Lecture 6: Advanced Unitary Synthesis and Simulation

Decomposition Strategies, Universality, and Trotterization

Moody T. Chu

North Carolina State University

Spring 2026

Lecture Overview

1. Decomposition of Unitaries
2. Constructing Controlled-U Gates
3. Multi-Controlled Operations
4. Measurement Principles
5. Universality Mechanisms
6. Quantum Simulation

Why This Lecture Matters

- So far, our circuits have used a handful of named gates: X, Y, Z, H, S, T, CNOT, Toffoli.
- Real algorithms (Shor's, Grover's, and beyond) require *arbitrary* unitaries — rotations by unusual angles, multi-qubit controlled operations, and simulated physical evolutions.
- This lecture supplies the **practical toolkit** for turning any such unitary into a circuit built from gates real hardware actually offers:
  - How to synthesize *any* single-qubit gate (Z-Y decomposition).
  - How to build a controlled version of *any* gate (ABC construction).
  - How to control on *many* qubits efficiently (ancilla ladders).
  - How to implement gates between qubits that aren't directly coupled (Gray codes).
  - How to simulate the time evolution of a physical system (Trotterization).

Motivation: 3-D Rigid Body Motion

- A rigid body free to rotate in 3D space has exactly three degrees of freedom.
  - ◇ It can yaw (turn left/right), pitch (tilt up/down), and roll (pivot side-to-side).
- **Euler Angles Theorem:** any orientation of a rigid body can be achieved by three rotations about two orthogonal axes.
  - ◇ A sequence such as Z–X–Z can map out every possible 3D orientation.
  - ◇ The third axis is unnecessary — two suffice.
- $SO(3)$ is the group of standard 3D rotations.
- **Why this matters here:** recall from Lecture 4 that every single-qubit unitary is a rotation $R_{\hat{n}}(\theta)$ of the Bloch sphere. The same Euler-angle logic applies: any point on the Bloch sphere can be reached by composing rotations about just *two* fixed axes. We make this precise next, using Z and Y.

The Z-Y Decomposition Theorem

- **Why we need this:** quantum hardware typically offers only a small set of *native* rotations (often R_z and R_y, or similar). This theorem guarantees those two types alone are enough to reach *any* single-qubit unitary — giving us a universal synthesis recipe.
- **Theorem:** for any single-qubit unitary $U \in U(2)$, there exist real numbers $\alpha, \beta, \gamma, \delta$ such that:

$$U = e^{i\alpha} R_z(\beta)\, R_y(\gamma)\, R_z(\delta). \tag{1}$$

  - ◇ Geometrically: rotate through the sequence Z–Y–Z on the Bloch sphere to attain U.
  - ◇ The $e^{i\alpha}$ term is a global phase, unobservable on its own, but it will matter once we build *controlled* versions of U (Section 2).
- The next two slides derive $\alpha, \beta, \gamma, \delta$ explicitly from the entries of U — a constructive recipe, not just an existence proof.

Construction: Finding α and γ

- **Step 1, find α:** since $U^\dagger U = I$, taking determinants gives $|\det(U)|^2 = 1$, so $\det(U) = e^{i\theta}$ for some real θ.
  - ◇ Set $\alpha = \theta/2$ and write $U = e^{i\alpha} V$. Then $\det(V) = e^{i\theta} e^{-2i\alpha} = 1$, so $V \in SU(2)$.
- **Step 2, parametrize V:** since $V \in SU(2)$, its inverse equals its adjoint and it takes the form

$$V = \begin{pmatrix} a & -\bar{b} \\ b & \bar{a} \end{pmatrix}, \qquad |a|^2 + |b|^2 = 1.$$

- **Step 3, find γ:** write a, b in polar form (always possible since $|a|^2 + |b|^2 = 1$),

$$a = \cos\left(\frac{\gamma}{2}\right) e^{i\phi_a}, \qquad b = \sin\left(\frac{\gamma}{2}\right) e^{i\phi_b},$$

giving:

$$V = \begin{pmatrix} \cos\left(\frac{\gamma}{2}\right) e^{i\phi_a} & -\sin\left(\frac{\gamma}{2}\right) e^{-i\phi_b} \\ \sin\left(\frac{\gamma}{2}\right) e^{i\phi_b} & \cos\left(\frac{\gamma}{2}\right) e^{-i\phi_a} \end{pmatrix}.$$

Construction: Finding β and δ

- Recall from Lecture 4 that $R_z(\phi) = \begin{pmatrix} e^{-i\phi/2} & 0 \\ 0 & e^{i\phi/2} \end{pmatrix}$ and $R_y(\gamma) = \begin{pmatrix} \cos(\gamma/2) & -\sin(\gamma/2) \\ \sin(\gamma/2) & \cos(\gamma/2) \end{pmatrix}$. Multiplying out $R_z(\beta)R_y(\gamma)R_z(\delta)$ directly gives:

$$R_z(\beta)R_y(\gamma)R_z(\delta) = \begin{pmatrix} \cos\left(\frac{\gamma}{2}\right) e^{-i(\beta+\delta)/2} & -\sin\left(\frac{\gamma}{2}\right) e^{-i(\beta-\delta)/2} \\ \sin\left(\frac{\gamma}{2}\right) e^{i(\beta-\delta)/2} & \cos\left(\frac{\gamma}{2}\right) e^{i(\beta+\delta)/2} \end{pmatrix}$$

- Comparing this with V from the previous slide, the phase arguments must match:

$$-\frac{\beta+\delta}{2} = \phi_a \implies \beta+\delta = -2\phi_a, \qquad \frac{\beta-\delta}{2} = \phi_b \implies \beta-\delta = 2\phi_b.$$

- Solving this linear system always yields real angles:

$$\beta = \phi_b - \phi_a, \qquad \delta = -\phi_a - \phi_b.$$

$$\implies U = e^{i\alpha} R_z(\beta)R_y(\gamma)R_z(\delta) \text{ is proven for any unitary } U.$$

The Full Synthesis Recipe

Given *any* single-qubit unitary $U = \begin{pmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \end{pmatrix}$, here is the complete step-by-step recipe to find its Z-Y-Z circuit:

1. Compute $\theta = \arg\det(U)$, set $\alpha = \theta/2$, and form $V = e^{-i\alpha}U$.
2. Read off $a = V_{11}$, $b = V_{21}$ from V.
3. Set $\gamma = 2\arccos|a|$, $\phi_a = \arg(a)$, $\phi_b = \arg(b)$.
4. Set $\beta = \phi_b - \phi_a$ and $\delta = -\phi_a - \phi_b$.

Worked example: apply this to the Hadamard gate, $H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$.

- $\det(H) = -1 = e^{i\pi}$, so $\alpha = \pi/2$ and $V = e^{-i\pi/2}H = -iH$.
- $a = V_{11} = -i/\sqrt{2}$, $b = V_{21} = -i/\sqrt{2}$, so $|a| = |b| = 1/\sqrt{2} \Rightarrow \gamma = 2\arccos\left(1/\sqrt{2}\right) = \pi/2$.
- $\phi_a = \phi_b = -\pi/2 \Rightarrow \beta = 0$, $\delta = \pi$.
- **Result:** $H = e^{i\pi/2}\, R_z(0)\, R_y(\pi/2)\, R_z(\pi) = i\, R_y(\pi/2) R_z(\pi)$ — only two physical rotations needed (since $\beta = 0$, $R_z(\beta)$ is the identity).

Why We Need Controlled-U

- We already know how to build Controlled-NOT (Lecture 4). But algorithms like Shor's and the Quantum Fourier Transform (next lecture) require **controlled versions of arbitrary single-qubit gates** U — controlled phase rotations, controlled Hadamards, and more.
- Given the Z-Y-Z decomposition $U = e^{i\alpha} R_z(\beta) R_y(\gamma) R_z(\delta)$ from the previous slides, we now show how to build a Controlled-U gate using only CNOTs and single-qubit gates.
- **The idea:** split $R_z(\beta) R_y(\gamma) R_z(\delta)$ into three pieces A, B, C such that $ABC = I$, but inserting Pauli-X gates between them (triggered by the control qubit) recovers U.

ABC Construction: The Recipe

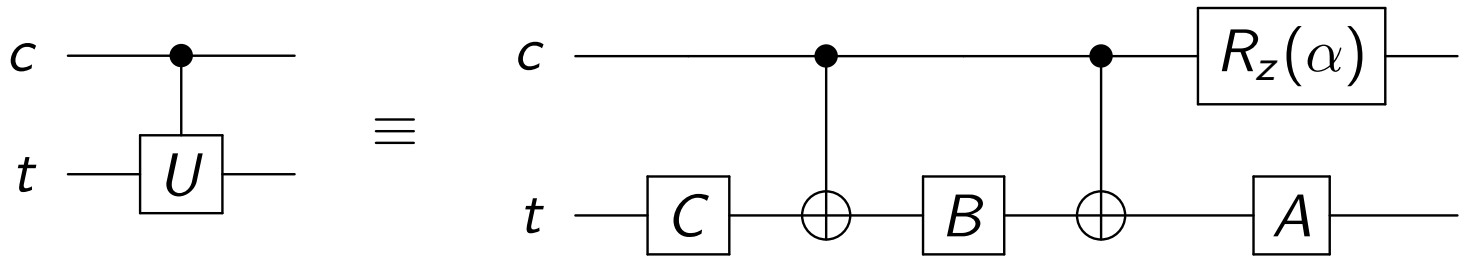
- Define:

$$\begin{aligned} A &:= R_z(\beta)\, R_y(\gamma/2), \\ B &:= R_y(-\gamma/2)\, R_z(-(\delta+\beta)/2), \\ C &:= R_z((\delta-\beta)/2), \end{aligned}$$

  Then:
  - $ABC = I$ (this handles the $|0\rangle$ branch of the control).
  - $e^{i\alpha}\, AXBXC = U$ (this handles the $|1\rangle$ branch).
- Both claims are verified explicitly on the next two slides.

ABC Construction: The Logic

- We create a structure where the Pauli-X gates act as "switches":
  - ◇ If the control is $|0\rangle$, nothing happens ($ABC = I$).
  - ◇ If the control is $|1\rangle$, U is applied to the target.
- The following two circuits are equivalent:



- **Reading note:** the circuit applies C first, then B, then A (left to right) — but as *matrices* these compose right to left, i.e., the overall target operation is $A(\cdot)B(\cdot)C$ applied to the incoming state, matching the order ABC used in the recipe above.
- The structure on the right is the universal template for any controlled single-qubit operation.

ABC Construction: Verifying the Identity Branch

- If the control qubit is $|0\rangle$, the CNOTs do not fire, so the target sees ABC directly. Expanding:

$$ABC = R_z(\beta)\, R_y(\gamma/2)\, R_y(-\gamma/2)\, R_z(-(\delta+\beta)/2)\, R_z((\delta-\beta)/2)$$

- The R_y terms cancel exactly (since $R_y(\gamma/2)R_y(-\gamma/2) = I$). Summing the remaining R_z angles:

$$\beta + \left(\frac{-\delta-\beta+\delta-\beta}{2}\right) = \beta - \beta = 0$$

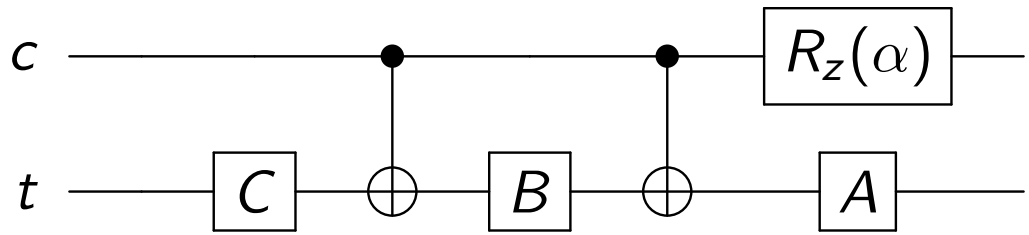
- Thus, $ABC = R_z(0) = I$. The target qubit remains unchanged, as required.

ABC Construction: Verifying the Unitary Branch

- If the control is $|1\rangle$, the X gates fire. The circuit performs $A(XBX)C$.
  - The identities $XR_y(\theta)X = R_y(-\theta)$ and $XR_z(\phi)X = R_z(-\phi)$ hold (conjugation by X flips the sign of a rotation angle, since X reverses the Y and Z axes of the Bloch sphere).
  - Thus,
$$XBX = R_y(\gamma/2)\, R_z((\delta+\beta)/2).$$
- Substituting A and C back in:
$$[R_z(\beta)R_y(\gamma/2)]\,[R_y(\gamma/2)R_z((\delta+\beta)/2)]\,[R_z((\delta-\beta)/2)]$$
and combining adjacent rotations of the same axis:
$$R_z(\beta)\, R_y(\gamma)\, R_z(\delta) = e^{-i\alpha}U$$
- This yields exactly the desired unitary U, up to the phase factor $e^{i\alpha}$ — handled next.

ABC Construction: Phase Compensation

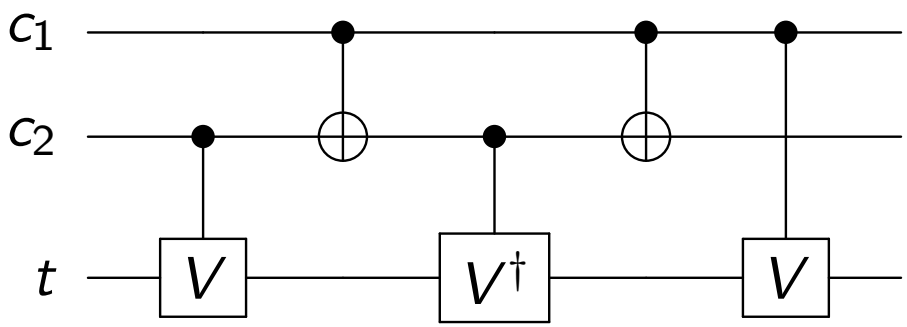
- The two branches disagree by a phase: $ABC = I$, while $e^{i\alpha}AXBXC = U$. Fix this with an *extra* phase rotation on the **control** qubit.



- **Control** $= |0\rangle$**:** CNOTs idle; target applies $ABC = I$; $R_z(\alpha)$ gives phase $e^{-i\alpha/2}$: $\quad |0\rangle\,|t\rangle \to e^{-i\alpha/2}\,|0\rangle\,|t\rangle\,.$
- **Control** $= |1\rangle$**:** CNOTs fire; target applies $AXBXC = e^{-i\alpha}U$; $R_z(\alpha)$ gives phase $e^{i\alpha/2}$: $\quad |1\rangle\,|t\rangle \to e^{-i\alpha/2}\,|1\rangle\;U\,|t\rangle\,.$
- Both branches carry the *same* factor $e^{-i\alpha/2}$ — an unobservable global phase. The circuit implements Controlled-U exactly, up to this irrelevant phase.

The Square-Root Construction for $C^2(U)$

- A $C^2(U)$ gate (a Toffoli-like gate controlled on two qubits) can be built from controlled-V and CNOT gates, where V is any unitary satisfying $V^2 = U$.
  - ◇ **Such a V always exists and is unitary:** write $U = \sum_k \lambda_k |e_k\rangle \langle e_k|$ (spectral decomposition, Lecture 2); since U is unitary, each $|\lambda_k| = 1$. Set $V := \sum_k \sqrt{\lambda_k} |e_k\rangle \langle e_k|$ (principal square root). Then $|\sqrt{\lambda_k}| = 1$ too, so V is unitary, and $V^2 = U$ by construction.
- Consider the following circuit:



  - ◇ If only $c_1 = 1$: V and $V^\dagger$ cancel, target untouched.
  - ◇ If only $c_2 = 1$: the final V gate does not fire, target untouched.
  - ◇ Only if $c_1 = c_2 = 1$: the sequence applies $V \cdot V = U$.

Ancilla Qubits and Linear Scaling

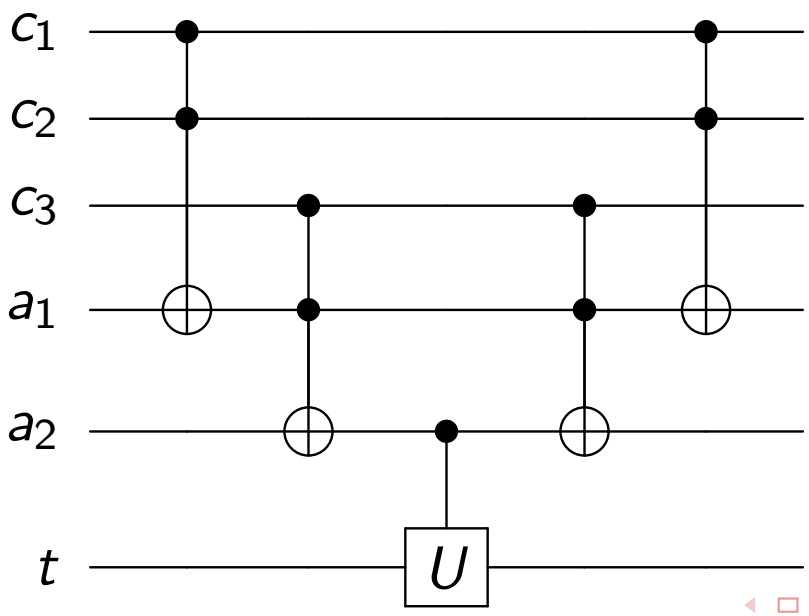
- For n controls ($C^n(U)$), naively nesting the $C^2(U)$ construction above (each control pair needs its own $\sqrt{\cdot}$ decomposition) doubles the gate count at every recursive level, so the gate count scales as $O(2^n)$ — intractable for large n.
- We can achieve *linear scaling* $O(n)$ instead, by introducing *ancilla* (work) qubits.
- An "ancilla ladder" uses Toffoli gates to compute the logical AND of pairs of control bits, storing the intermediate result in the next ancilla.
- The final ancilla serves as the single control for the target gate U — turning an n-control problem into a sequence of ordinary, single-control Toffoli/$C(U)$ gates.

Ancilla Ladder Case Study: $C^3(U)$

- To execute $C^3(U)$ on target $|t\rangle$ given controls $|c_1, c_2, c_3\rangle$, we want the state condition:

$$|111\rangle\,|t\rangle\,|00\rangle \longrightarrow |111\rangle\,U\,|t\rangle\,|00\rangle$$

- This requires $n-1=2$ clean workspace ancillas, $|a_1, a_2\rangle$, and the ladder is run forward (compute), then backward (uncompute), exactly addressing the garbage problem above.



Mathematical Trace of the Ladder Steps

- Toffoli on c_1, c_2 stores the first evaluation on a_1:

$$|c_1, c_2, c_3\rangle\,|t\rangle\,|0,0\rangle \xrightarrow{\text{Toffoli}_1} |c_1, c_2, c_3\rangle\,|t\rangle\,|c_1 \wedge c_2,\ 0\rangle$$

- Toffoli on a_1, c_3 captures the full 3-way AND on a_2:

$$\xrightarrow{\text{Toffoli}_2} |c_1, c_2, c_3\rangle\,|t\rangle\,|c_1 \wedge c_2,\ c_1 \wedge c_2 \wedge c_3\rangle$$

- Use a_2 to trigger $C(U)$:

$$\xrightarrow{C(U)} |c_1, c_2, c_3\rangle\,\left[U^{c_1 \wedge c_2 \wedge c_3}\,|t\rangle\right]\,|c_1 \wedge c_2,\ c_1 \wedge c_2 \wedge c_3\rangle$$

- **Uncompute:** reverse Toffoli_2 then Toffoli_1 (in that order) to reset $|a_1, a_2\rangle \to |0,0\rangle$, exactly undoing the garbage-creation discussed two slides ago:

$$\xrightarrow{\text{Toffoli}_2^\dagger,\ \text{Toffoli}_1^\dagger} |c_1, c_2, c_3\rangle\,\left[U^{c_1 \wedge c_2 \wedge c_3}\,|t\rangle\right]\,|0,0\rangle$$

Why Garbage Ancillas Are a Problem

- The ancilla ladder above computes intermediate AND values *into* the ancilla qubits — but if we simply leave them there after using the final ancilla to control U, they remain entangled with the control qubits as "garbage."
- **Concretely:** suppose the controls start in a superposition, e.g. $|\psi_0\rangle = \frac{1}{\sqrt{2}}(|110\rangle + |111\rangle)\,|t\rangle\,|00\rangle$. If we apply only the *forward* half of the ladder and the controlled-U, but never reverse it, the ancillas end up correlated with the controls:

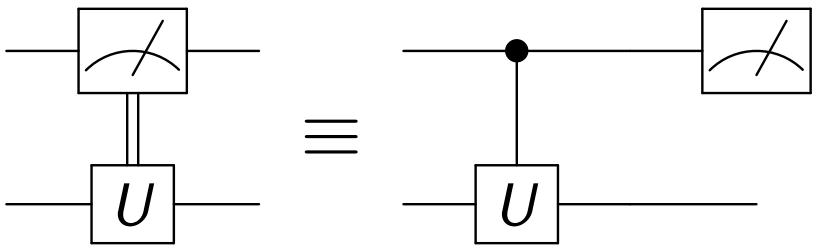
$$|\psi_{\text{err}}\rangle = \frac{1}{\sqrt{2}}\,|110\rangle\,|t\rangle\,|10\rangle + \frac{1}{\sqrt{2}}\,|111\rangle\,U\,|t\rangle\,|11\rangle\,.$$

The Fix: Uncompute the Ladder

- Since $\langle 10|11\rangle = 0$, the two control branches from the previous slide are now **distinguishable** via the ancilla register alone — which destroys any interference between them in later stages of an algorithm relying on this superposition.
- **The fix:** run the AND-computation steps of the ladder again, in *reverse*, after the controlled-U fires — restoring the ancillas to $|0\rangle$ without disturbing the controls or the target.
- This is exactly the same **uncomputation** idea from Lecture 5's Bennett's trick. The circuit on the next slide already builds this in.

The Principle of Deferred Measurement

- *Statement:* measurements can always be moved from an intermediate stage of a quantum circuit to the end of the circuit.
  - ◇ If a measurement result is used to classically control a later gate, that classical control wire can be replaced by a *quantum control* (e.g., a CNOT) acting directly on the not-yet-measured qubit.
- This is vital because it lets us treat an entire algorithm as a single *unitary* evolution up until the final readout — exactly the assumption used throughout this lecture (and needed to apply the deferred-uncomputation arguments of the previous section).



Principle of Implicit Measurement

- Qubits that are never measured at the end of a circuit can be *assumed* to be measured (and the result discarded) without affecting the statistics of the qubits that *are* read out.
  - ◇ This is mathematically equivalent to *tracing out* the unmeasured qubits to obtain the reduced density matrix of the system (Lecture 3) — the partial trace and a discarded measurement give the same reduced state on the remaining qubits.
- Together with deferred measurement, this confirms that the "collapse" of the wavefunction can always be modeled as occurring only at the very end, when an observer explicitly requests a classical readout — a convenient and fully general simplification for circuit design.

Two-Level Unitaries (Generalized Givens Rotation)

- A $d \times d$ unitary is a *two-level unitary* if it acts as the identity on all but two basis components.
  - ◇ In real vector spaces, this is precisely the Givens rotation $G(i, j, \theta)$.
  - ◇ In quantum (complex) spaces, the Givens rotation needs *two* angles instead of one.
- *Decomposition strategy:*
  - ◇ Use a sequence of two-level unitaries to zero out the off-diagonal elements of U, one by one, similar to Gaussian elimination or QR decomposition.
  - ◇ Any U can be decomposed into a product of two-level unitaries.
- For an n-qubit system ($d = 2^n$): a $d \times d$ unitary has $\binom{d}{2} = \frac{d(d-1)}{2}$ off-diagonal entries below the diagonal to zero out, one two-level unitary per entry, giving at most $\frac{d(d-1)}{2} = 2^{n-1}(2^n - 1)$ such gates.
- Any two-level unitary can be implemented using CNOT and single-qubit gates (we show how below) — so the set {CNOT, single-qubit gates} is **universal**.

Real Givens Rotation

- The Givens rotation $G(i, j, \theta)$ performs a rotation in the (i, j)-plane, leaving all other axes invariant:

$$G(i,j,\theta) = \begin{pmatrix} I_{i-1} & 0 & 0 & 0 & 0 \\ 0 & \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 0 & I_{j-i-1} & 0 & 0 \\ 0 & \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 0 & I_{d-j} \end{pmatrix}.$$

- For a given column vector $[a, b]^\top \in \mathbb{R}^2$, choose θ such that

$$\cos\theta = \frac{a}{\sqrt{a^2+b^2}}; \quad \sin\theta = \frac{b}{\sqrt{a^2+b^2}}.$$

  Then

$$\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sqrt{a^2+b^2} \\ 0 \end{pmatrix}$$

- Repeat this process column by column to zero out an entire matrix's off-diagonal entries, exactly as in QR decomposition.

Complex Givens Rotation

- A complex Givens rotation in $SU(2)$ uses *two* angles instead of one:

$$G(i,j,\theta,\phi) = \begin{pmatrix} \cos\theta & -e^{i\phi}\sin\theta \\ e^{-i\phi}\sin\theta & \cos\theta \end{pmatrix}$$

- If $a = |a|e^{i\theta_a}$, $b = |b|e^{i\theta_b}$, choose

$$\begin{cases} \theta & := & \arctan\left(\frac{|b|}{|a|}\right), \\ \phi & := & \theta_a - \theta_b \pm \pi. \end{cases}$$

  Then

$$\begin{pmatrix} \cos\theta & -e^{i\phi}\sin\theta \\ e^{-i\phi}\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sqrt{|a|^2+|b|^2}\, e^{i\theta_a} \\ 0 \end{pmatrix}$$

- Exactly as in the real case, repeating this column by column zeros out all off-diagonal entries of any U, proving the two-level decomposition exists.

The Hardware Obstacle: Non-Adjacent Bitstrings

- A two-level unitary acting on basis states $|a\rangle$ and $|b\rangle$ may involve *non-adjacent* bitstrings (e.g., $|00\rangle$ and $|11\rangle$, which differ in *two* bits).
  - ◇ In linear algebra, this is never an issue — a two-level unitary is just a $d \times d$ matrix, regardless of which two basis labels it touches.
  - ◇ Physical quantum hardware, however, is fundamentally constrained to *local* operations: a standard single- or two-qubit gate can only directly couple two basis states that differ by exactly **one** bit (e.g., $|00\rangle$ and $|01\rangle$).
  - ◇ If the two active bitstrings are non-adjacent, no single gate can directly couple them.
- **The fix:** use a *Gray code* — a sequence of single-bit-flip steps connecting the two bitstrings — to walk the amplitude over to an adjacent pair, apply the rotation there, then walk it back. We build this up over the next several slides, starting with the easy (already-adjacent) case.

Adjacent Case: The Algebra

- Consider $\mathscr{H} = \mathbb{C}^2 \otimes \mathbb{C}^2$. Any state vector $|\psi\rangle \in \mathscr{H}$ can be uniquely partitioned as:

$$\begin{aligned} |\psi\rangle &= c_{00}|00\rangle + c_{01}|01\rangle + c_{10}|10\rangle + c_{11}|11\rangle \quad (2) \\ &= |0\rangle \otimes v_0 + |1\rangle \otimes v_1, \quad v_0, v_1 \in \mathbb{C}^2. \quad (3) \end{aligned}$$

- The adjacent two-level unitary U_{adj} acting on coordinates $|00\rangle$ and $|01\rangle$ (which differ only in the second/last bit) can be expressed as

$$\begin{aligned} U_{\text{adj}} &= \left(\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \otimes R(\theta)\right) + \left(\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \otimes I\right) \\ &= \left(|0\rangle\langle 0| \otimes R(\theta)\right) + \left(|1\rangle\langle 1| \otimes I\right). \end{aligned}$$

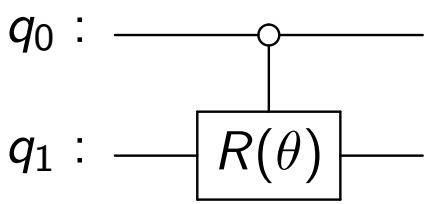
- Applying U_{adj} to the partitioned state vector $|\psi\rangle$:

$$U_{\text{adj}}|\psi\rangle = |0\rangle \otimes R(\theta)v_0 + |1\rangle \otimes v_1$$

- **Key point:** this action is completely *local* — it touches only the second qubit, conditioned on the first. We make this circuit explicit next.

Adjacent Case: The Circuit

- The transformation $U_{\text{adj}} |\psi\rangle = |0\rangle \otimes R(\theta) v_0 + |1\rangle \otimes v_1$ is *block diagonal*, hence directly implementable on hardware: the first tensor factor maps to the control wire (q_0), and the second factor maps to the target wire (q_1).
  - The **open control** circle ($\circ$), the natural variant of a closed control ($\bullet$), fires the rotation *only* when $q_0 = |0\rangle$ — restricting $R(\theta)$ exclusively to the v_0 component:



  - The v_1 component (the $|1\rangle_{q_0}$ subspace) is paired with the identity operator and remains untouched, exactly as required.
- This handles any two-level unitary acting on *adjacent* bitstrings. What about the non-adjacent case motivating this whole discussion?

Non-Adjacent Case: The Obstruction

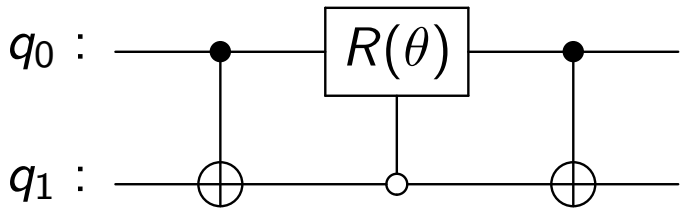
- Consider the non-adjacent active coordinates $|00\rangle$ and $|11\rangle$ (differing in *both* bits). The matrix representation U_{non} of this Givens rotation has the form:

$$U_{\text{non}} = \begin{pmatrix} \cos\theta & 0 & 0 & -\sin\theta \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \sin\theta & 0 & 0 & \cos\theta \end{pmatrix}$$

- When applied to $|\psi\rangle = |0\rangle \otimes v_0 + |1\rangle \otimes v_1$:
  - Entries of v_0 and v_1 get mixed *across* the boundary between the two subsystems.
  - The result cannot be factored into the decoupled, block-diagonal form $|0\rangle\langle 0| \otimes A + |1\rangle\langle 1| \otimes B$ that made the adjacent case implementable.
  - Consequently, U_{non} cannot be executed directly as a single local gate on hardware.

Non-Adjacent Case: The Remedy

- **The trick:** temporarily relabel the basis with a CNOT, so the two active states *become* adjacent, apply the easy adjacent-case rotation, then undo the relabeling.
  - Formally, this is a similarity transformation by a CNOT-generated permutation P:
    $$U_{\text{non}} = P^{-1} U_{\text{adj}} P$$
  - Physical implementation:



- Converting the non-adjacent pair $|11\rangle$ to the adjacent pair $|10\rangle$:
  - **Stage 1 (P):** the first CNOT maps $|11\rangle \to |10\rangle$, bringing the target amplitudes into single-bit adjacency on q_1.
  - **Stage 2 (U_{adj}):** $R(\theta)$ acts on q_1, restricted (via open control) to $q_0 = |0\rangle$.
  - **Stage 3 (P^{-1}):** the terminal CNOT restores the original basis labeling.

Gray Code: Step-by-Step State Tracking (I)

- Suppose $v_0 = \alpha |0\rangle + \beta |1\rangle$ and $v_1 = \gamma |0\rangle + \delta |1\rangle$, so

$$|\psi_0\rangle = \alpha |00\rangle + \beta |01\rangle + \gamma |10\rangle + \delta |11\rangle .$$

- **Stage 1 (First CNOT):**

$$\begin{aligned} |\psi_1\rangle &= P |\psi_0\rangle = \alpha |00\rangle + \beta |01\rangle + \gamma |11\rangle + \delta |10\rangle \\ &= \underbrace{(\alpha |0\rangle + \delta |1\rangle)}_{\tilde{v}_0} \otimes |0\rangle + \underbrace{(\beta |0\rangle + \gamma |1\rangle)}_{\tilde{v}_1} \otimes |1\rangle . \end{aligned}$$

- **Stage 2 (Open-controlled rotation, fires when $q_0 = |0\rangle$):** let $R(\theta)\tilde{v}_0 = \alpha' |0\rangle + \delta' |1\rangle$.

$$|\psi_2\rangle = U_{\text{adj}} |\psi_1\rangle = \alpha' |00\rangle + \beta |01\rangle + \gamma |11\rangle + \delta' |10\rangle$$

Gray Code: Step-by-Step State Tracking (II)

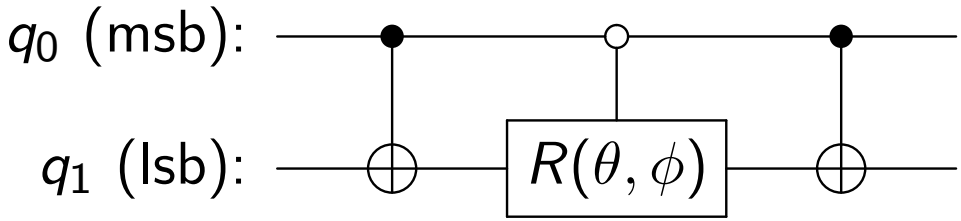
- **Stage 3 (Terminal CNOT):** reverses the initial permutation, restoring the standard basis labeling:

$$\begin{aligned} |\psi_3\rangle &= P^{-1}|\psi_2\rangle = \alpha'|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta'|11\rangle \\ &= |0\rangle \otimes R(\theta)v_0 + |1\rangle \otimes v_1. \end{aligned}$$

- This matches *exactly* the target U_{non} action — only v_0 (associated with $|00\rangle, |11\rangle$, the two non-adjacent original states) was rotated; v_1 is untouched. ✓

The General n-Qubit Gray Code Circuit

- For more than 2 qubits, the same idea generalizes: walk the active amplitude through a sequence of single-bit-flip (Gray code) steps until the two target bitstrings are adjacent, apply the local rotation, then walk back.



- *Mechanism:* CNOTs "swap" the amplitude through the Gray-code path until the two active states are exactly one bit apart; after the rotation, the swap sequence is reversed to restore the original basis.
- **Conclusion:** since (a) any U decomposes into two-level unitaries (Givens rotations), and (b) any two-level unitary — adjacent or not — can be implemented with CNOTs and single-qubit gates via this Gray-code procedure, the set {CNOT, single-qubit gates} is confirmed **universal** for quantum computation.

The Challenge of Hamiltonian Simulation

- Simulating a quantum system means solving the Schrödinger equation:

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

  - For time-independent H, the solution is $|\psi(t)\rangle = e^{-iHt} |\psi(0)\rangle$ (setting $\hbar = 1$).
  - Most physical Hamiltonians are sums of *local* interactions: $H = \sum_k H_k$, each H_k acting on only a few qubits.
- *Why this is intractable directly:*
  - **Exponential growth:** if $n = 100$, H is a $2^{100} \times 2^{100}$ matrix — too large to even write down.
  - **Non-commutativity:** if $[H_j, H_k] \neq 0$, then $e^{-i(H_j+H_k)t} \neq e^{-iH_j t} e^{-iH_k t}$, so we cannot simulate each H_k separately and combine results.
- **The path forward:** approximate e^{-iHt} using only the *easy* pieces $e^{-iH_k t}$, accepting a small, controllable error. This is the Trotter-Suzuki idea, next.

Why Not Solve It Exactly? The BCH Formula

- Let $X, Y \in \mathbb{C}^{m\times m}$. They commute iff $[X, Y] = 0$.
- **Theorem (Baker-Campbell-Hausdorff):** if $e^X e^Y = e^Z$, then

$$Z = X + Y + \frac{1}{2}[X, Y] + \frac{1}{12}[X, [X, Y]] - \frac{1}{12}[Y, [X, Y]] + \mathcal{O}(\text{brackets}^4).$$

- **Theorem (Zassenhaus):** expanding e^{X+Y} instead,

$$e^{X+Y} = e^X e^Y e^{-\frac{1}{2}[X,Y]} e^{\frac{1}{6}(2[Y,[X,Y]]+[X,[X,Y]])} \dots .$$

- Both series terminate after the third term *only* if $[X, Y]$ commutes with both X and Y — otherwise they continue indefinitely with ever more deeply nested commutators.
- **The takeaway:** these formulas are mathematically *exact*, but for generic non-commuting Hamiltonian pieces they require infinitely many terms — useless as a practical circuit recipe. This is precisely why we instead use an *approximate* formula, accurate only to a controlled, finite order: the Trotter-Suzuki formula, next.

The Trotter Product Formula (First Order)

- For a small time step Δt, truncating the BCH series after the leading term gives an error of only $O(\Delta t^2)$:

$$e^{-i(H_1+H_2)\Delta t} = e^{-iH_1\Delta t}\, e^{-iH_2\Delta t} + O(\Delta t^2).$$

- **Application:** partition the total simulation time t into N segments of size $\Delta t = t/N$, and apply the gates for each H_k in sequence, once per slice.
  - Local error per slice is $O((t/N)^2)$; summing over N slices gives total error $O(t^2/N)$.
  - **Practical consequence:** to reach a target precision ϵ, we need $N = O(t^2/\epsilon)$ time slices — a concrete, computable resource estimate for any simulation.

The Suzuki Product Formula (Second Order)

- A more accurate, time-symmetric variant:

$$S_2(\Delta t) = e^{-iH_1\Delta t/2}\, e^{-iH_2\Delta t}\, e^{-iH_1\Delta t/2}$$

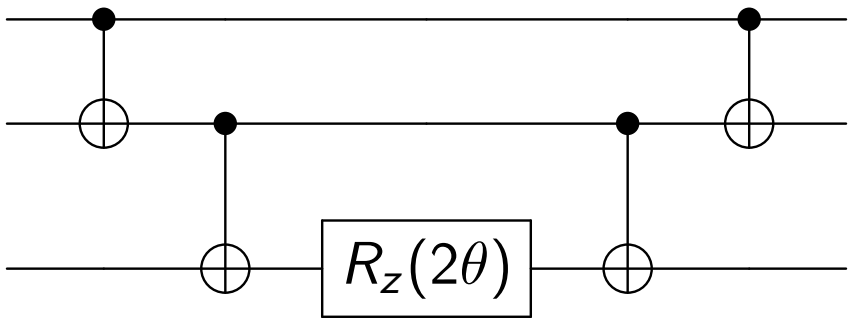
- This symmetric form has local error $O(\Delta t^3)$, giving total error $O(t^3/N^2)$ over N slices — a *quadratic* improvement over the first-order formula's $O(t^2/N)$, at the cost of one extra gate layer per slice.
- There are many other, even higher-order, variants, all trading extra gate layers for faster error suppression as N grows.

The General Recipe: Phase Gadgets

- For a Hamiltonian term that is a single **Pauli string** (e.g. $H_k = Z \otimes Z \otimes Z$), the recipe for implementing $e^{-i\theta H_k}$ has a clean, reusable structure:
  1. **Compute the parity:** use CNOTs to XOR all the qubits where H_k acts as Z into a single qubit.
  2. **Apply a phase:** a single R_z rotation on that qubit applies the correct phase, conditioned on the parity.
  3. **Uncompute the parity:** reverse the CNOTs from step 1 to restore the original qubits.
- This is exactly the same compute→act→uncompute pattern from Bennett's trick (Lecture 5) and the ancilla ladder (Section 3) — a recurring theme in practical quantum circuit design.
- We now work through the concrete case $H = Z \otimes Z \otimes Z$ in full detail.

Simulating Parity: $H = Z \otimes Z \otimes Z$

- Following the recipe above: compute the parity of the three qubits, apply a conditional Z-rotation, then uncompute the parity.
- Apply the circuit to $|\psi_0\rangle = |x, y, z\rangle$ where $x, y, z \in \{0, 1\}$:



ZZZ Phase Gadget: State Evolution

- The state propagates sequentially through each gate slice:

$$
\begin{aligned}
|\psi_1\rangle &= |x, x \oplus y, z\rangle\,, \\
|\psi_2\rangle &= |x, x \oplus y, x \oplus y \oplus z\rangle\,, \\
|\psi_3\rangle &= e^{-i\theta(-1)^{x\oplus y\oplus z}} |x, x \oplus y, x \oplus y \oplus z\rangle\,, \\
|\psi_4\rangle &= e^{-i\theta(-1)^{x\oplus y\oplus z}} |x, x \oplus y, z\rangle\,, \\
|\psi_5\rangle &= e^{-i\theta(-1)^{x\oplus y\oplus z}} |x, y, z\rangle\,.
\end{aligned}
$$

- Step $\psi_2 \to \psi_3$: $R_z(2\theta)\,|b\rangle = e^{-i\theta(-1)^b}\,|b\rangle$. Steps $\psi_3 \to \psi_5$: uncompute (reverse the two CNOTs).

ZZZ Phase Gadget: Interpreting the Phase Kick

- Phase kick:

$$(-1)^{x \oplus y \oplus z} |x, y, z\rangle = (Z \otimes Z \otimes Z) |x, y, z\rangle ,$$

since each Z contributes a factor $(-1)^{\text{bit}}$, and XOR of bits matches multiplication of their ± 1 signs.

  - This circuit exactly synthesizes the non-local operator $U(\theta) = e^{-i\theta(Z \otimes Z \otimes Z)}$, confirming the recipe from the previous slide.

Basis Changes for Generic Pauli Simulation

- The phase-gadget recipe only directly handles Z-type Pauli strings. To simulate $H = X \otimes X \otimes X$, transform into the Z basis using Hadamards:

$$e^{-i(X\otimes X\otimes X)\theta} = (H^{\otimes 3})\, e^{-i(Z\otimes Z\otimes Z)\theta}\, (H^{\otimes 3}).$$

  - Conjugation by H swaps the Pauli-Z and Pauli-X bases: $HXH = Z$ and $HZH = X$, so

$$(H^{\otimes 3})(Z \otimes Z \otimes Z)(H^{\otimes 3}) = X \otimes X \otimes X.$$

- For Y interactions, use the basis-change $S^\dagger H$ instead:
  - $U = S^\dagger H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ -i & i \end{pmatrix}$, and $(S^\dagger H)\, Z\, (S^\dagger H)^\dagger = -Y$.

Putting It All Together

- Any local Hamiltonian $H = \sum_k H_k$ decomposed into a sum of Pauli strings can be simulated by:
  1. **Trotterizing** across the different terms H_k (Trotter or Suzuki formula, two slides ago).
  2. For each H_k, **basis-changing** into the all-Z frame (using H for X-terms, $S^\dagger H$ for Y-terms, identity for Z-terms), applying the **phase-gadget circuit**, then basis-changing back.
- This is the complete, practical recipe for simulating *any* local quantum system on a quantum computer — the synthesis of every tool developed in this lecture.

Summary of Lecture 6 (I)

- **Synthesis:** any single-qubit unitary decomposes as $U = e^{i\alpha} R_z(\beta) R_y(\gamma) R_z(\delta)$ (Z-Y decomposition), with an explicit formula for each angle from U's entries.
- **Controlled-U:** the ABC construction ($ABC = I$, $e^{i\alpha} AXBXC = U$) builds a controlled version of *any* single-qubit gate from CNOTs and single-qubit gates.
- **Multi-control:** $C^n(U)$ scales linearly, not exponentially, using an ancilla ladder — but the ladder must be *uncomputed* afterward, or garbage ancillas destroy interference.

Summary of Lecture 6 (II)

- **Measurement:** the deferred- and implicit-measurement principles let us treat any algorithm as purely unitary until the final readout.
- **Universality:** any unitary decomposes into two-level (Givens) rotations; Gray codes implement even *non-adjacent* two-level unitaries using only local gates — confirming {CNOT, single-qubit gates} is universal.
- **Simulation:** the Trotter-Suzuki formula approximates e^{-iHt} for non-commuting Hamiltonian pieces; phase-gadget circuits (compute parity, rotate, uncompute) implement each Pauli-string term, with basis changes (H, $S^\dagger H$) handling X and Y interactions.

Looking Ahead: Lecture 7

With a full synthesis and simulation toolkit in hand, we turn to one of the most important subroutines in quantum computing. In Lecture 7 we will cover:

- **The Quantum Fourier Transform (QFT)** and its circuit implementation.
- **Quantum phase estimation**, built directly from controlled-U gates (this lecture's Section 2).
- The foundation for **Shor's factoring algorithm**, covered in Lectures 7–8.