Definitions of Quantum Gates

This notebook is based on

1. M. Nielsen and I. Chuang, Quantum Computation and Quantum Information, Cambridge University Press, 2010 (tenth anniversary edition)
2. Bittner, Eric R. Quantum Dynamics: Applications in Biological and Materials Systems. CRC Press, 2009.
3. arXiv:1403.7050 [quant-ph]

Single-qubit pseudo spin operators: $\mathbb{1}$, $\hat{\sigma}_x$, $\hat{\sigma}_y$, $\hat{\sigma}_z$

```
In[•]:= {id, σx, σy, σz} = Table[SparseArray[PauliMatrix[i]], {i, 0, 3}];
```

Projectors onto $\left|0\right\rangle$ and $\left|1\right\rangle$ states

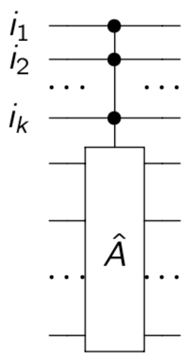
```
In[•]:= proj0 = (id + σz)/2 // SparseArray;
        proj1 = (id - σz)/2 // SparseArray;
```

Operator $\hat{a}$ acting on the k^{th} qubit of a set of n qubits: $\mathbb{1}\otimes\mathbb{1}\otimes\cdots\otimes\mathbb{1}\otimes\hat{a}\otimes 1\otimes\cdots\otimes\mathbb{1}\otimes\mathbb{1}$

```
In[•]:= op[n_Integer, k_Integer, a_] /; 1 ≤ k ≤ n ∧ Dimensions[a] == {2, 2} :=
         KroneckerProduct[IdentityMatrix[2^(k-1), SparseArray],
          a, IdentityMatrix[2^(n-k), SparseArray]]
```

Controlled operator

In a set of n qubits, the n-qubit operator `ctrl[n,λ,A]` acts like the operator $\hat{A}$ if all qubits in the list $\lambda = \{i_1, i_2, \ldots, i_k\}$ are in the $\left|1\right\rangle$ state, and has no action (acts like the identity operator, $\mathbb{1}$ on n qubits) if any of the qubits in the list λ are in the $\left|0\right\rangle$ state:



$$\text{CTRL} = \left[\bigotimes_{j=1}^{k} |1\rangle\langle 1|^{(i_j)}\right] \cdot \hat{A} + \left[\mathbb{1} - \bigotimes_{j=1}^{k} |1\rangle\langle 1|^{(i_j)}\right] \cdot \mathbb{1} = \mathbb{1} + \left[\bigotimes_{j=1}^{k} |1\rangle\langle 1|^{(i_j)}\right] \cdot (\hat{A} - \mathbb{1})$$

In[•]:=
```
ctrl[n_Integer, λ_ /; VectorQ[λ, IntegerQ], A_] /;
   (Unequal @@ λ) ∧ Min[λ] ≥ 1 ∧ Max[λ] ≤ n ∧ Dimensions[A] == {2^n, 2^n} :=
  IdentityMatrix[2^n, SparseArray] +
   Apply[Dot, op[n, #, proj1] & /@ λ].(A - IdentityMatrix[2^n, SparseArray])
```

Single-Qubit Gates

Pauli gates

Pauli gates acting on the k^{th} qubit of a set of n qubits:

- The Pauli-*X* gate $-\boxed{X}-$ acts like $\hat{\sigma}_x$ on the desired qubit, and has no effect on all other qubits.

In[•]:=
```
xGate[n_Integer, k_Integer] /; 1 ≤ k ≤ n := op[n, k, σx]
```

- The Pauli-*Y* gate $-\boxed{Y}-$ acts like $\hat{\sigma}_y$ on the desired qubit, and has no effect on all other qubits.

In[•]:=
```
yGate[n_Integer, k_Integer] /; 1 ≤ k ≤ n := op[n, k, σy]
```

- The Pauli-*Z* gate $-\boxed{Z}-$ acts like $\hat{\sigma}_z$ on the desired qubit, and has no effect on all other qubits.

In[•]:=
```
zGate[n_Integer, k_Integer] /; 1 ≤ k ≤ n := op[n, k, σz]
```

Pauli rotations

- $\hat{R}_x(\phi) \propto e^{-i\,\phi\,\hat{\sigma}_x/2}$

In[•]:=
```
rxGate[n_Integer, k_Integer, ϕ_] /; 1 ≤ k ≤ n := op[n, k, (1 + ⅇ^(ⅈ ϕ))/2 id + (1 - ⅇ^(ⅈ ϕ))/2 σx]
```

- $\hat{R}_y(\phi) \propto e^{-i\,\phi\,\hat{\sigma}_y/2}$

In[•]:=
```
ryGate[n_Integer, k_Integer, ϕ_] /; 1 ≤ k ≤ n := op[n, k, (1 + ⅇ^(ⅈ ϕ))/2 id + (1 - ⅇ^(ⅈ ϕ))/2 σy]
```

- $\hat{R}_z(\phi) \propto e^{-i\,\phi\,\hat{\sigma}_z/2}$

In[•]:=
```
rzGate[n_Integer, k_Integer, ϕ_] /; 1 ≤ k ≤ n := op[n, k, (1 + ⅇ^(ⅈ ϕ))/2 id + (1 - ⅇ^(ⅈ ϕ))/2 σz]
```

Hadamard gate

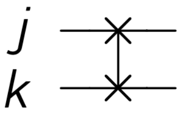
The Hadamard gate $-\boxed{H}-$ acts like $\frac{\hat{\sigma}_x+\hat{\sigma}_z}{\sqrt{2}}$ on the desired qubit, and has no effect on all other qubits.

In[•]:=
```
hadamardGate[n_Integer, k_Integer] /; 1 ≤ k ≤ n := op[n, k, (σx + σz)/√2]
```

Two-qubit gates

SWAP gate

Exchanges the state of qubits *j* and *k* in a set of *n* qubits



$$\mathrm{SWAP}^{(jk)} = (\mathbb{1}^{(j)} \otimes \mathbb{1}^{(k)} + \hat{\sigma}_x^{(j)} \otimes \hat{\sigma}_x^{(k)} + \hat{\sigma}_y^{(j)} \otimes \hat{\sigma}_y^{(k)} + \hat{\sigma}_z^{(j)} \otimes \hat{\sigma}_z^{(k)})$$

In[•]:=
```
swapGate[n_Integer, {j_Integer, k_Integer}] /; 1 ≤ j ≤ n ∧ 1 ≤ k ≤ n ∧ j ≠ k :=
 (IdentityMatrix[2^n, SparseArray] + xGate[n, j].xGate[n, k] +
   yGate[n, j].yGate[n, k] + zGate[n, j].zGate[n, k]) / 2
```

The matrix representation of a two - qubit SWAP takes on the familiar form

In[•]:=
```
swapGate[2, {1, 2}] // MatrixForm
```

Out[•]//MatrixForm=

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

Square root of the SWAP gate

In[•]:=
```
sqrtSwapGate[n_Integer, {j_Integer, k_Integer}] /; 1 ≤ j ≤ n ∧ 1 ≤ k ≤ n ∧ j ≠ k :=
 (3 + i)/4 IdentityMatrix[2^n, SparseArray] +
  (1 - i)/4 (xGate[n, j].xGate[n, k] + yGate[n, j].yGate[n, k] + zGate[n, j].zGate[n, k])
```

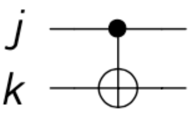
In[•]:=
```
sqrtSwapGate[2, {1, 2}] // MatrixForm
```

Out[•]//MatrixForm=

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

CNOT gate

In a set of *n* qubits, if qubit *j* is in state $|0\rangle$ then there is no effect on qubit *k*, whereas if qubit *j* is in state $|1\rangle$ then the NOT operator $\hat{\sigma}_x$ acts on qubit *k*.



$$\mathrm{CNOT}^{(jk)} = |0\rangle\langle 0|^{(j)} \otimes \mathbb{1}^{(k)} + |1\rangle\langle 1|^{(j)} \otimes \sigma_x^{(k)}$$

CNOT is simply the CTRL operator with a single element in the list $\lambda = \{j\}$ and a single-qubit $\hat{A} = \hat{\sigma}_x{}^{(k)}$ operator.

In[∘]:=
```
cnotGate[n_Integer, j_Integer → k_Integer] /; 1 ≤ j ≤ n ∧ 1 ≤ k ≤ n ∧ j ≠ k :=
  ctrl[n, {j}, op[n, k, σx]]
```

In[∘]:=
```
MatrixForm@cnotGate[2, 1 → 2]
```

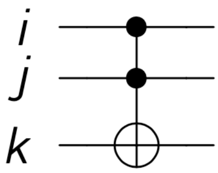
Out[∘]//MatrixForm=

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

Three-qubit gates

CCNOT (Toffoli) gate

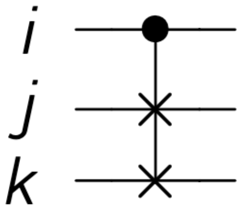
In a set of n qubits, if both qubits i and j are in state 1, then the NOT operator $\hat{\sigma}_x$ acts on qubit k,



In[∘]:=
```
ccnotGate[n_Integer, {i_Integer, j_Integer} → k_Integer] /;
   1 ≤ i ≤ n ∧ 1 ≤ j ≤ n ∧ 1 ≤ k ≤ n ∧ Unequal[i, j, k] := ctrl[n, {i, j}, op[n, k, σx]]
```

Controlled SWAP (Fredkin) gate

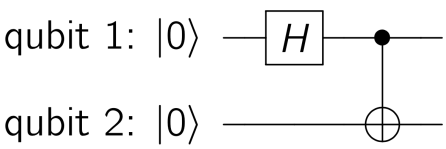
In a set of n qubits, if qubit i is in state 0 then there is no effect on qubits j and k, whereas if qubit i is in state 1 then the state of qubits j and k is exchanged,



In[∘]:=
```
cswapGate[n_Integer, i_Integer → {j_Integer, k_Integer}] /;
   1 ≤ i ≤ n ∧ 1 ≤ j ≤ n ∧ 1 ≤ k ≤ n ∧ Unequal[i, j, k] := ctrl[n, {i}, swapGate[n, {j, k}]]
```

A Simple Circuit

As a simple example, we study the quantum circuit,



The operator for the circuit,

In[•]:=
```
s = cnotGate[2, 1 → 2].hadamardGate[2, 1];
MatrixForm[s]
```

Out[•]//MatrixForm=

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

the basis set is $\mathcal{B}_2 = \left\{ \left|00\right\rangle, \left|01\right\rangle, \left|10\right\rangle, \left|11\right\rangle \right\}$

In[•]:=
```
bas[n_Integer /; n ≥ 1] := Tuples[{0, 1}, n]
bas[2]
```

Out[•]= {{0, 0}, {0, 1}, {1, 0}, {1, 1}}

operate on the $\left|\psi_{\text{in}}\right\rangle = \left|0\right\rangle \otimes \left|0\right\rangle = \left|00\right\rangle$ input state:

In[•]:=
```
ψin = {1, 0, 0, 0};
ψout = s.ψin;
MatrixForm@ψout
```

Out[•]//MatrixForm=

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

$\mathcal{B}_2$ shows that we have $\left|\psi_{\text{out}}\right\rangle = \frac{\left|00\right\rangle + \left|11\right\rangle}{\sqrt{2}}$, and projective measurements on the two qubits, give $\frac{1}{2}$ probability of finding the classical result $\left|00\right\rangle$ and $\frac{1}{2}$ probability of finding $\left|11\right\rangle$, whereas the bit combinations $\left|10\right\rangle$ and $\left|01\right\rangle$ never occur

In[•]:=
```
Abs[ψout]^2
```

Out[•]= $\left\{\frac{1}{2}, 0, 0, \frac{1}{2}\right\}$

Quantum Fourier Transform

The quantum circuit for a QFT is assembled from blocks that connect the i^{th} qubit to the qubits $\{i+1, i+2, \ldots, n\}$ via controlled phase gates:

In[•]:=
```
qftBlock[n_Integer, i_Integer] /; 1 ≤ i ≤ n :=
 Apply[Dot, Table[ctrl[n, {j}, rzGate[n, i, 2 π / 2^(j+1-i)]], {j, n, i + 1, -1}]].
  hadamardGate[n, i]
```

assemble the quantum Fourier transformation by connecting all qubits:

1. Connect all qubits through the above QFTblock operation
2. Swap the order of the qubits

In[•]:=
```
qft[n_Integer] /; n ≥ 1 := Apply[Dot, Table[swapGate[n, {i, n + 1 - i}], {i, 1, n/2}]].
  Apply[Dot, Table[qftBlock[n, i], {i, n, 1, -1}]]
```

QFT consists of a polynomial number of gates:

- n Hadamard gates
- $\frac{n(n-1)}{2}$ controlled rotations
- $\left\lfloor \frac{n}{2} \right\rfloor$ swap gates

simple formula for the resulting matrix representation:

In[•]:=
```
Table[qft[n] == 1/2^(n/2) Table[ⅇ^(2 π ⅈ j k/2^n), {j, 0, 2^n - 1}, {k, 0, 2^n - 1}], {n, 6}] //
 FullSimplify
```

Out[•]=
```
{True, True, True, True, True, True}
```

Quantum phase estimation

estimate the phase of this:

In[•]:=
```
w = {1};
u[ϕ_] = {{ⅇ^(2 π ⅈ ϕ)}};
```

check that we set up the problem correctly:

In[•]:=
```
{u[ϕ].w === ⅇ^(2 π ⅈ ϕ) w, Norm[w] == 1}
```

Out[•]=
```
{True, True}
```

unit operator in the space of the system U:

In[•]:=
```
u0 = IdentityMatrix[Length[w], SparseArray];
```

U operator attached to n qubits, controlled by the i^{th} qubit:

In[•]:=
```
ctrlU[n_Integer, i_Integer, ϕ_] /; 1 ≤ i ≤ n :=
 KroneckerProduct[op[n, i, proj0], u0] + KroneckerProduct[op[n, i, proj1], u[ϕ]]
```

estimate to t digits of precision:

In[•]:=
```
t = 4;
ϵ[ϕ_] = KroneckerProduct[ConjugateTranspose[qft[t]], u0].
    Apply[Dot, Table[ctrlU[t, i, 2^(t-i) ϕ], {i, t, 1, -1}]].
    KroneckerProduct[Apply[Dot, Table[hadamardGate[t, i], {i, t}]], u0].
    Flatten[KroneckerProduct[SparseArray[1 → 1, 2^t], w]] // Normal;
```

probabilities for measuring the different basis states: trace out the SUT and look at the diagonal elements of the reduced density matrix
(see ReducedDensityMatrix.nb)

In[•]:=
```
rdm[ψABC_?VectorQ, {dA_Integer /; dA ≥ 1, dB_Integer /; dB ≥ 1,
    dC_Integer /; dC ≥ 1}] /; Length[ψABC] == dA dB dC :=
  With[{P = ArrayReshape[ψABC, {dA, dB, dC}]},
   Flatten[Transpose[P, {1, 3, 2}].ConjugateTranspose[P], {{1, 2}, {3, 4}}]]
traceout[ψ_?VectorQ, d_Integer /; d ≥ 1] /; Divisible[Length[ψ], d] :=
  rdm[ψ, {1, d, Length[ψ]/d}]
traceout[ψ_?VectorQ, d_Integer /; d ≤ -1] /; Divisible[Length[ψ], -d] :=
  rdm[ψ, {Length[ψ]/(-d), -d, 1}]
```

In[•]:=
```
prob[ϕ_?NumericQ] := Re[Diagonal[traceout[ϵ[N[ϕ]], -Length[w]]]]
```

When ϕ is an integer multiple of 2^{-t}, only one basis state has 100% probability of occurring. The i^{th} basis state corresponds to a measurement of $\phi = \frac{i-1}{2^t}$:

In[•]:=
```
Table[prob[ϕ], {ϕ, 0, 1, 2^-t}] // Chop // MatrixForm
```

Out[•]//MatrixForm=

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

When ϕ is not an integer multiple of 2^{-t}, all basis states can occur in measurement:

In[•]:=
```
Round[prob[0.2], 0.001]
```

Out[•]=
```
{0.004, 0.008, 0.025, 0.876, 0.055, 0.011, 0.005,
 0.003, 0.002, 0.002, 0.001, 0.001, 0.001, 0.002, 0.002, 0.003}
```

The mean measurement is a bad estimator (doesn't converge as $t \to \infty$).
The most likely measurement is a good estimator.

In[∘]:=
```
mean[ϕ_?NumericQ] := prob[ϕ] . Range[0, 2^t - 1] / 2^t;
mostprobable[ϕ_?NumericQ] := (Ordering[prob[ϕ], -1]⟦1⟧ - 1) / 2^t;
Plot[{ϕ, mean[ϕ], mostprobable[ϕ]}, {ϕ, 0, 1},
 AxesLabel → {"phase setting", "phase measurement"},
 PlotRange → {{0, 1}, {0, 1}}, PlotRangePadding → None,
 PlotStyle → {Black, {Thick, Blue}, {Thick, Red}}, AspectRatio → {1},
 PlotLegends → {"exact", "mean of measurements", "most frequent measurement"},
 GridLines → {Range[1, 2^t - 1] / 2^t, Range[1, 2^t - 1] / 2^t}]
```

Out[∘]=