_{11} & 0 & 0 & 0 \\ 0 & U'''_{22} & U''_{23} & U'_{24} \\ 0 & U'''_{32} & U''_{33} & U'_{34} \\ 0 & U'''_{42} & U''_{43} & U'_{44} \end{bmatrix}. \tag{2.63}$$

At this stage, all elements except for the first of the first column vanish —$U'''_{21} = U'''_{31} = U'''_{41} = 0$—because the product $UT_1T_2T_3$ is a unitary matrix. Repeating the procedure, we can remove all off-diagonal elements to get

$$UT_1T_2 \dots T_L = I. \tag{2.64}$$

As T_j are all unitary, it enables us to rewrite U in the combination of two-level unitary transformation

$$U = T_L^\dagger \dots T_2^\dagger T_1^\dagger. \tag{2.65}$$

Consider a Hermitian operator on a two-qubit system. Physically, it corresponds to a transverse-field Ising model.

```
In[•]:= H = S[1, 1] ** S[2, 1] + S[1, 2] + S[2, 2] + S[1, 3] + S[2, 3]
        matH = Matrix[H];
        matH // MatrixForm
```

Out[•]= $S_1^x S_2^x + S_1^y + S_1^z + S_2^y + S_2^z$

Out[•]//MatrixForm=

$$\begin{pmatrix} 2 & -i & -i & 1 \\ i & 0 & 1 & -i \\ i & 1 & 0 & -i \\ 1 & i & i & -2 \end{pmatrix}$$

This is the unitary operator generated by the above Hermitian operator.

```
In[•]:= U = MultiplyExp[-I H Pi / 4]
```

Out[•]= $e^{-\frac{1}{4} i \pi \left(S_1^x S_2^x + S_1^y S_2^y + S_1^z S_2^z\right)}$

This shows the matrix representation of `U` in the logical basis.

```
In[•]:= matU = Matrix[U] // Simplify;
        matU // MatrixForm
```

Out[•]//MatrixForm=

$$\begin{pmatrix} -\frac{\frac{1}{2}+\frac{5i}{6}}{\sqrt{2}} & -\frac{\frac{1}{6}-\frac{i}{2}}{\sqrt{2}} & -\frac{\frac{1}{6}-\frac{i}{2}}{\sqrt{2}} & \frac{\frac{1}{2}-\frac{i}{2}}{\sqrt{2}} \\ \frac{\frac{1}{6}-\frac{i}{2}}{\sqrt{2}} & \frac{\frac{1}{2}+\frac{i}{6}}{\sqrt{2}} & -\frac{\frac{1}{2}+\frac{5i}{6}}{\sqrt{2}} & -\frac{\frac{1}{2}+\frac{i}{2}}{\sqrt{2}} \\ \frac{\frac{1}{6}-\frac{i}{2}}{\sqrt{2}} & -\frac{\frac{1}{2}+\frac{5i}{6}}{\sqrt{2}} & \frac{\frac{1}{2}+\frac{i}{6}}{\sqrt{2}} & -\frac{\frac{1}{2}+\frac{i}{2}}{\sqrt{2}} \\ \frac{\frac{1}{2}-\frac{i}{2}}{\sqrt{2}} & \frac{\frac{1}{2}+\frac{i}{2}}{\sqrt{2}} & \frac{\frac{1}{2}+\frac{i}{2}}{\sqrt{2}} & -\frac{\frac{1}{2}-\frac{i}{2}}{\sqrt{2}} \end{pmatrix}$$

This shows the decomposition of the unitary matrix into two-level matrices (displaying the first three elements). For the purpose of efficiency, the resulting list is given in terms of `TwoLevelU`.

```
In[•]:= twl = TwoLevelDecomposition[matU] // Simplify;
        twl[[ ;; 3]]
```

Out[•]=

$$\left\{\text{TwoLevelU}\left[\left\{\left\{1, 0\right\}, \left\{0, 1\right\}\right\}, \left\{3, 4\right\}, 4\right],\right.$$

$$\text{TwoLevelU}\left[\left\{\left\{\frac{\frac{1}{2}-\frac{3i}{2}}{\sqrt{7}}, -\frac{\frac{3}{2}+\frac{3i}{2}}{\sqrt{7}}\right\}, \left\{\frac{\frac{3}{2}-\frac{3i}{2}}{\sqrt{7}}, \frac{\frac{1}{2}+\frac{3i}{2}}{\sqrt{7}}\right\}\right\}, \left\{3, 4\right\}, 4\right],$$

$$\left.\text{TwoLevelU}\left[\left\{\left\{-\frac{1-3i}{\sqrt{38}}, \sqrt{\frac{14}{19}}\right\}, \left\{-\sqrt{\frac{14}{19}}, -\frac{1+3i}{\sqrt{38}}\right\}\right\}, \left\{2, 3\right\}, 4\right]\right\}$$

For a more intuitive form, you can convert `TwoLevelU` into the normal matrix form using `Matrix`. Here the first three are shown.

```
In[•]:= MatrixForm /@ Matrix /@ twl[[ ;; 3]]
```

Out[•]=

$$\left\{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{\frac{1}{2}-\frac{3i}{2}}{\sqrt{7}} & -\frac{\frac{3}{2}+\frac{3i}{2}}{\sqrt{7}} \\ 0 & 0 & \frac{\frac{3}{2}-\frac{3i}{2}}{\sqrt{7}} & \frac{\frac{1}{2}+\frac{3i}{2}}{\sqrt{7}} \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -\frac{1-3i}{\sqrt{38}} & \sqrt{\frac{14}{19}} & 0 \\ 0 & -\sqrt{\frac{14}{19}} & -\frac{1+3i}{\sqrt{38}} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}\right\}$$

Indeed, they reconstruct the original matrix.

```
In[•]:= new = Dot @@ Matrix /@ twl;
        matU - new // Simplify // MatrixForm
```

Out[•]//MatrixForm=

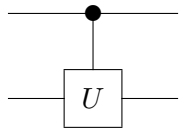
$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

We are still to express the two-level unitary transformation in terms of a controlled-U gate: The two-level unitary matrix—for example, T_1 in (2.58)—of the form

$$\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & U_{11} & U_{12} \\ & & U_{21} & U_{22} \end{bmatrix}, \tag{2.66}$$

is already in the form of the matrix representation of a controlled-U gate in (2.47), and just correponds to a single controlled-U:

$$\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & U_{11} & U_{12} \\ & & U_{21} & U_{22} \end{bmatrix} = \quad . \tag{2.67}$$



Consider a two-level matrix of the form.

```
In[•]:= matU = TheHadamard[];
        code = TwoLevelU[matU, {3, 4}, 4];
        full = Matrix[code];
        full // MatrixForm
```

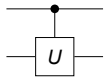
Out[•]//MatrixForm=

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

It corresponds to a single controlled-U gate.

```
In[•]:= ctrlU = GrayTwoLevelU[matU, {3, 4}, S@{1, 2}];
        QuissoCircuit[ctrlU]
```

Out[•]=

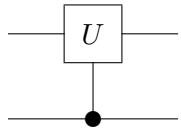


A two-level unitary matrix of the form

$$\begin{bmatrix} 1 & & & \\ & U_{11} & & U_{12} \\ & & 1 & \\ & U_{21} & & U_{22} \end{bmatrix} \tag{2.68}$$

also corresponds to a single controlled-U gate with the control and target qubit exchanged—here the first qubit is the target qubit and the second the control qubit:

$$\begin{bmatrix} 1 & & & \\ & U_{11} & & U_{12} \\ & & 1 & \\ & U_{21} & & U_{22} \end{bmatrix} = \quad . \tag{2.69}$$



Consider a two-level matrix of the form.

```
In[•]:= matU = TheHadamard[];
        code = TwoLevelU[matU, {2, 4}, 4];
        full = Matrix[code];
        full // MatrixForm
```

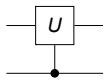
Out[•]//MatrixForm=

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

It corresponds to a single controlled-U gate with the control and target qubit exchanged.

```
In[•]:= ctrlU = GrayTwoLevelU[matU, {2, 4}, S@{1, 2}];
        QuissoCircuit[ctrlU]
```

Out[•]=

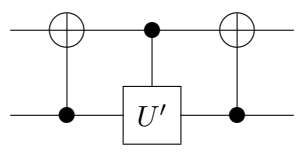


More complicated is the two-level unitary matrix of the form

$$\begin{bmatrix} 1 & & & \\ & U_{11} & U_{12} & \\ & U_{21} & U_{22} & \\ & & & 1 \end{bmatrix} . \tag{2.70}$$

As it affects both qubits simultaneously, it cannot be represented by a single controlled-U gate. However, it is possible to bring it to the form (2.67) by exchanging the second and forth columns and rows. The specified exchanges correspond to flipping the bit values of the first qubit only when the second qubit has the value 1, that is, the CNOT gate with the second qubit as the control qubit and the first qubit as the target qubit. Through the exchange, the unitary matrix U itself is modified and the rows and columns are exchanged, which corresponds to the basis change by the Pauli X matrix, $U \to U' = XUX$. Putting all together, the two-level unitary matrix is implemented as

$$\begin{bmatrix} 1 & & & \\ & U_{11} & U_{12} & \\ & U_{21} & U_{22} & \\ & & & 1 \end{bmatrix} = \tag{2.71}$$



.

Consider a two-level matrix of the form.

```
In[•]:= matU = TheHadamard[];
        code = TwoLevelU[matU, {2, 3}, 4];
        full = Matrix[code];
        full // MatrixForm
```

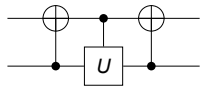
Out[•]//MatrixForm=

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

In this case, we need to apply the CNOT gate before and after the controlled-U gate.

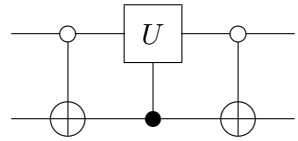
```
In[•]:= ctrlU = GrayTwoLevelU[matU, {2, 3}, S@{1, 2}];
        QuissoCircuit[ctrlU]
```

Out[•]=

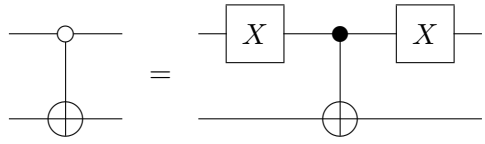


In a similar manner, we can implement another form of two-level unitary matrix as

$$\begin{bmatrix} U_{11} & & & U_{12} \\ & 1 & & \\ & & 1 & \\ U_{21} & & & U_{22} \end{bmatrix} = \quad . \tag{2.72}$$



Here we have adopted a short-hand diagram for the modified-CNOT gate, which flips the bit value of the target qubit when the control qubit is in the state $|0\rangle$ rather than $|1\rangle$. It is achieved by operating the Pauli X gate before and after the usual CNOT gate,



$$= \quad . \tag{2.73}$$

Consider a two-level matrix of the form.

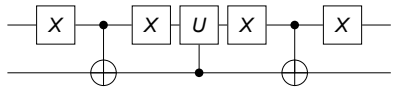
```
In[•]:= matU = TheHadamard[];
        code = TwoLevelU[matU, {1, 4}, 4];
        full = Matrix[code];
        full // MatrixForm
```

Out[•]//MatrixForm=

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

```
In[•]:= ctrlU = GrayTwoLevelU[matU, {1, 4}, S@{1, 2}];
        QuissoCircuit@@ctrlU
```

Out[•]=



So far, we have figured out the implementations of 4×4 two-level unitary matrices of different forms just by simple inspection. For more than two qubits, the size of the two-level unitary matrices is much larger and it is difficult to find proper implementations in such a way. Fortunately, there is a systematic way based on the Gray code, which will be discussed later in Section 2.4.

In summary, an arbitrary two-qubit unitary gate can be carried out by first decomposing its matrix representation into two-level unitary matrices and then implementing the two-level unitary matrices by means of the CNOT gate and the controlled-U gate.

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

```
In[•]:= H = S[1, 1] ** S[2, 1] + S[1, 2] + S[2, 2] + S[1, 3] + S[2, 3]
        U = MultiplyExp[-I H Pi / 4]
        matU = Matrix[U];
```

Out[•]= $S_1^x S_2^x + S_1^y + S_1^z + S_2^y + S_2^z$

Out[•]= $e^{-\frac{1}{4} i \pi \left(S_1^x S_2^x + S_1^y + S_1^z + S_2^y + S_2^z\right)}$

This is the decomposition into the two-level matrices, just showing the first four.

```
In[•]:= twl = TwoLevelDecomposition[matU] // Simplify;
        MatrixForm /@ Matrix /@ twl[[ ;; 3]]
```

Out[•]=
$$\left\{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{\frac{1}{2}-\frac{3 i}{2}}{\sqrt{7}} & -\frac{\frac{3}{2}+\frac{3 i}{2}}{\sqrt{7}} \\ 0 & 0 & \frac{\frac{3}{2}-\frac{3 i}{2}}{\sqrt{7}} & \frac{\frac{1}{2}+\frac{3 i}{2}}{\sqrt{7}} \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -\frac{1-3 i}{\sqrt{38}} & \sqrt{\frac{14}{19}} & 0 \\ 0 & -\sqrt{\frac{14}{19}} & -\frac{1+3 i}{\sqrt{38}} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}\right\}$$

The two-level matrices are written in terms of the controlled-U and CNOT gate, again showing the first four. The first of the list `twl` happens to be the identity matrix in this case, and it is excluded in the further analysis.

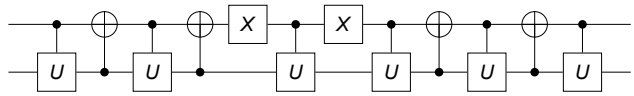
```
In[•]:= gates = GrayTwoLevelU[#, S@{1, 2}] & /@ Rest[twl];
        gates[[ ;; 3]]
```

Out[•]=
$$\left\{\left\{\text{ControlledU}\left[\{S_1\}, \frac{1}{2\sqrt{7}} - \frac{3 i S_2^x}{2\sqrt{7}} - \frac{3 i S_2^y}{2\sqrt{7}} - \frac{3 i S_2^z}{2\sqrt{7}}, \text{Label} \to U\right]\right\}, \left\{\text{CNOT}[\{S_2\}, \{S_1\}],\right.\right.$$
$$\text{ControlledU}\left[\{S_1\}, -\frac{1}{\sqrt{38}} - i\sqrt{\frac{14}{19}}\, S_2^y - \frac{3 i S_2^z}{\sqrt{38}}, \text{Label} \to U\right], \text{CNOT}[\{S_2\}, \{S_1\}]\Big\},$$
$$\left.\left\{\text{ControlledU}\left[\{S_1\}, -\frac{5}{14\sqrt{2}} + \frac{3}{14} i \sqrt{19}\, S_2^y - \frac{5 i S_2^z}{14\sqrt{2}}, \text{Label} \to U\right]\right\}\right\}$$

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

```
In[•]:= qc = QuissoCircuit @@ Reverse@Flatten[gates]
```

Out[•]=



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

```
In[•]:= new = Matrix[qc] // Simplify;
        new // MatrixForm
```

Out[•]//MatrixForm=

$$\begin{pmatrix} -\frac{1}{2}+\frac{5i}{6\sqrt{2}} & -\frac{1}{6\sqrt{2}}-\frac{i}{2} & -\frac{1}{6\sqrt{2}}-\frac{i}{2} & \frac{1}{2\sqrt{2}}-\frac{i}{2} \\ \frac{1}{6\sqrt{2}}-\frac{i}{2} & \frac{1}{2\sqrt{2}}+\frac{i}{6} & -\frac{1}{2}+\frac{5i}{6\sqrt{2}} & -\frac{1}{2\sqrt{2}}+\frac{i}{2} \\ \frac{1}{6\sqrt{2}}-\frac{i}{2} & -\frac{1}{2}+\frac{5i}{6\sqrt{2}} & \frac{1}{2}+\frac{i}{6\sqrt{2}} & -\frac{1}{2\sqrt{2}}+\frac{i}{2} \\ \frac{1}{2\sqrt{2}}-\frac{i}{2} & \frac{1}{2\sqrt{2}}+\frac{i}{2} & \frac{1}{2}+\frac{i}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}}+\frac{i}{2} \end{pmatrix}$$

```
In[•]:= new - matU // Simplify // MatrixForm
```

Out[•]//MatrixForm=

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

2.3 Multi-Qubit Controlled Gates

Let $\mathcal{S}^{\otimes m}$ and $\mathcal{S}^{\otimes n}$ be the Hilbert spaces of the control and target register consisting of m and n qubits, respectively. Suppose that $\hat{U}$ is a unitary operator on the target register. The multi-qubit controlled-U operator is defined by

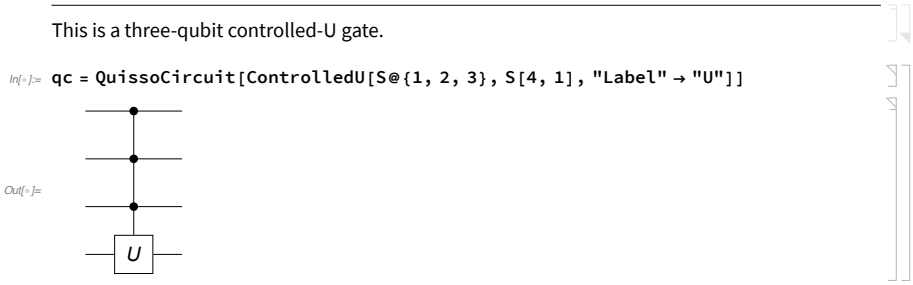
$$\mathrm{Ctrl}(\hat{U}) : |c\rangle \otimes |t\rangle \mapsto |c\rangle \otimes \hat{U}^{c_1 c_2 \cdots c_m} |t\rangle \ , \tag{2.74}$$

where $c \equiv (c_1 c_2 \ldots c_m)_2$. The unitary transformation $\hat{U}$ acts on the target qubits only when every control qubit is set to $|1\rangle$. We will be most interested in the case with $n = 1$, where the matrix representation takes the form

$$\mathrm{Ctrl}(\hat{U}) \doteq \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & U_{11} & U_{12} \\ & & & U_{21} & U_{22} \end{bmatrix} . \tag{2.75}$$

It is the prototype form of a two-level unitary matrix on $(m+1)$ qubits. Indeed, any two-level unitary matrix can be put into this form by exchanging columns

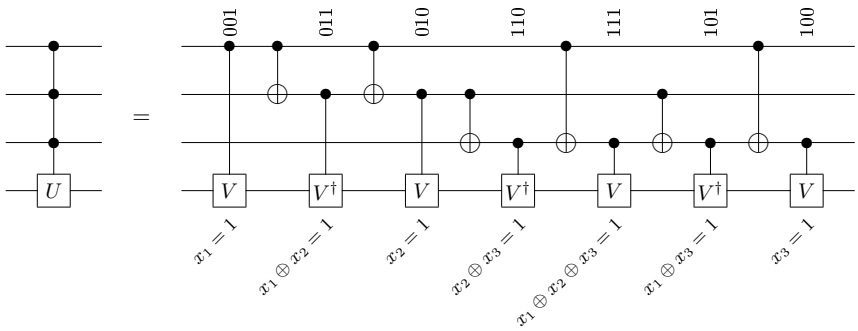
and rows—equivalent to basis changes. Multi-qubit controlled-U gates thus arise naturally as we discuss the universal quantum computation in Section 2.4.



A reliable implementation of multi-qubit controlled-$\hat{U}$ gate is essential in many quantum algorithms. A notable example is the quantum oracle (see Section 4.2.1), which is a key component of quantum decision problems as well as quantum search problems. Here we introduce some widely known methods to implement it efficiently.

2.3.1 Gray Code

We first discuss a systematic method based on the Gray code to decompose a multi-qubit controlled-U gate into factors of either the single-qubit controlled-U or CNOT gate: For example, consider a 3-qubit controlled-U gate as shown in the following quantum circuit model (Barenco *et al.*, 1995)



, (2.76)

where $\hat{V}$ is another unitary operator such that $\hat{V}^4 = \hat{U}$. When every control qubit is set to $|1\rangle$, all $\hat{V}$ gates in the diagram take effects on the target qubit while $\hat{V}^\dagger$ gates are ineffective. To examine the case with the contol qubits in a general state $|x_1\rangle \otimes \cdots \otimes |x_n\rangle$, note that the gates on the target qubit are effective under the conditions specified in terms of the bit values at the bottom of the diagram—see also Eq. (2.53). The conditions are fulfilled systematically by following the Gray code sequence,[1] the strings of which are indicated at the top of the diagram. The

[1]The *Gray code sequence* is an arrange of bits such that two successive values differ in only one bit.

identity for the bitwise AND,

$$2^{n-1}(x_1 x_2 \cdots x_n) = \sum_{k_1} x_{k_1} - \sum_{k_1<k_2} (x_{k_1} \oplus x_{k_2}) + \sum_{k_1<k_2<k_3} (x_{k_1} \oplus x_{k_2} \oplus x_{k_3}) - \cdots + (-1)^{n-1}(x_1 \oplus x_2 \oplus \cdots \oplus x_n)\,, \quad (2.77)$$

ensures that the quantum circuit model on the right-hand side of (2.76) reproduces the desired multi-qubit controlled-U gate on the left-hand side.

In general, a n-qubit controlled-U gate can be implemented by combining 2^{n-1} controlled-V gates, $(2^{n-1}-1)$ controlled-$V^\dagger$ gates, and (2^n-2) CNOT gates, where $\hat{V}$ is a unitary operator satisfying $\hat{V}^{2^{n-1}} = \hat{U}$. For relatively small n ($n \leq 8$), the Gray code is known to be the most efficient method. However, it grows exponentially and eventually becomes impractical. For those cases, several methods have been proposed where the computational cost increases quadratically with the size of the quantum register (Barenco *et al.*, 1995; Nielsen & Chuang, 2011).

Consider a three-qubit register as an example.

```
$L = 3;
jj = Range[$L];
cc = S[jj, None];
```

This is the gate to operate on the target qubit.

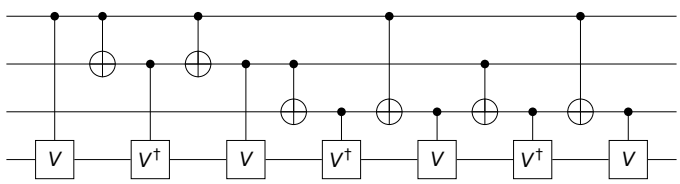
```
In[•]:= T = S[$L + 1, None];
        op = QuissoRotation[Pi / 3, T[1]]
Out[•]= Sqrt[3]/2 - (i S_4^x)/2
```

This decomposes the multi-qubit controlled-U gate based on the Gray code -- showing only a part of it. Each component is either CNOT or a two-qubit controlled-U gate.

```
In[•]:= gc = GrayControlledU[cc, op]; gc[[ ;; 3]]
Out[•]= {ControlledU[{S_1}, (2^(1/4) e^(-i π/24) + 2^(1/4) e^(i π/24))/(2 × 2^(1/4)) + ((2^(1/4) e^(-i π/24) - 2^(1/4) e^(i π/24)) S_4^x)/(2 × 2^(1/4)), Label → V],
         CNOT[{S_1}, {S_2}],
         ControlledU[{S_2}, (2^(1/4) e^(-i π/24) + 2^(1/4) e^(i π/24))/(2 × 2^(1/4)) + ((-2^(1/4) e^(-i π/24) + 2^(1/4) e^(i π/24)) S_4^x)/(2 × 2^(1/4)), Label → V^†]}
```

This is a quantum circuit model of the decomposition.

```
In[•]:= qc = QuissoCircuit[gc]
```



Finally check the result.

```
In[•]:= expr = QuissoExpression[qc];
        expr2 = QuissoControlledU[cc, op];
        expr - expr2 // Garner
Out[•]= 0
```

2.3.2 Multi-Qubit Controlled-NOT

The multi-qubit controlled-NOT gate is an important special case of the multi-qubit controlled-U gate, where the unitary operator $\hat{U}$ equals to the Pauli $\hat{X}$. It transforms the logical basis states as [see Eq. (2.74)]

$$|c\rangle \otimes |t\rangle \mapsto |c\rangle \otimes \hat{X}^{c_1 c_2 \cdots c_m} |t\rangle \ , \tag{2.78a}$$

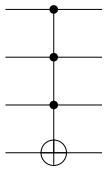
or equivalently,

$$|c\rangle \otimes |t\rangle \mapsto |c\rangle \otimes |(c_1 c_2 \cdots c_n) \oplus t\rangle \ . \tag{2.78b}$$

The multi-qubit controlled-NOT gate commonly occurs when one converts the two-level unitary transformation—see Section 2.2.3—on more than two qubits. Indeed, we will generalize the procedure and discuss a systematic way of conversion based on the Gray code in Section 2.4.

This shows a generalized CNOT gate, controlled by three qubits rather than a single qubit.

```
In[•]:= qc = QuissoCircuit[CNOT[S@{1, 2, 3}, S[4]]]
```



Here is the explicit expression of the multi-qubit CNOT gate in terms of the Pauli operators.

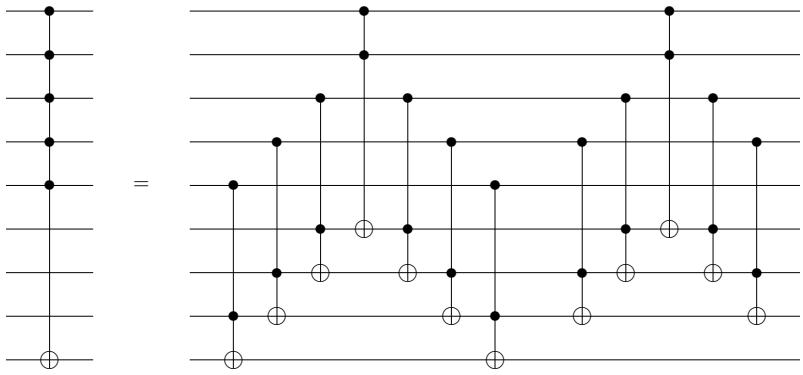
```
In[•]:= op = QuissoExpression[qc]
```

$$\text{Out}[\bullet]= \frac{7}{8} - \frac{1}{8} S_1^z S_2^z - \frac{1}{8} S_1^z S_3^z - \frac{1}{8} S_1^z S_4^x - \frac{1}{8} S_2^z S_3^z - \frac{1}{8} S_2^z S_4^x - \frac{1}{8} S_3^z S_4^x + \frac{1}{8} S_1^z S_2^z S_3^z + \frac{1}{8} S_1^z S_2^z S_4^x + \frac{1}{8} S_1^z S_3^z S_4^x + \frac{1}{8} S_2^z S_3^z S_4^x - \frac{1}{8} S_1^z S_2^z S_3^z S_4^x + \frac{S_1^z}{8} + \frac{S_2^z}{8} + \frac{S_3^z}{8} + \frac{S_4^x}{8}$$

Here we compare it with the expression explicitly constructed in terms of a projection operator.

```
In[•]:= prj = Multiply @@ S[{1, 2, 3}, 11];
        new = (1 - prj) + prj ** S[4, 1]
        op - new // Elaborate
Out[•]= 1 - (|1⟩⟨1|)_S1 (|1⟩⟨1|)_S2 (|1⟩⟨1|)_S3 + (|1⟩⟨1|)_S1 (|1⟩⟨1|)_S2 (|1⟩⟨1|)_S3 S4^x

Out[•]= 0
```

An efficient implementation of a multi-qubit CNOT gate uses additional qubits not directly involvined in the gate operation itself as in the followign quantum circuit model (Barenco *et al.*, 1995):



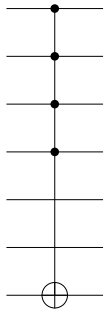
(2.79)

It is emphasized that the quantum states of the extra qubits should not be confused with the so-called "ancillary qubits" in the sense that they do not have to be initialized in a certain fixed quantum state and their state is not altered. The desired gate operation is performed properly regardless of the initial state of the extra qubits and their quantum state is restored after the gate operation.

Here we demonstrate a multi-qubit CNOT gate.

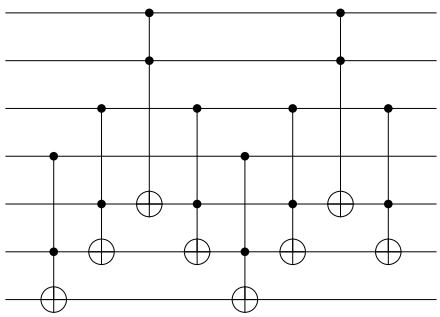
```
$n = 4;
$m = 2;
cc = Range[$n];
aa = Range[$n + 1, $n + $m];
tt = $n + $m + 1;

In[•]:= qc = QuissoCircuit[CNOT[S[cc], S[tt]], "Visible" → S[aa]]
```

```
Out[•]=
```



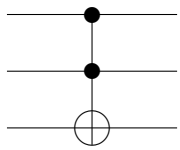
```
In[•]:= tofa = Table[Toffoli[S[$n - j + 1], S[$n + $m - j + 1], S[tt - j + 1]], {j, 1, $m}];
        tofb = Toffoli[S[1], S[2], S[$n + 1]];
        tofc = Rest@tofa;
        new = QuissoCircuit[
          Sequence @@ Flatten@{tofa, tofb, Reverse@tofa, tofc, tofb, Reverse@tofc}]
```

Out[•]=



```
In[•]:= Elaborate[qc - new]
Out[•]= 0
```

In the quantum circuit model (2.79), we have introduced another special case of multi-qubit CNOT gates, which is called the *Toffolli gate*, denoted by the quantum circuit element



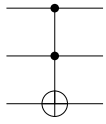
. (2.80)

The Toffoli gate has attracted interested as it is univesal for classical reversible computation. Unfortunately, however, it is not universal for quantum computation.

This is a quantum circuit model of the Toffoli gate.

```
In[•]:= qc = QuissoCircuit[Toffoli[S[1], S[2], S[3]]]
        toff = QuissoExpression[qc]
```

Out[•]=

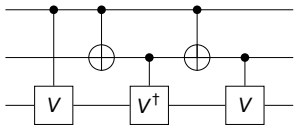


Out[•]= $\frac{3}{4} - \frac{1}{4} S_1^z S_2^z - \frac{1}{4} S_1^z S_3^x - \frac{1}{4} S_2^z S_3^x + \frac{1}{4} S_1^z S_2^z S_3^x + \frac{S_1^z}{4} + \frac{S_2^z}{4} + \frac{S_3^x}{4}$

It can be implemented by a combination of two-qubit gates.

```
In[•]:= gray = GrayControlledU[S@{1, 2}, S[3, 1]];
        qc = QuissoCircuit[gray]
        expr = QuissoExpression[qc]
        toff - expr
```

Out[•]=



Out[•]= $\frac{3}{4} - \frac{1}{4} S_1^z S_2^z - \frac{1}{4} S_1^z S_3^x - \frac{1}{4} S_2^z S_3^x + \frac{1}{4} S_1^z S_2^z S_3^x + \frac{S_1^z}{4} + \frac{S_2^z}{4} + \frac{S_3^x}{4}$

Out[•]= 0

Here $V = \sqrt{X}$, and its matrix representation looks like this.

```
In[•]:= matV = MatrixPower[ThePauli[1], 1 / 2];
        matV * 2 / (1 + I) // MatrixForm
        opV = QuissoExpression[matV, S[3]]
```

Out[•]//MatrixForm=

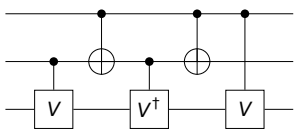
$$\begin{pmatrix} 1 & -i \\ -i & 1 \end{pmatrix}$$

Out[•]= $\left(\frac{1}{2} + \frac{i}{2}\right) + \left(\frac{1}{2} - \frac{i}{2}\right) S_3^x$

Reversing the above circuit gives the identical result.

```
In[•]:= qc = QuissoCircuit[Reverse@gray]
        expr = QuissoExpression[qc]
        toff - expr
```

Out[•]=



Out[•]= $\frac{3}{4} - \frac{1}{4} S_1^z S_2^z - \frac{1}{4} S_1^z S_3^x - \frac{1}{4} S_2^z S_3^x + \frac{1}{4} S_1^z S_2^z S_3^x + \frac{S_1^z}{4} + \frac{S_2^z}{4} + \frac{S_3^x}{4}$

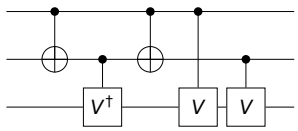
Out[•]= 0

As noted by Smolin & DiVincenzo (1996), it can also be reordered as following. This rearrangement is useful in optimizing the *Fredkin gate*, another universal gate for classical reversible computation.

This shows another slightly different implementation of the Toffoli gate.

```
In[•]:= qc = QuissoCircuit[Permute[gray, Cycles[{{4, 3, 2, 1}}]]]
        expr = QuissoExpression[qc]
        toff - expr
```

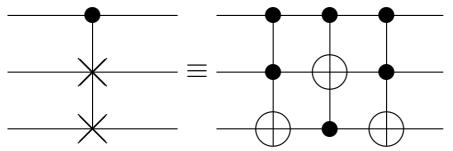
Out[•]=



$$\text{Out[•]=}\quad \frac{3}{4}-\frac{1}{4}S_1^z S_2^z-\frac{1}{4}S_1^z S_3^x-\frac{1}{4}S_2^z S_3^x+\frac{1}{4}S_1^z S_2^z S_3^x+\frac{S_1^z}{4}+\frac{S_2^z}{4}+\frac{S_3^x}{4}$$

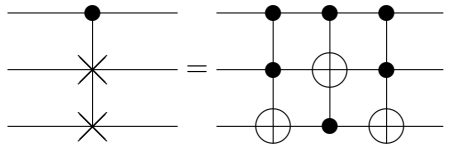
```
Out[•]= 0
```

The Fredkin gate "swaps" the states of the two target qubits when the control qubit is set in the state $|1\rangle$, as depicted by the following two equivalent quantum circuit elements



(2.81)

In fact, the relation between the SWAP gate and the CNOT gate suggests that there is another equivalent quantum circuit model of the Fredkin gate,



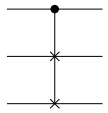
(2.82)

which is simpler than the above. This equivalent quantum circuit model was used by Smolin & DiVincenzo (1996) for an efficient implement of the Fredkin gate in terms of the elementary gates: Like the Toffoli gate, the Fredkin gate is not universal for quantum computation.

This shows the quantum circuit model of the quantum Fredkin gate.

```
In[•]:= qc = QuissoCircuit[Fredkin[S[1], S[2], S[3]]]
```

Out[•]=



This is an explicit operator expression of the Fredkin gate in terms of the Pauli operators.

```
In[•]:= op = Elaborate[qc]
```

$$\text{Out[•]=}\quad \frac{3}{4}+\frac{1}{4}S_2^x S_3^x+\frac{1}{4}S_2^y S_3^y+\frac{1}{4}S_2^z S_3^z-\frac{1}{4}S_1^z S_2^x S_3^x-\frac{1}{4}S_1^z S_2^y S_3^y-\frac{1}{4}S_1^z S_2^z S_3^z+\frac{S_1^z}{4}$$

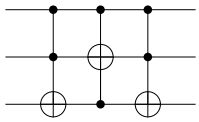
This is the matrix representation of the Fredkin gate in the logical basis.

```
In[•]:= mat = Matrix[qc];
        mat // MatrixForm
Out[•]//MatrixForm=
```

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

The Fredkin gate is equivalent to a combination of three Toffoli gates.

```
In[•]:= qc2 = QuissoCircuit[
          Toffoli[S[1], S[2], S[3]],
          Toffoli[S[1], S[3], S[2]],
          Toffoli[S[1], S[2], S[3]]]
Out[•]=
```



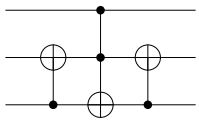
```
In[•]:= op2 = Elaborate[qc2]
```

$$\text{Out}[\bullet]= \frac{3}{4} + \frac{1}{4} S_2^x S_3^x + \frac{1}{4} S_2^y S_3^y + \frac{1}{4} S_2^z S_3^z - \frac{1}{4} S_1^z S_2^x S_3^x - \frac{1}{4} S_1^z S_2^y S_3^y - \frac{1}{4} S_1^z S_2^z S_3^z + \frac{S_1^z}{4}$$

```
In[•]:= op - op2
Out[•]= 0
```

In fact, the relation between the SWAP gate and the CNOT gate suggests that there is another equivalent quantum circuit model of the Fredkin gate.

```
In[•]:= new = QuissoCircuit[
          CNOT[S[3], S[2]],
          Toffoli[S[1], S[2], S[3]],
          CNOT[S[3], S[2]]
        ]
Out[•]=
```



```
In[•]:= qc - new // Elaborate
Out[•]= 0
```

2.4 Universal Quantum Computation

In classical computation, it is known that a finite set of logic gates—typically including AND, OR, and NOT—is sufficient to calculate any binary function. The set is said to be *universal* for classical computation. One can ask whether there exists a universal set of elementary quantum logic gates for quantum computation that enables to implement any arbitrary quantum unitary operation? In this section, we examine this question.

Consider a system of n qubits. Suppose that we want to implement an arbitrary unitary operation $\hat{U}$ on the n-qubit register. We start by decomposing $\hat{U}$ into a set of two-level unitary transformations, which we discussed for the case of two-qubit systems in Section 2.2.3.

Consider a three-qubit system, and an arbitrary unitary operation on it.

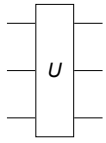
```
In[•]:= $n = 3;
        SS = S[Range[$n], None];
        mat = RandomUnitary[2^$n];
        mat[[ ;; 3, ;; 3]] // MatrixForm
```

Out[•]//MatrixForm=

$$\begin{pmatrix} -0.289955+0.087655\,i & 0.095513-0.0895934\,i & 0.167464-0.303327\,i \\ -0.269555-0.0172308\,i & -0.178701-0.205085\,i & 0.00218922+0.56633\,i \\ -0.191626+0.127416\,i & 0.0926635+0.233537\,i & -0.0087163+0.0102968\,i \end{pmatrix}$$

```
In[•]:= op = QuissoExpression[mat, SS];
        qc = QuissoCircuit[{op, "Label" → "U"}]
```

Out[•]=




```
In[•]:= twl = TwoLevelDecomposition[mat];
        MatrixForm /@ Matrix /@ twl[[ ;; 3]] // TableForm
```

Out[•]//TableForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1.+0.\,i & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0191735+0.999816\,i \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.45304-0.218959\,i & 0.118465-0.856024\,i \\ 0 & 0 & 0 & 0 & 0 & 0 & -0.118465-0.856024\,i & 0.45304+0.218959\,i \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -0.102438+0.122007\,i & 0.987229 & 0 \\ 0 & 0 & 0 & 0 & 0 & -0.987229 & -0.102438-0.122007\,i & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Now a remaining question is how to implement the two-level unitary transformation in terms of elementary gates.

Let us consider a particular two-level unitary matrix on a three-qubit system. We want to implement it in terms of elementary quantum logic gates.

```
In[•]:= x = 4; y = 5;
        U = TheRotation[Pi / 3, 2];
        mat = Matrix@TwoLevelU[U, {x, y}, 2^$n];
        mat // MatrixForm
```

Out[•]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\sqrt{3}}{2} & -\frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{\sqrt{3}}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

```
In[•]:= op = QuissoExpression[mat, SS]
```

$$\text{Out}[\bullet]= \frac{1}{8}\left(6+\sqrt{3}\right)+\frac{1}{8}\left(2-\sqrt{3}\right)S_1^z S_2^z+\frac{1}{8}\left(2-\sqrt{3}\right)S_1^z S_3^z+$$
$$\frac{1}{8}\left(-2+\sqrt{3}\right)S_2^z S_3^z+\frac{1}{8}\,i\,S_1^x S_2^x S_3^y+\frac{1}{8}\,i\,S_1^x S_2^y S_3^x-\frac{1}{8}\,i\,S_1^y S_2^x S_3^x+\frac{1}{8}\,i\,S_1^y S_2^y S_3^y$$

Our implementation is based on the Gray code sequence. Notice the function **Reverse**.

```
In[•]:= gates = Flatten@GrayTwoLevelU[U, {x, y}, SS]
```

$$\text{Out}[\bullet]= \Big\{S_1^x,\ \text{CNOT}[\{S_1, S_2\}, \{S_3\}],\ S_1^x,\ S_3^x,\ \text{CNOT}[\{S_2, S_3\}, \{S_1\}],$$
$$S_3^x,\ \text{CNOT}[\{S_1, S_2\}, \{S_3\}],\ \text{CNOT}[\{S_1, S_3\}, \{S_2\}],\ S_2^x,$$
$$\text{ControlledU}\Big[\{S_1, S_2\},\ \frac{\sqrt{3}}{2}+\frac{i\,S_3^y}{2},\ \text{Label}\to U\Big],\ S_2^x,\ \text{CNOT}[\{S_1, S_3\}, \{S_2\}],$$
$$\text{CNOT}[\{S_1, S_2\}, \{S_3\}],\ S_3^x,\ \text{CNOT}[\{S_2, S_3\}, \{S_1\}],\ S_3^x,\ S_1^x,\ \text{CNOT}[\{S_1, S_2\}, \{S_3\}],\ S_1^x\Big\}$$

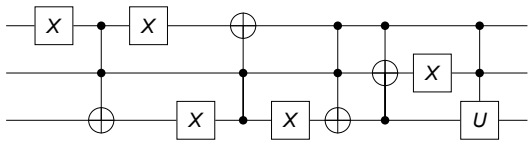
```
In[•]:= new = Apply[Dot, Matrix[#, SS] & /@ Elaborate /@ gates] // Simplify;
        new // MatrixForm
```

Out[•]//MatrixForm=

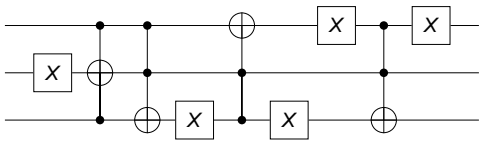
$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\sqrt{3}}{2} & -\frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{\sqrt{3}}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

```
In[•]:= qc1 = QuissoCircuit[gates[[ ;; 10]]]
        qc2 = QuissoCircuit[gates[[11 ;;]]]
```

Out[•]=



Out[•]=



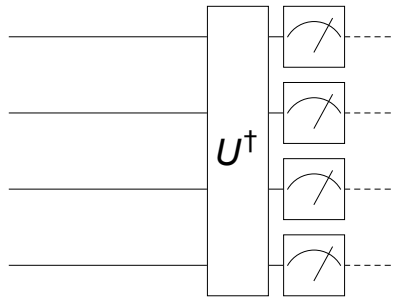


Figure 2.4: Measurement in a basis $\{|\alpha_x\rangle\}$ other than the logical basis. The unitary operator $\hat{U}$ here corresponds to the basis change $|\alpha_y\rangle = \hat{U}|y\rangle = \sum_x |x\rangle\langle x|\alpha_y\rangle$.

```
In[•]:= new = Matrix[qc2].Matrix[qc1] // Simplify;
        new // MatrixForm
Out[•]//MatrixForm=
```

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\sqrt{3}}{2} & -\frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{\sqrt{3}}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

2.5 Measurements

We conclude this chapter with a few discussions on measurement. In quantum computers, measurement is assumed to be performed independently on individual qubits in the logical basis, $\{|x\rangle : x = 0, 1, \ldots, 2^n - 1\}$.

What if a measurement in another basis, say, $\{|\alpha_x\rangle = \hat{U}|x\rangle\}$ is required? We require that the input state $|\alpha_x\rangle$ should end up with the logical basis state $|x\rangle$ with unit probability so that the measurement yields the outcome x. This process is described by the measurment operator $\hat{P}_x := |x\rangle\langle\alpha_x| = |x\rangle\langle x|\hat{U}^\dagger$. Evidently they satisfy the condition $\sum_x \hat{P}_x^\dagger \hat{P}_x = \hat{I}$ for measurement operators (see Postulate 1.3). Now we note that $\hat{E}_x := |x\rangle\langle x|$ is nothing but the measurement operators in the logical basis state. This implies that simply by applying the inverse unitary operation $\hat{U}^\dagger$ before the measurement, the measurement in the new basis $\{|\alpha_x\rangle\}$ can be achieved through the measurement in the logical basis; see Fig. 2.4. For example, suppose that a qubit is in the state $|\psi\rangle = |0\rangle c_0 + |1\rangle c_1$. By default, a measurement is assumed to be in the logical basis and the measurement statistics reflects the probability distribution $P_0 = |c_0|^2$ and $P_1 = |c_1|^2$ as illustrated in the demonstration below.

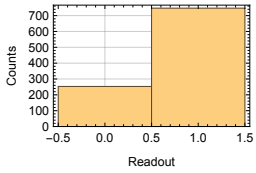
First consider a measurement in the logical basis.

```
In[•]:= Let[Complex, c]
        vec = ProductState[S[1] → {c[0], c[1]}];
        qc = QuissoCircuit[vec, "Spacer", Measurement[S[1]], "PortSize" → {2, 0.2}]
```

Out[•]= $|0\rangle c_0 + |1\rangle c_1$ —— [measurement] - - -

```
In[•]:= Block[
         {c, data},
         c[0] = 1 / 2;
         c[1] = Sqrt[3] / 2;
         data = Table[out = QuissoExpression[qc];
           Readout[out, S[1]], 1000];
         Histogram[data, FrameLabel → {"Readout", "Counts"}, ImageSize → Small]
        ]
```

Out[•]=



We now want to make a measurement on the eigenbasis $\{|\pm\rangle\}$ of the Pauli X operator. In this case, it is the Hadamard gate that gives the desired basis change. In the new basis, the state vector $|\psi\rangle$ is given by

$$|\psi\rangle = |+\rangle \frac{c_0 + c_1}{\sqrt{2}} + |-\rangle \frac{c_0 - c_1}{\sqrt{2}}, \tag{2.83}$$

Now the measurement statistics is in accordance with the probability distribution $P_\pm = |c_0 \pm c_1|^2/2$.

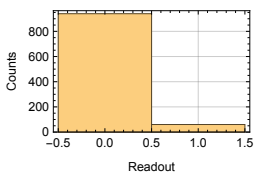
Now consider a measurement in the eigen-basis of the Pauli X operator.

```
In[•]:= Let[Complex, c]
        vec = ProductState[S[1] → {c[0], c[1]}];
        qc = QuissoCircuit[vec, S[1, 6], Measurement[S[1]], "PortSize" → {2, 0.2}]
```

Out[•]= $|0\rangle c_0 + |1\rangle c_1$ —— [H] [measurement] - - -

```
In[•]:= Block[
         {c, data},
         c[0] = 1 / 2;
         c[1] = Sqrt[3] / 2;
         data = Table[out = QuissoExpression[qc];
           Readout[out, S[1]], 1000];
         Histogram[data, FrameLabel → {"Readout", "Counts"}, ImageSize → Small]
        ]
```

Out[•]=



Another interesting example is the *Bell measurement*, that is, the measurement on two qubits in the basis of Bell states,

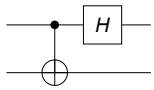
$$
\begin{aligned}
|\beta_0\rangle &= \frac{|00\rangle + |11\rangle}{\sqrt{2}} \\
|\beta_1\rangle &= \frac{|01\rangle + |10\rangle}{\sqrt{2}} \\
|\beta_2\rangle &= \frac{|01\rangle - |10\rangle}{\sqrt{2}} \\
|\beta_3\rangle &= \frac{|00\rangle - |11\rangle}{\sqrt{2}}
\end{aligned}
\tag{2.84}
$$

Recall that the Bell states can be generated from the logical basis states by the so-called entangler, a combination of the Hadamard gate and the CNOT gate; see Section 2.2.1. Therefore, the Bell measurement can be achieved by applying the inverse of the entangler operation.

This is the "disentangler" quantum circuit, which is the inverse of the entangler quantum circuit .

```
In[•]:= disentangler = QuissoCircuit[CNOT[S[1], S[2]], S[1, 6]]
```

Out[•]=



This shows that the disentangler quantum circuit maps the Bell states into the logical basis states.

```
In[•]:= op = QuissoExpression[disentangler];
        bs = BellState@S@{1, 2};
        Thread[bs → op ** bs] // LogicalForm // TableForm
```

Out[•]//TableForm=

$$\frac{|0_{S_1}0_{S_2}\rangle + |1_{S_1}1_{S_2}\rangle}{\sqrt{2}} \to |0_{S_1}0_{S_2}\rangle$$
$$\frac{|0_{S_1}1_{S_2}\rangle + |1_{S_1}0_{S_2}\rangle}{\sqrt{2}} \to |0_{S_1}1_{S_2}\rangle$$
$$\frac{|0_{S_1}1_{S_2}\rangle - |1_{S_1}0_{S_2}\rangle}{\sqrt{2}} \to |1_{S_1}1_{S_2}\rangle$$
$$\frac{|0_{S_1}0_{S_2}\rangle - |1_{S_1}1_{S_2}\rangle}{\sqrt{2}} \to |1_{S_1}0_{S_2}\rangle$$

For fast quantum algorithms and quantum error corrections, more sophisticated meausurements may be necessary. A common example is the quantum phase estimation, which is one of the core parts of Shor's factorization algorithm. We will discuss it in Section 4.4.

Problems

1. Let $\hat{\Phi}(\phi)$ be the *phase gate*, which gives rise to a *relative* phase shift by $\phi \in [0, 2\pi)$,

$$\hat{\Phi}(\phi) : |0\rangle \mapsto |0\rangle \ , \quad |1\rangle \mapsto |1\rangle \, e^{i\phi} \, . \tag{2.85}$$

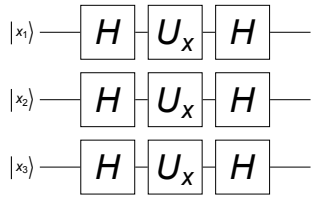
   (a) Show that on a n-qubit quantum register,

$$[\hat{\Phi}(\phi)]^{\otimes n} \, |x\rangle = |x\rangle \, e^{i\phi(x_1 + \cdots + x_n)} \tag{2.86}$$

   for any $x = 0, 1, \ldots, 2^n - 1$.

   (b) Evaluate explicitly the state $[\hat{\Phi}(\phi)]^{\otimes n} \hat{H}^{\otimes n} \, |0\rangle$, where $\hat{H}$ is the Hadamard gate.

2. Let $\hat{U}_x(\phi)$ be the rotatin around the x-axis by angle ϕ on a single qubit. Analyze and evaluate explicitly the following quantum circuit model



, (2.87)

   where $|x\rangle := |x_1\rangle \otimes |x_2\rangle \otimes |x_3\rangle$ is a 3-qubit logical basis state and the labels "H" and "U_x" indicate the Hadamard gate $\hat{H}$ and the rotation gate $\hat{U}_x(\phi)$, respectively. Generalize the result and show that

$$\hat{H}^{\otimes n} [\hat{U}_x(\phi)]^{\otimes n} \hat{H}^{\otimes n} \, |x\rangle = e^{-i\phi n/2} \, |x\rangle \, e^{i\phi(x_1 + \cdots x_n)} \tag{2.88}$$

   for any $x = 0, 1, \ldots, 2^n - 1$.

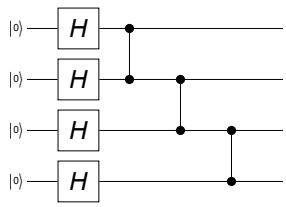
3. Let $\hat{S}^\mu$ be the Pauli operarors. Show that

$$e^{i\hat{S}^\mu \phi/2} \hat{S}^\nu e^{-i\hat{S}^\mu \phi/2} = \hat{S}^\nu \cos(\phi) + \hat{S}^\lambda \epsilon_{\lambda\mu\nu} \sin(\phi) \tag{2.89}$$

   for all $\mu, \nu = x, y, z$ and $\mu \neq \nu$.

4. Consider a quantum register of four qubits.

   (a) Analyze the following quantum circuit model

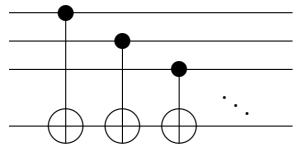


(2.90)

   and evaluate the resulting state $|\Psi\rangle$ explicitly. The state is a so-called *cluster state* or *graph state*, a crucial resource in the measurement-based quantum computation—see Section 3.4.

(b) Show that in the state $|\Psi\rangle$ from (b), every qubit is *maximally entangled* with the rest qubits, that is, the *reduced density matrix* $\hat{\rho}_j$ of the jth qubit, $\hat{\rho}_j := \mathrm{Tr}_{k\neq j} |\Psi\rangle\langle\Psi|$, is given by $\hat{\rho}_j = \hat{I}/2$—it exhibits coherence in no basis.

5. Consider the following quantum circuit model consisting of the CNOT gates on a n-qubit quantum register



(2.91)

(a) Find the output state for the input of a logical basis state $|x_1\rangle \otimes \cdots \otimes |x_n\rangle \in \mathcal{S}^{\otimes n}$.

(b) Find the output state for the input state $|+\rangle \otimes \cdots \otimes |+\rangle \otimes |-\rangle$, where $|\pm\rangle := (|0\rangle \pm |1\rangle)/\sqrt{2}$.

6. Let $P(i \leftrightarrow j)$ be the matrix exchanging the ith and jth rows (columns) of vectors/matrices. For example, on a four-dimenisonal space,

$$P(1 \leftrightarrow 2) = \begin{bmatrix} 0 & 1 & & \\ 1 & 0 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} . \tag{2.92}$$

$\hat{P}(i \leftrightarrow j)$ is the corresponding operator.

(a) Find the quantum circuit model for $\hat{P}(2 \leftrightarrow 3)$ on a two-qubit system (four-dimensional vector space $\mathcal{S} \otimes \mathcal{S}$) using CNOT gates only.

(b) Find the quantum circuit model for $\hat{P}(2 \leftrightarrow 3)$ on a three-qubit system ($\mathcal{S}^{\otimes 3}$) using only mult-qubit CNOT gates only.

(c) Find the quantum circuit model for $\hat{P}(4 \leftrightarrow 7)$ on a three-qubit system ($\mathcal{S}^{\otimes 3}$) using only multi-qubit CNOT gates onlys.