

FROM QUANTUM ALGORITHM TO QUANTUM HARDWARE

Part 1: Gate-based quantum computing

18.09.2026 | DigiQ school | JUNIQ

From quantum evolution to quantum logic

Quantum evolution

$$i\hbar \frac{d}{dt} |\psi(t)\rangle = H(t) |\psi(t)\rangle$$

For time-independent H,

$$|\psi(t)\rangle = e^{-iHt/\hbar} |\psi(0)\rangle .$$

Continuous time evolution

$\Longleftrightarrow$

Quantum computation

$$|\psi_0\rangle \xrightarrow{U_1} |\psi_1\rangle \xrightarrow{U_2} \cdots \xrightarrow{U_N} |\psi_N\rangle$$

$U_k \in \{\text{available quantum gates}\}.$

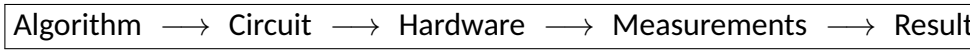
Discrete sequence of operations

The central practical question

How do we turn the desired quantum evolution into a sequence of gates that a real quantum processor can implement?

JÜLICH Forschungszentrum | JUNIQ QUANTUM USER FACILITY

From an algorithm to a measured answer



- The mathematical algorithm is only the starting point.
- Each step introduces constraints, costs, and possible errors.
- Today: what do we need to think about when we actually want to run the algorithm?

Guiding question

What determines whether a quantum algorithm is actually useful on hardware?

QAOA: from an optimization problem to a quantum circuit

Cost Hamiltonian

$$H_C = \sum_j c_j P_j$$

Cost evolution

$$U_C(\gamma) = e^{-i\gamma H_C}$$

Mixing evolution

$$U_M(\beta) = e^{-i\beta H_M}$$

QAOA circuit

$$|+\rangle^{\otimes n} \xrightarrow{U_C(\gamma_1)} \xrightarrow{U_M(\beta_1)} \cdots \xrightarrow{U_C(\gamma_p)} \xrightarrow{U_M(\beta_p)}$$

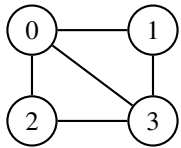
$$|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle = \prod_{k=1}^{p} U_M(\beta_k) U_C(\gamma_k) |+\rangle^{\otimes n}$$

$$\boxed{\min \langle H_C \rangle}$$

But what does $e^{-i\gamma H_C}$ actually mean on hardware?

JÜLICH Forschungszentrum | JUNIQ QUANTUM USER FACILITY

A concrete example: MaxCut



Assign a binary variable to each vertex,

$$z_i \in \{0, 1\}.$$

Goal:
Partition the vertices into two sets such that as many edges as possible cross the partition.

An edge (i, j) contributes

$$C_{ij} = -\frac{1 - Z_i Z_j}{2}.$$

Therefore,

$$\boxed{H_C = -\sum_{(i,j)\in E} \frac{1 - Z_i Z_j}{2}}$$

and maximizing the cut is equivalent to

$$\max \langle H_C \rangle.$$

Already we see the connection to gates

The QAOA cost evolution $e^{-i\gamma H_C}$ consists of ZZ rotations—one for each edge of the graph.

A common language: Pauli Hamiltonians

Many quantum algorithms lead to Hamiltonians of the form

$$H = \sum_j c_j P_j,$$

where P_j are Pauli strings. For example,

$$H = 0.7X_0X_1 - 0.4Z_1Z_2 + 0.2X_0Y_2 + \cdots$$

QAOA, VQE, Hamiltonian simulation, and counterdiabatic constructions can all lead to such terms. The algorithm may require an operation such as

$$U(t) = e^{-itH}.$$

But hardware cannot execute arbitrary e^{-itH} directly.

We need to turn the mathematical expression into an executable circuit.

JÜLICH Forschungszentrum | JUNIQ QUANTUM USER FACILITY

From a Pauli rotation to gates

A basic building block with a Pauli string P is

$$R_P(\theta) = e^{-i\theta P/2}.$$

Conceptually:

$$e^{-i\theta P/2} \quad \longrightarrow \quad \text{basis changes} + \text{entangling gates} + R_Z + \text{uncompute}$$

For example,

$$P = X_0 Z_1 Y_2$$

requires basis changes on the qubits with X and Y, plus an entangling circuit.

Important distinction

The algorithm specifies the operation; the compiler determines how that operation is realized as gates.

One logical operation, many possible circuits

The same logical operation can have very different implementations:

logical circuit $\longrightarrow$ native gate set

Examples of costs we may care about:

- Number of one- and two-qubit gates
- Circuit depth
- Native gate compatibility
- Connectivity / routing overhead

Question

If two circuits implement the same unitary, are they equally good on hardware?

No, the hardware cost can be very different.

JÜLICH Forschungszentrum | JUNIQ QUANTUM USER FACILITY

One logical operation, many possible circuits

The same logical operation can have very different implementations:

$$\text{logical circuit} \longrightarrow \text{native gate set}$$

Examples of costs we may care about:

- Number of one- and two-qubit gates
- Circuit depth
- Native gate compatibility
- Connectivity / routing overhead

Question

If two circuits implement the same unitary, are they equally good on hardware?

No, the hardware cost can be very different.

Different hardware $\Rightarrow$ different native gates

Platform	Physical qubit	Typical 2Q operations
Superconducting	Transmon	CZ, CR-derived gates
Trapped ion	Atomic ion	XX / MS-type gates
Neutral atom	Atomic states	Rydberg-mediated gates
Photonic	Optical modes	Interference / measurement

Abstract circuit $\longrightarrow$ native gates of a specific device

The gate set is part of the hardware architecture.

The “same” logical operation can have very different implementation costs on different platforms.

Where does Trotterization enter?

Suppose

$$H = A + B.$$

If A and B commute,

$$e^{-it(A+B)} = e^{-itA}e^{-itB}.$$

But if

$$[A, B] \neq 0,$$

then in general

$$e^{-it(A+B)} \neq e^{-itA}e^{-itB}.$$

Need Trotter approximations!

Key point

Non-commutativity produces an **algorithmic approximation error**.

JÜLICH Forschungszentrum | JUNIQ QUANTUM USER FACILITY

Where does Trotterization enter?

Suppose

$$H = A + B.$$

If A and B commute,

$$e^{-it(A+B)} = e^{-itA}e^{-itB}.$$

But if

$$[A, B] \neq 0,$$

then in general

$$e^{-it(A+B)} \neq e^{-itA}e^{-itB}.$$

Need Trotter approximations!

Key point

Non-commutativity produces an **algorithmic approximation error**.

JÜLICH Forschungszentrum | JUNIQ QUANTUM USER FACILITY

Trotter steps: accuracy versus circuit cost

Increasing the number of Trotter steps r gives a finer approximation:

$$r \uparrow \quad \Rightarrow \quad \text{Trotter error} \downarrow$$

But each step requires gates:

$$r \uparrow \quad \Rightarrow \quad \text{gate count / depth} \uparrow$$

Therefore,

$$\boxed{\text{better approximation} \not\Rightarrow \text{better hardware result}}$$

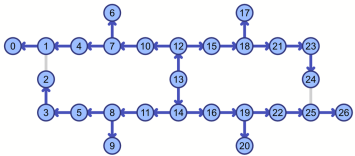
On real hardware, a deeper circuit also has more opportunities for decoherence and gate errors.

$$\boxed{\text{Trotter error} \quad \leftrightarrow \quad \text{hardware error}}$$

JÜLICH Forschungszentrum | JUNIQ QUANTUM USER FACILITY

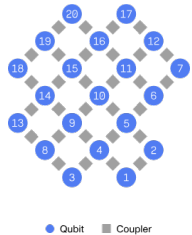
Hardware connectivity adds another cost

Superconducting



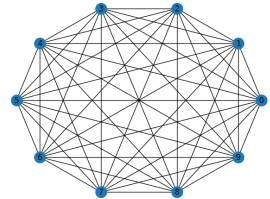
Sparse local connectivity
$\Rightarrow$ routing overhead

Superconducting



Square-lattice connectivity

Trapped ions



All-to-all connectivity

Why does this matter?

A logical interaction between two qubits may require additional gates on one architecture but not on another.

JÜLICH Forschungszentrum | JUNIQ QUANTUM USER FACILITY

And then there is hardware noise

The compiled circuit is still an ideal description.

In reality: $U_{\text{hardware}} \neq U_{\text{ideal}}$.

- Gate and readout errors
- Decoherence
- Crosstalk
- Calibration drift

And then there is hardware noise

The compiled circuit is still an ideal description.

In reality: $U_{\text{hardware}} \neq U_{\text{ideal}}$.

- Gate and readout errors
- Decoherence
- Crosstalk
- Calibration drift

Trotter steps	$\rightarrow$	algorithmic error
$\Downarrow$		
gates / depth	$\rightarrow$	hardware exposure
$\Downarrow$		
connectivity	$\rightarrow$	routing overhead

What will we investigate in the notebook?

1. **Pauli rotations**
   How does a high-level Hamiltonian term become gates?
2. **Trotterization**
   How does the number/order of terms affect accuracy and circuit cost?
3. **Hardware constraints**
   What happens when the gate set and connectivity are restricted?
4. **Noise**
   When does reducing Trotter error stop improving the hardware result?

The central question: gate-based notebook

If we can make the circuit in theory more accurate with respect to the algorithm,
should we always do so on the imperfect hardware?

Accuracy	↔	Circuit cost	↔	Hardware noise	↔	Measurement cost

The goal is not necessarily the most accurate circuit.

The goal is a useful answer at an acceptable total cost.

Not covered: measurement cost

When running a quantum job we extract information at the end via measurements. For finite shots,

$$\text{statistical uncertainty} \sim \frac{1}{\sqrt{N_{\text{shots}}}}.$$

Thus,

$$N_{\text{shots}} \uparrow \quad \Rightarrow \quad \text{statistical error} \downarrow$$

but also

$$N_{\text{shots}} \uparrow \quad \Rightarrow \quad \text{experimental cost} \uparrow .$$

measurement settings.

We also might need more information than just the bistrings.

JÜLICH Forschungszentrum | JUNIQ QUANTUM USER FACILITY

Outlook: from physical to logical qubits

Today we are thinking about

$$\text{algorithm} \rightarrow \text{physical circuit} \rightarrow \text{noisy measurements.}$$

With fault-tolerant quantum computing:

$$\text{algorithm} \rightarrow \text{logical circuit} \rightarrow \text{error correction} \rightarrow \text{physical implementation.}$$

The optimization problem does not disappear, it changes:

- logical qubit overhead, physical qubits per logical qubit
- logical gate cost
- non-Clifford resources
- error-correction overhead

The deeper lesson

There is no hardware-independent notion of the cost of a quantum algorithm.