CPEN 400Q / EECE 571Q Lecture 20
Crash course on Hamiltonian simulation

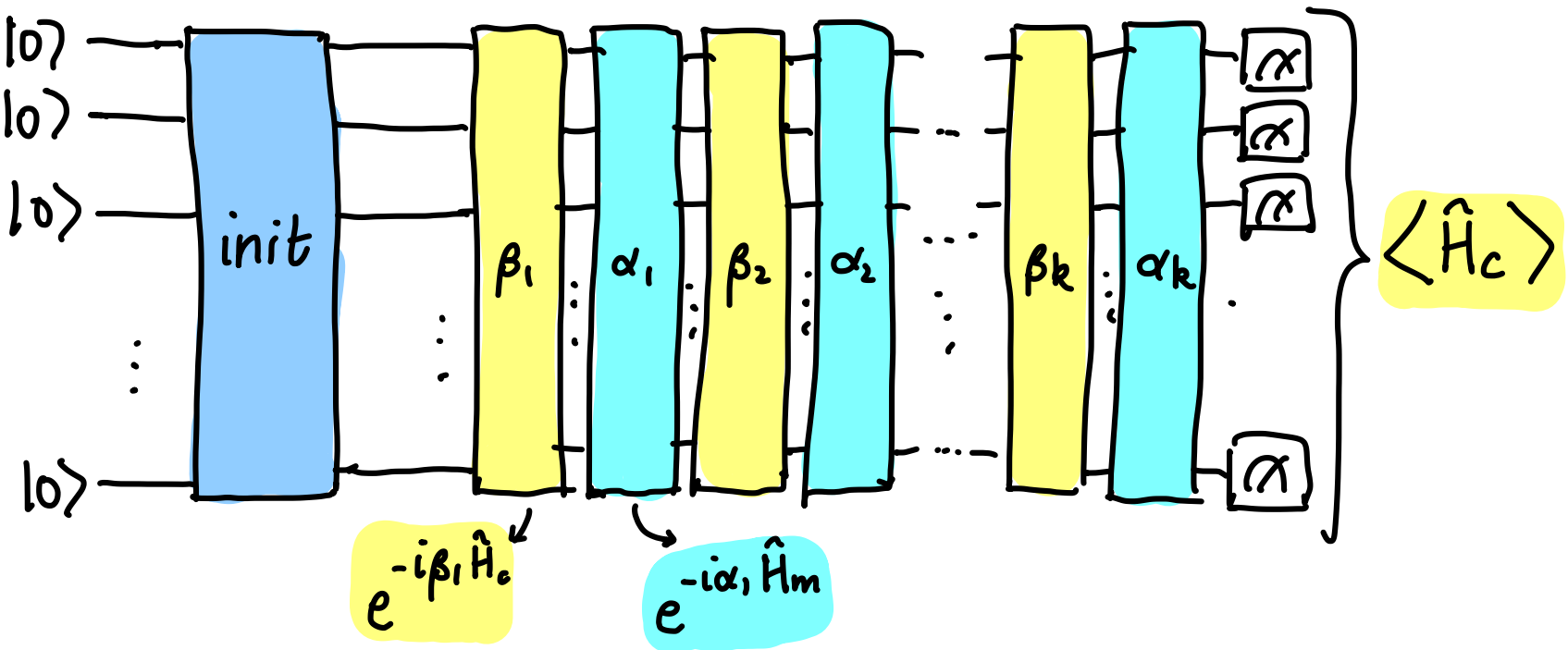
Thursday 24 March 2022

Announcements

- Assignment 4 available (due Friday 8 April at 23:59)
- Project presentations start next Tuesday
  - Do not come to the classroom: all presentations on Zoom (links in Canvas, same as previous class link)
  - Optionally recorded, only with your full group's consent
  - Check grading rubric for what should be included in presentation (in particular, should have some live coding / demonstration of results)

Last time

We implemented the quantum approximate optimization algorithm and used it to solve a small combinatorial optimization problem.



Last time

During this process, I introduced to you a new PennyLane operation, `qml.ApproxTimeEvolution`:

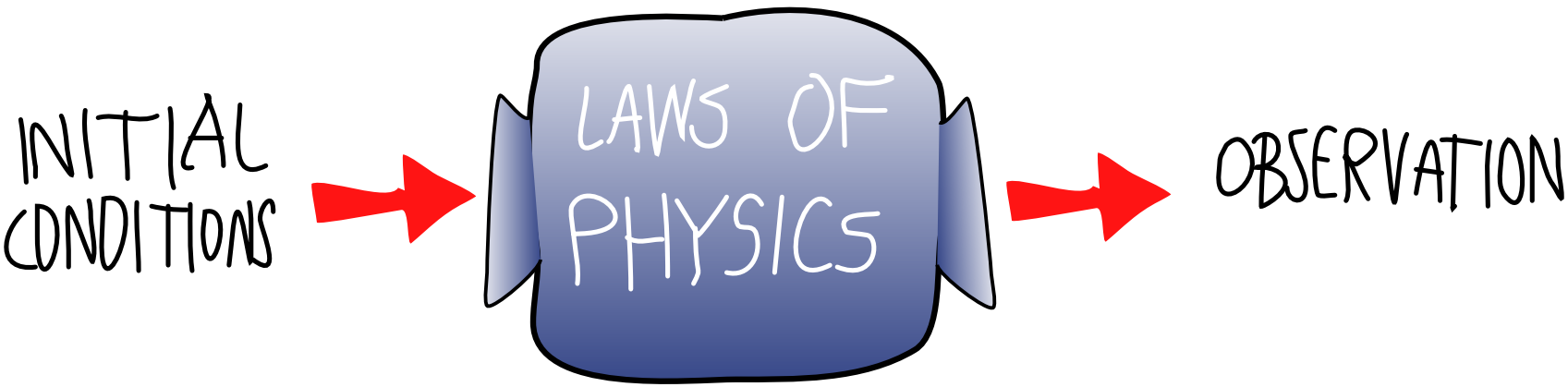
```
@qml.qnode(dev)
def qaoa_circuit(beta_c, alpha_m):
  # Initalize
  for wire in range(5):
    qml.Hadamard(wires=wire)

  # Apply alternating layers
  for i in range(len(beta_c)):
    qml.ApproxTimeEvolution(cost_h, beta_c[i], 1)
    qml.ApproxTimeEvolution(mixer_h, alpha_m[i], 1)

  # Return expval of cost H
  return qml.expval(cost_h)
```

This leads us naturally to Hamiltonian simulation.

Learning outcomes

- Turn exponentials of Pauli operators into sequences of elementary gates
- Trotterize a Hamiltonian
- Estimate the Trotterization error incurred in simulating a Hamiltonian

Motivation



The role of a physicist is to work out what happens in the middle.

Image credit: Xanadu Quantum Codebook node H.1

Motivation

It is not easy in general to describe mathematically how a system evolves with time. We can use computers to approximately model or simulate their behaviour; this involves *discretizing* the systems.

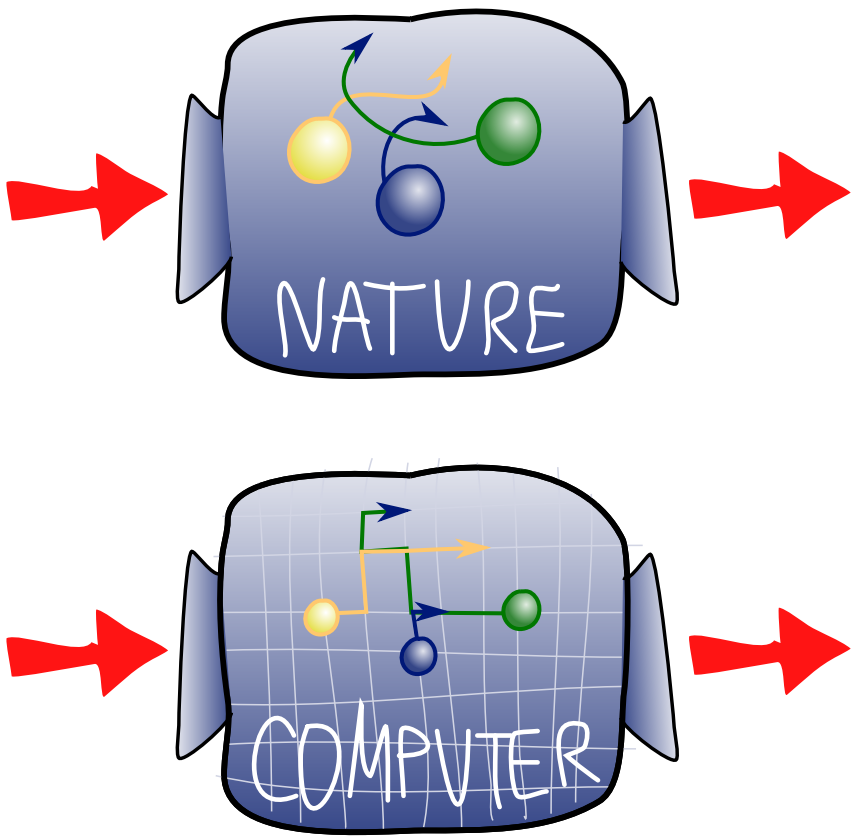


Image credit: Xanadu Quantum Codebook node H.1

Hamiltonians

We saw, when discussing VQE, that **Hamiltonians** are the "energy operators" of systems, and describe how different parts interact.

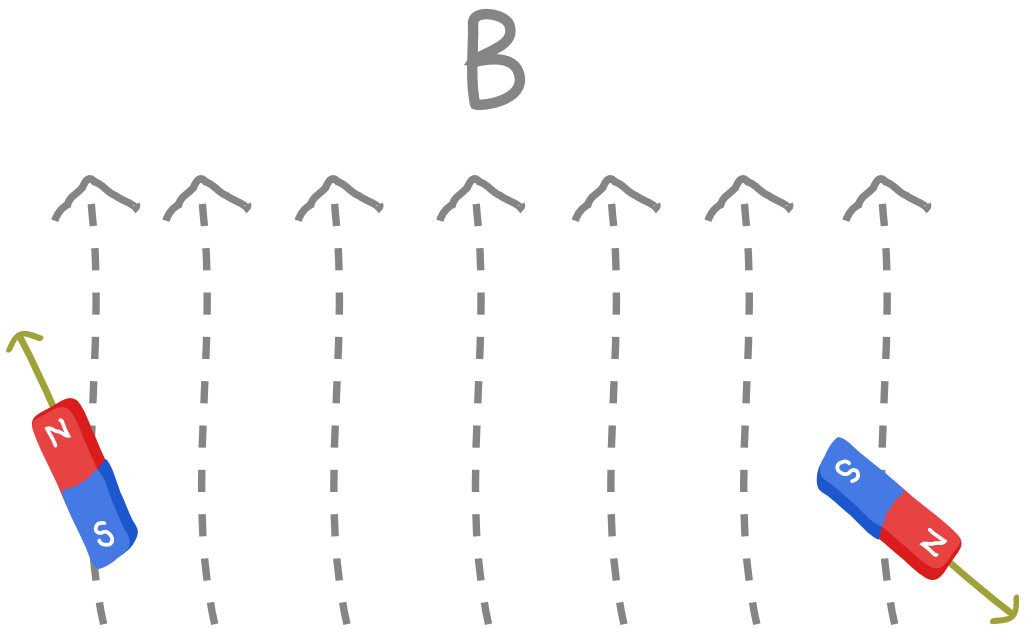
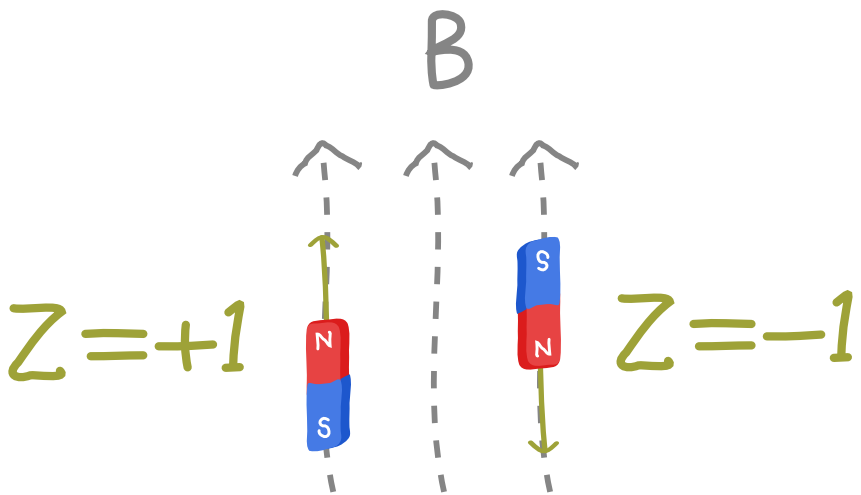


Image credit: Xanadu Quantum Codebook node H.5

This has physical meaning:



Energy is minimized when both magnets point up (same direction as the field).

Image credit: Xanadu Quantum Codebook node H.3

Hamiltonians

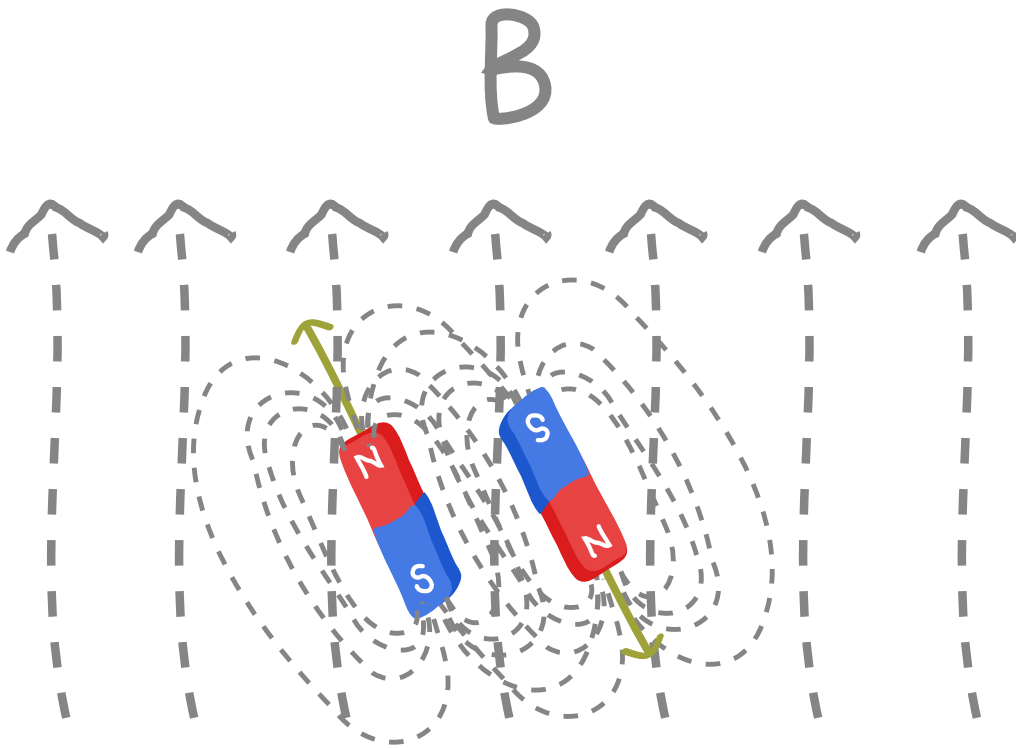


Image credit: Xanadu Quantum Codebook node H.5

Hamiltonians

If we know the Hamiltonian, we can solve the Schrödinger equation to determine how the state of a system evolves with time.

The solution is:

$e^{-i\hat{H}t/\hbar}$ is the *time evolution operator*, and $\hbar$ is a fundamental physical constant which we set to 1 for simplicity.

Hamiltonians

As long as we can write down a Hamiltonian, we can simulate the behaviour of a system evolving over time.

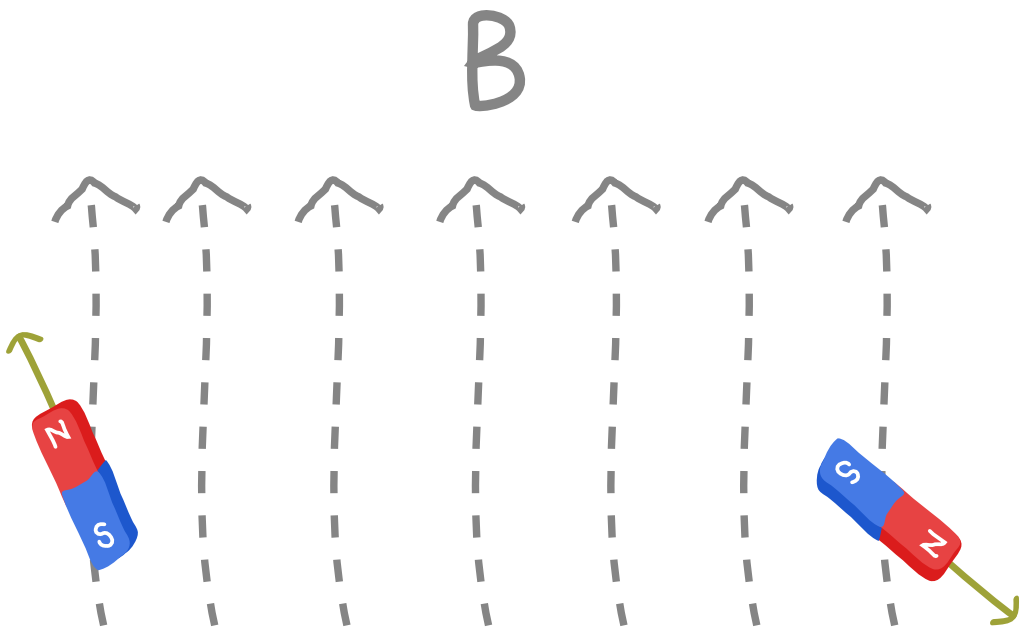
$$|\psi(t)\rangle = e^{-i\hat{H}t}|\psi(0)\rangle$$

We could in principle do this with a classical computer, but even for small problems, the state/Hamiltonian may be exponentially large.

Since $\hat{H}$ is Hermitian, $U = e^{-i\hat{H}t}$ is unitary. If we can prepare the states, and implement U, we can use a quantum computer instead!

Example: non-interacting magnets

How do we represent this system using qubits?

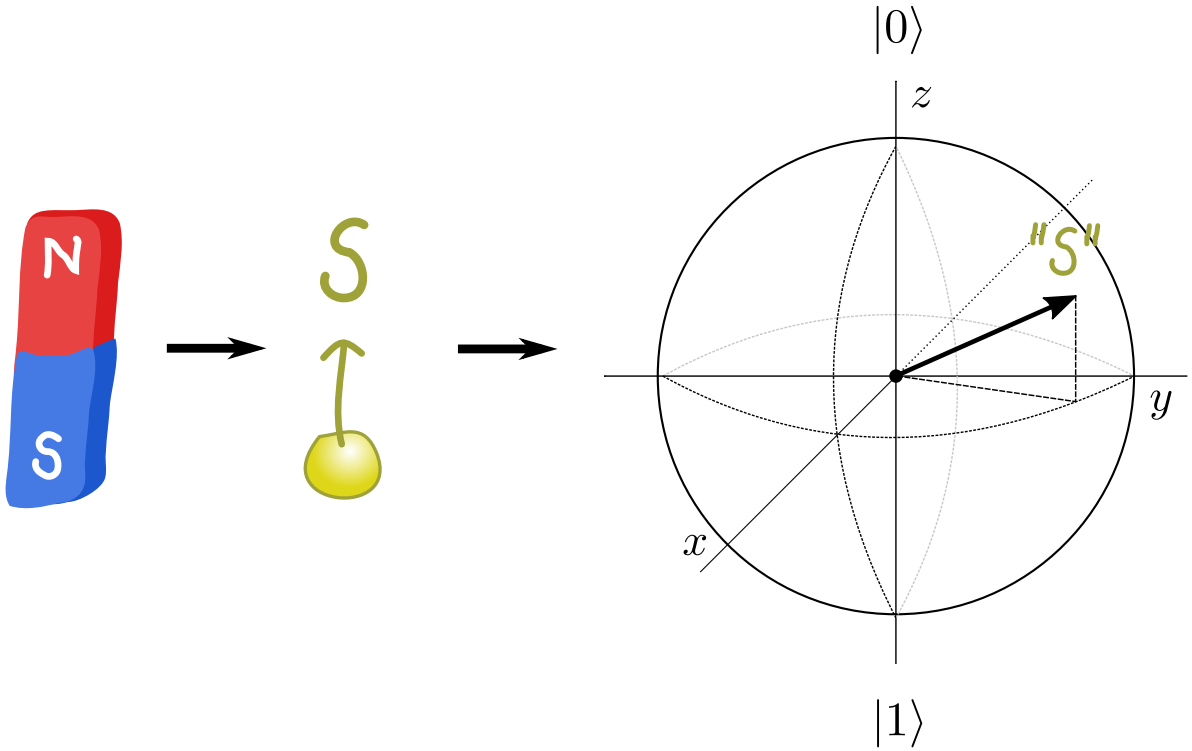


$$\hat{H} = \hat{H}_0 + \hat{H}_1 = -\alpha Z_0 - \alpha Z_1$$

Image credit: Xanadu Quantum Codebook node H.5

Example: non-interacting magnets

Magnets have some orientation represented by a vector in 3-dimensional space. A qubit's Bloch vector can play this role.



More generally: need to perform an *encoding* to translate a Hamiltonian from one domain to domain of qubits.

Image credit: Xanadu Quantum Codebook node H.5

Example: non-interacting magnets

We can describe the qubit system using the same Hamiltonian:

$$\hat{H} = -\alpha Z_0 - \alpha Z_1$$

The energy of the system is minimized when both qubits are in the $|0\rangle$ state ($+1$ eigenvalue of Z).

Example: non-interacting magnets

Consider just one qubit for now.

$$\hat{H} = -\alpha Z_0$$

If a qubit starts in state $|\psi(0)\rangle$ at time $t = 0$, at a later time, we should find it in state

We already know how to simulate this!

Example: non-interacting magnets

Recall how we expressed RZ as a matrix exponential:

We have

So to evolve this system over time, we just need to apply an RZ where the rotation angle is parametrized by time.

Example: non-interacting magnets

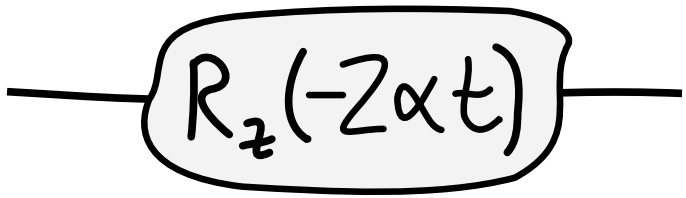
What about two qubits?

$$\hat{H} = -\alpha Z_0 - \alpha Z_1$$

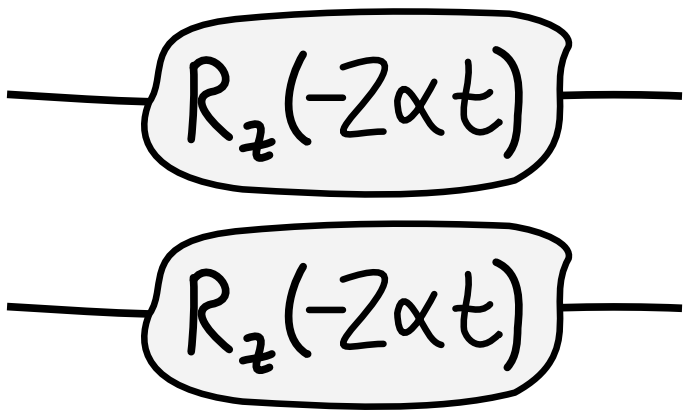
If they don't interact, it would make sense that our gates act on the qubits independently.

Example: non-interacting magnets

As quantum circuits, for one qubit:



Two qubits:



With only single-qubit Pauli Z terms in the Hamiltonian, and non-interacting qubits, Hamiltonian simulation looks pretty easy!

Image credit: Xanadu Quantum Codebook node H.5

Hamiltonian simulation in general

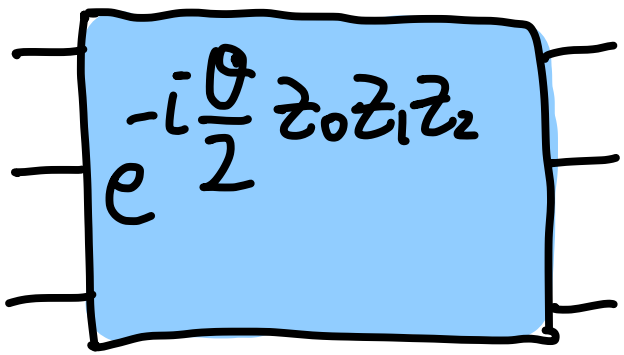
What happens when

1. we have Paulis that are acting on more than one qubit?
2. we have interaction terms?

For the first, we can deal with individual terms exactly. For the second, things get hard.

Circuits for individual Pauli terms

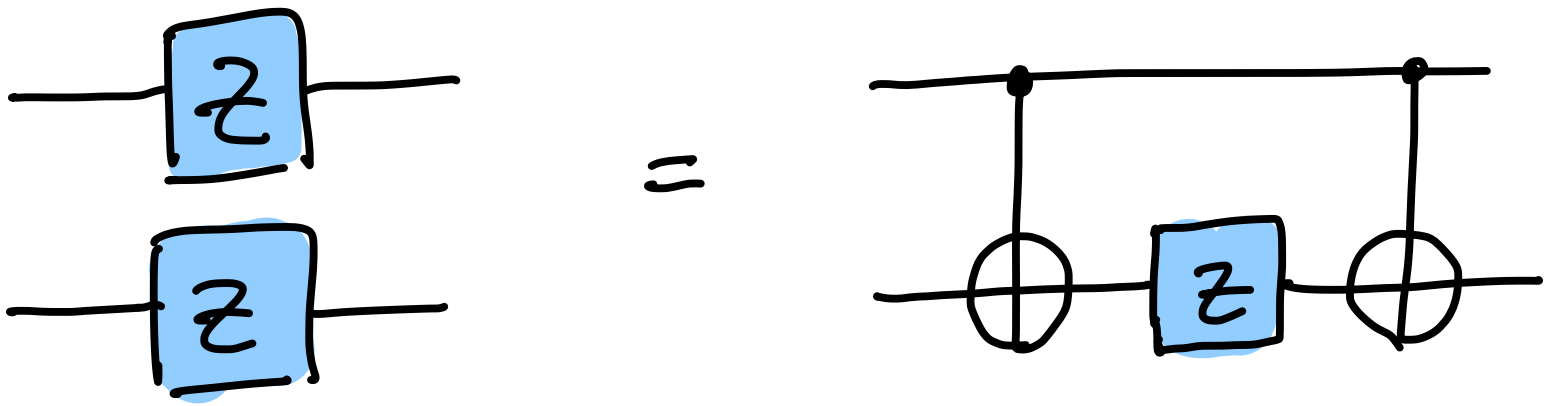
Example: suppose we have the Pauli term $Z_0 Z_1 Z_2$ in the Hamiltonian. How do we implement this as a circuit?



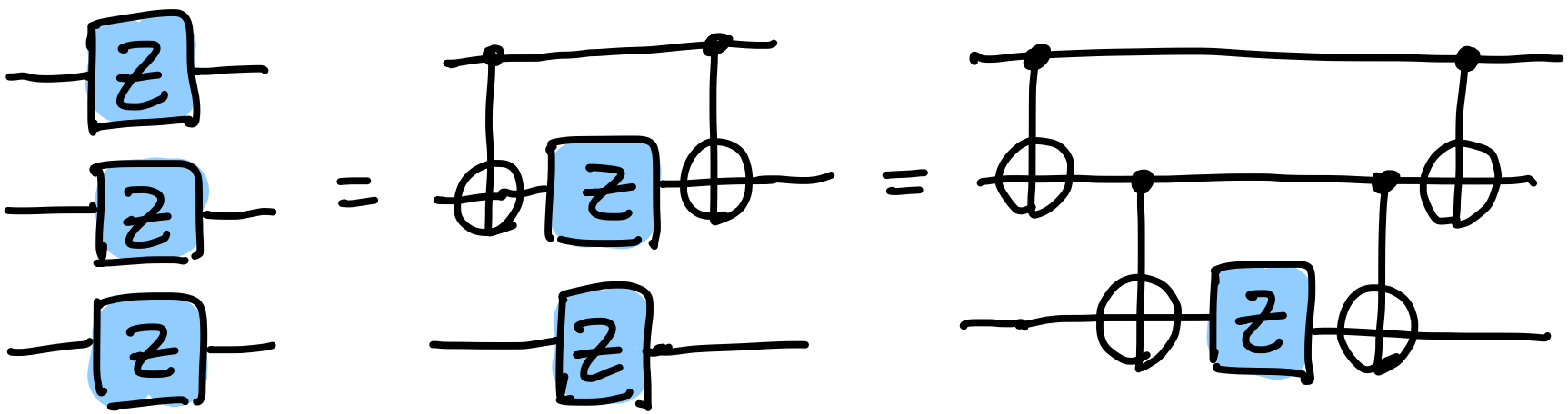
Idea: use some special gates to transform this into an exponential using only single-qubit Pauli Z terms, then implement those.

Circuits for individual Pauli terms

There is a useful circuit identity that relates Z and CNOT:

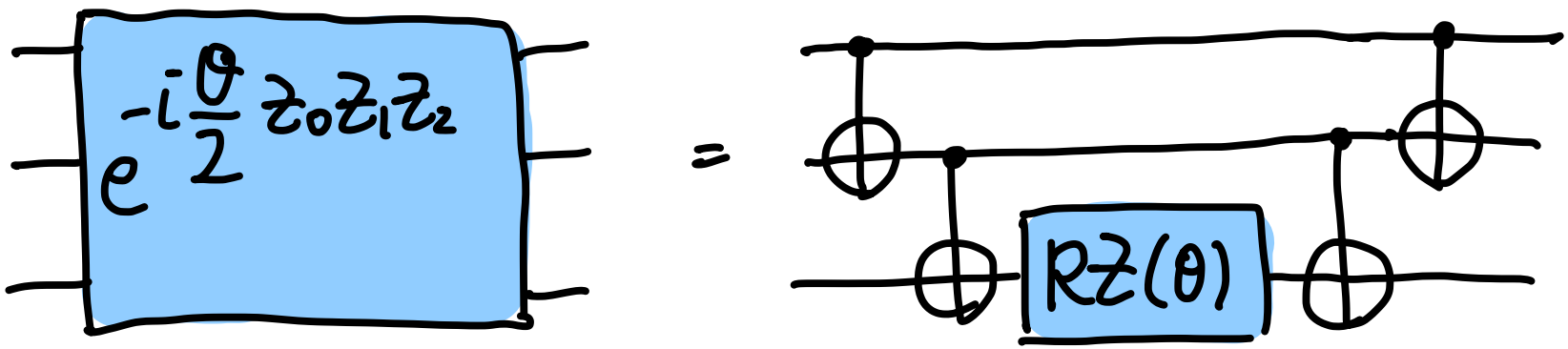


We can apply this multiple times



Circuits for individual Pauli terms

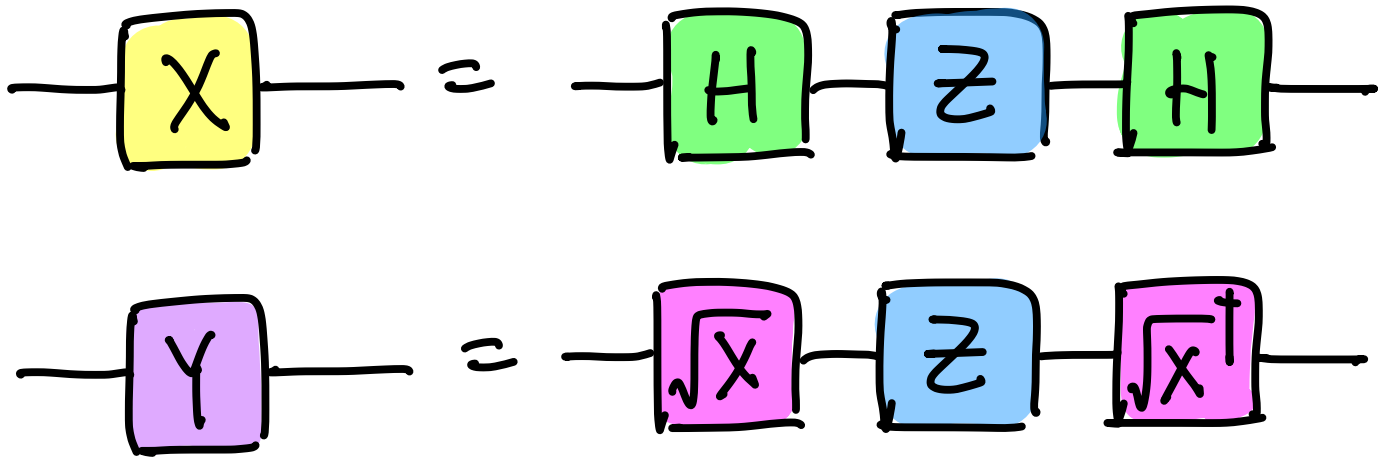
Z is just a special case of RZ though; so this works in general:



We can deal with any product of Pauli Z in this way.

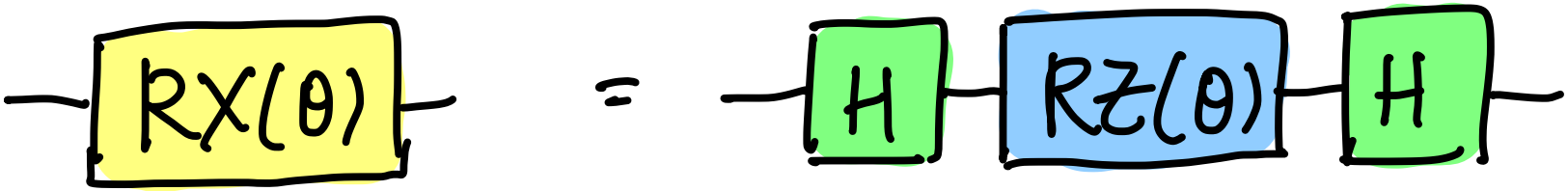
Circuits for individual Pauli terms

X and Y take a little more work; but they are related to Z in special ways:

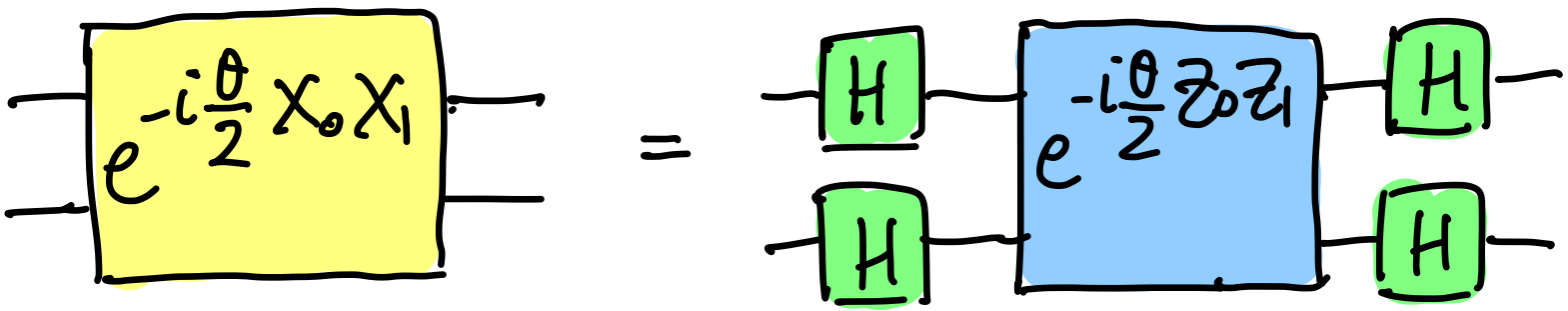


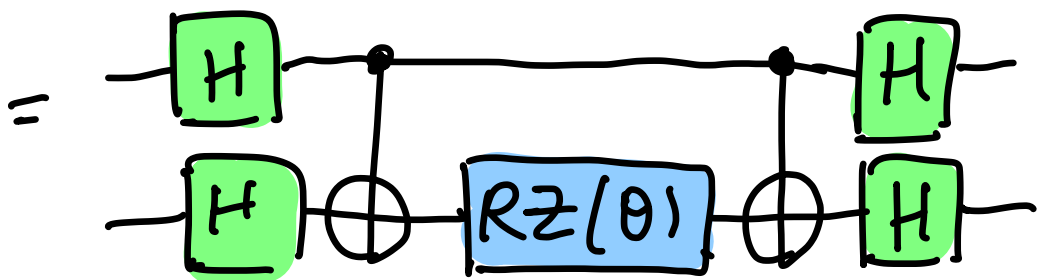
Circuits for individual Pauli terms

RX has the same relationship:



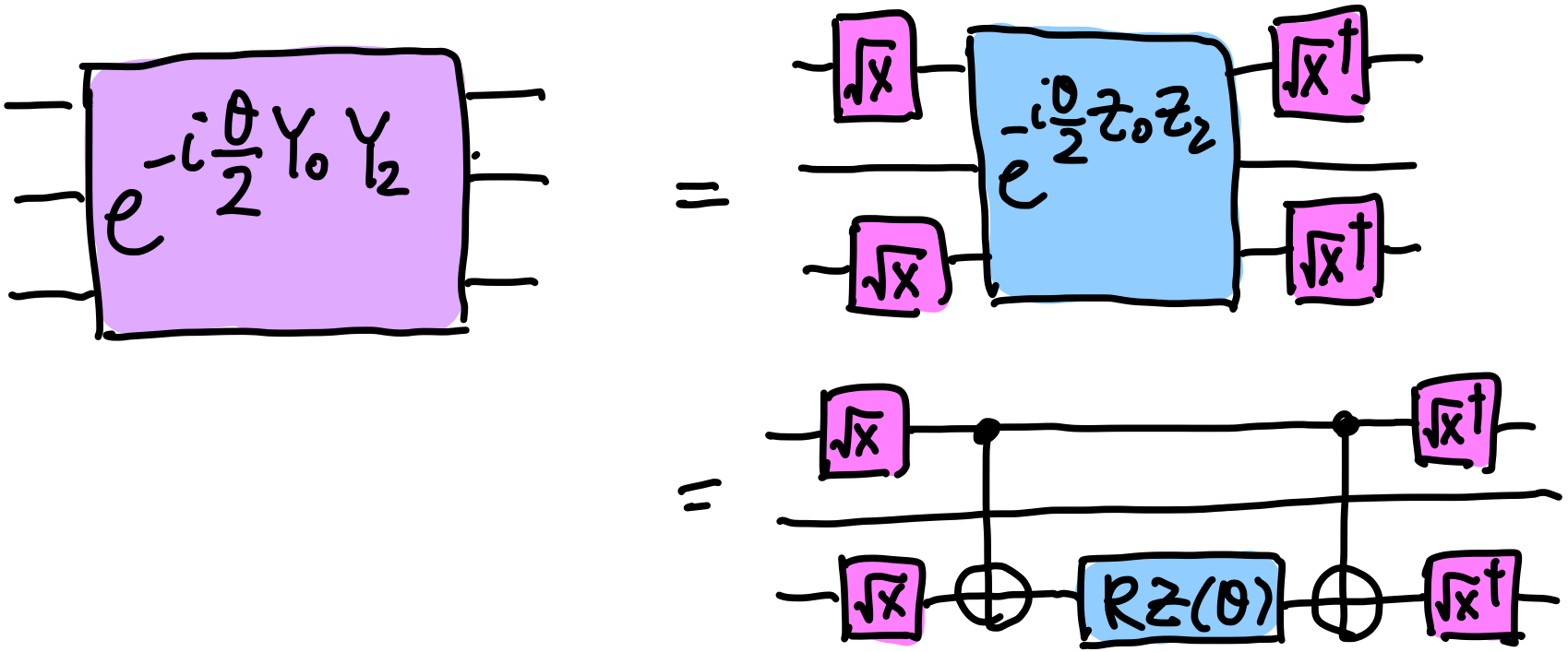
So when we exponentiate...





Circuits for individual Pauli terms

Similarly for Y



We can do this for arbitrary Paulis with X, Y, and Z terms.

Fun fact: the operators applied on either side of the Z are elements of the **Clifford group**. Cliffords send Paulis to Paulis.

Circuits for individual Pauli terms

In PennyLane, there are two special gates to take care of this:

- `qml.PauliRot`
- `qml.MultiRZ`

Let's try a couple of these identities and see how they work and get decomposed.

Dealing with sums of Pauli terms

Now for the hard part. What do we do when we have something like this:

$$e^{-i\alpha Z_0 X_1 - i\beta Z_0 Y_1}$$

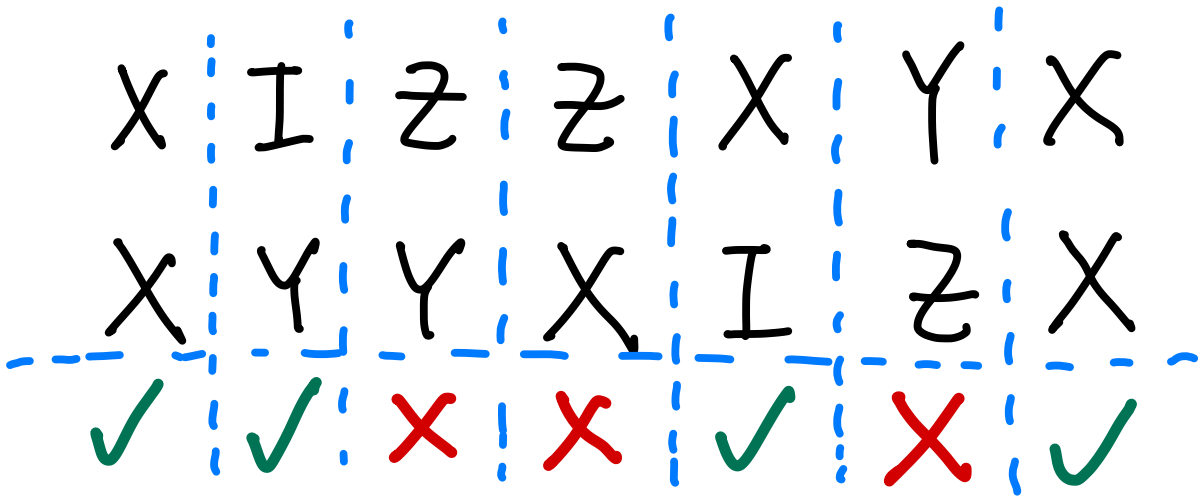
More generally,

$$e^{-i\alpha P - i\beta Q}, \quad P, Q \in \mathcal{P}_n$$

What we do depends on whether P and Q commute, i.e., if $[P, Q] = PQ - QP = 0$.

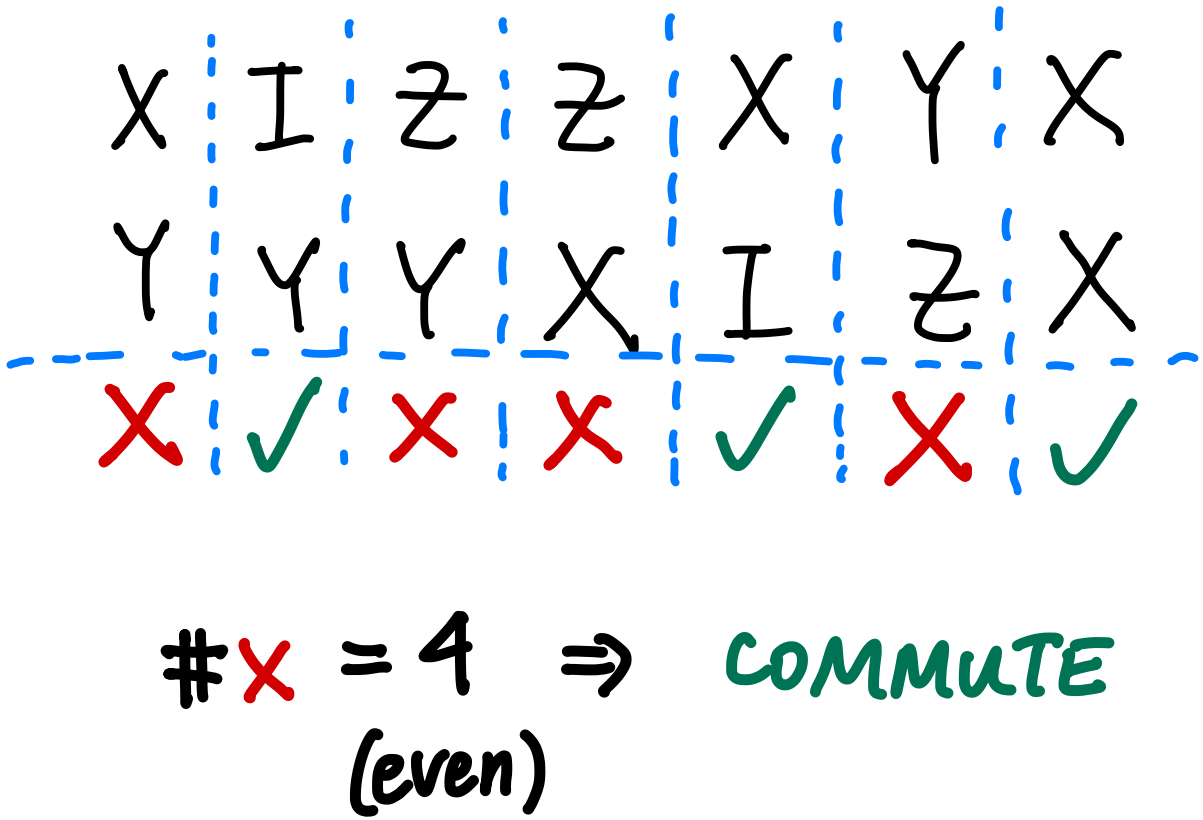
Dealing with sums of Pauli terms

How to tell if two multi-qubit Paulis commute: check number of non-identity qubits on which they differ.



X = 3 (odd) ⇒ DO NOT COMMUTE

Dealing with sums of Pauli terms

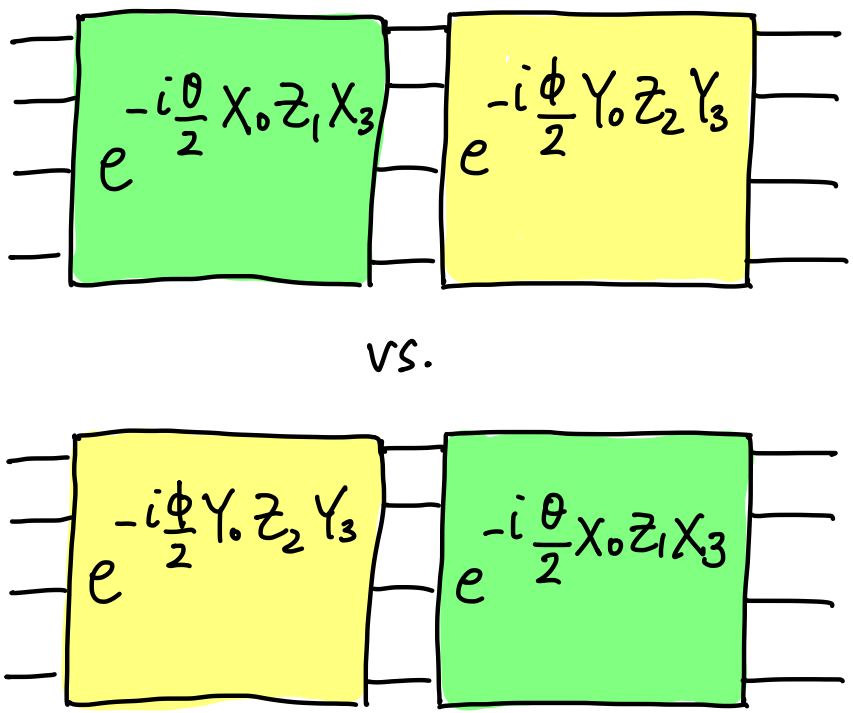


(Under the hood: binary symplectic representation of Pauli terms, and symplectic inner products.)

Dealing with sums of Pauli terms

When two (or more) Paulis commute, the exponential of their linear combination can be factored exactly:

Let's code up an example:



Dealing with sums of Pauli terms: Trotterization

When two (or more) Paulis do not commute, the exponential of their linear combination can be decomposed approximately using the **Trotter-Suzuki decomposition**:

where $O(1/n)$ is an error term that depends on n.

The smaller n is, the better the approximation:

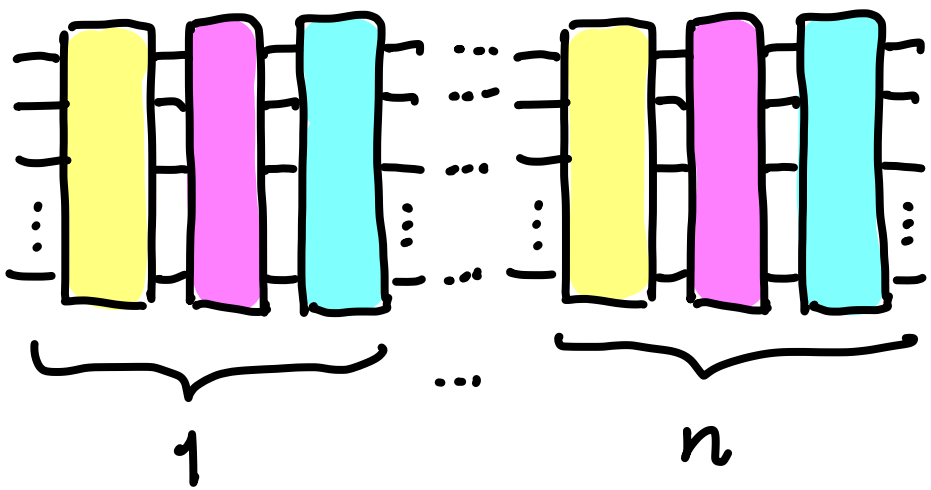
Dealing with sums of Pauli terms: Trotterization

$$\hat{H} = \alpha P_1 + \beta P_2 + \gamma P_3$$

$$\downarrow$$

$$e^{-i\hat{H}} \approx \left(e^{-i\frac{\alpha P_1}{n}} e^{-i\frac{\beta P_2}{n}} e^{-i\frac{\gamma P_3}{n}} \right)^n$$

$$\downarrow$$



Dealing with sums of Pauli terms: Trotterization

In PennyLane, we can use `ApproxTimeEvolution`:

```
def circuit(H):
    # ...
    qml.ApproxTimeEvolution(H, t, n)
    # ...
```

This gate implements exactly the formula from the previous page.

$$e^{-i\alpha P - i\beta Q} \approx \left(e^{-\frac{i\alpha P}{n}} e^{-\frac{i\beta Q}{n}}\right)^n$$

"Higher-order" Trotter formulas also exist, e.g., second order:

$$e^{-i\alpha P - i\beta Q} \approx \left(e^{-\frac{i\alpha P}{2n}} e^{-\frac{i\beta Q}{n}} e^{-\frac{i\alpha P}{2n}}\right)^n$$

which can lead to lower approximation error, at cost of more gates!

Dealing with sums of Pauli terms: other methods

Trotterization and product formulae are not the only way to perform Hamiltonian simulation, but they are the most straightforward to understand, and there are many educational resources.

Other methods include:

- Linear combination of unitaries (see Codebook H.6-H.7)
- Qubitization

All these methods are more "long term" applications of quantum computers as they require huge amounts of qubits and gates to simulate non-trivial Hamiltonians.

Next time

Content:

- You are the teachers

Action items:

1. Assignment 4 (can do all problems)
2. Final project

Recommended reading:

- Codebook Hamiltonian simulation module, nodes H.1-H.5 (optionally H.6-H.7)