Module 2 — Parameterized Circuits

Variational Quantum Algorithms

hbar.university

Ansatz Design

Position in Knowledge Graph

System: Variational Quantum Algorithms **Structural Domain:** Parameterized Circuits **Node:** 2.1

This is the first node of Module 2 and the **anchor of the quantum half** of the course. Module 1 told you *that* the optimizer interacts with a parameterized state $|\psi(\boldsymbol{\theta})\rangle$. Module 2 starts here by asking the prior question: **what is $U(\boldsymbol{\theta})$, and how is it chosen?**

The answer determines almost everything downstream — what states are reachable, what the cost landscape looks like, whether a barren plateau will appear, and how many shots an optimizer will need. Choosing an ansatz is the single highest-leverage design decision in a VQA.

What an Ansatz Is

An **ansatz** is a parameterized recipe for preparing a family of quantum states. Formally, it is a unitary

$$U(\boldsymbol{\theta}) = \prod_{\ell=1}^{L} U_\ell(\boldsymbol{\theta}_\ell)$$

built from a sequence of layers U_ℓ, each of which depends on a subset of the parameter vector. Acting on a fixed reference state (typically $|0\rangle^{\otimes n}$), the ansatz defines

$$|\psi(\boldsymbol{\theta})\rangle = U(\boldsymbol{\theta})|0\rangle^{\otimes n}.$$

The set of states reachable by varying $\boldsymbol{\theta} \in \mathbb{R}^P$ is the **ansatz manifold** — a strict, structured subset of the full Hilbert space.

The word "ansatz" is borrowed from German theoretical physics and means "starting form" or "trial function." A variational quantum algorithm is, at its core, the act of letting an optimizer slide along this trial form looking for the best parameter assignment.

The Three Design Axes

Every ansatz design choice can be located along three axes.

1. Expressivity — *what states can it reach?* A more expressive ansatz covers more of Hilbert space. At the extreme, a fully universal ansatz can prepare any state, but at the cost of needing exponentially many parameters or exponentially deep circuits. At the other extreme, a 1-parameter ansatz can only prepare

a 1-parameter family, regardless of how cleverly you optimize. The interesting region is in between: **just expressive enough to contain a good approximation to the target state, and no more.**

2. Trainability — *can the optimizer find a good point on the manifold?* A maximally expressive ansatz often has barren plateaus (Module 4), meaning the cost gradient vanishes over almost all of parameter space and the optimizer cannot make progress. A less expressive ansatz with the right inductive bias may be much easier to train, even if its theoretical reach is smaller. **Expressivity and trainability are in tension** — Module 8 is dedicated entirely to this tension.

3. Hardware compatibility — *can it run on the device you have?* Real near-term devices have a limited gate set (typically single-qubit rotations + a 2-qubit entangler such as CNOT or CZ), restricted qubit connectivity (only some pairs of qubits can be entangled directly), and short coherence times (gates must complete before noise destroys the state). An ansatz that is mathematically beautiful but compiles to a circuit that is too deep, requires too many SWAPs, or uses gates the device doesn't support is operationally useless.

A good ansatz design is a Pareto-optimal point on these three axes for the problem you actually care about.

The Two Cultural Camps

VQA ansatz design splits into two communities with very different starting points.

Hardware-first designers (node 2.2): build the ansatz from gates the hardware natively supports, lay them out in a regular pattern (often just "alternating layers of single-qubit rotations and entanglers"), and hope the resulting parameterization is rich enough to contain the answer. This is the **hardware-efficient ansatz**. Simple, agnostic to the problem, runs on anything. Frequently barren-plateau-prone.

Problem-first designers (node 2.3): start from physics or chemistry knowledge, write down a state-preparation recipe motivated by the problem structure (e.g., a unitary coupled-cluster expansion for molecules), then compile that down to hardware gates. This is the **chemically-inspired** or more generally **problem-inspired** ansatz. Better inductive bias, often deeper, often less hardware-friendly.

Both camps are correct about something the other camp is wrong about, and the right answer for any given VQA usually involves elements of both. Modules 2.2 and 2.3 examine each in detail.

What an Ansatz Is Not

It is worth being explicit about what the ansatz does not do.

- **An ansatz does not encode the Hamiltonian.** The Hamiltonian H is the *measurement*, applied to the state the ansatz prepares. The ansatz only prepares the state. (See node 2.5 for how the Hamiltonian is encoded — separately — into a measurement protocol.)
- **An ansatz does not adapt during a single run.** The structure of $U(\boldsymbol{\theta})$ is fixed throughout the optimization; only the parameters $\boldsymbol{\theta}$ change. *Adaptive ansatz methods* (ADAPT-VQE) violate this by growing the ansatz iteratively, but they are a special case treated separately.
- **An ansatz is not unique.** For any target state, infinitely many ansätze can prepare it. The art is choosing one whose parameter landscape is friendly to the optimizer you intend to use.

Structural Role

This node opens Module 2 and sets up everything that follows in it. The next four nodes are direct elaborations of the design axes introduced here:

- 2.2: hardware-efficient ansätze (the hardware-first culture)

- 2.3: chemically-inspired ansätze (the problem-first culture)
- 2.4: compilation, depth, and width — the hardware compatibility axis in detail
- 2.5: Hamiltonian encoding — the *separate* problem of how the Hamiltonian becomes a measurement

The remaining nodes (2.6-2.9) deal with how states are represented, how measurement actually happens on hardware, and the geometric view of the ansatz as a manifold — the deeper structural understanding that pays off when you start reasoning about cost landscapes (Module 3) and barren plateaus (Module 4).

Relations to Other Concepts

- **Variational principle (1.1)** — guarantees that minimizing $\langle\psi(\boldsymbol{\theta})|H|\psi(\boldsymbol{\theta})\rangle$ is a meaningful thing to do regardless of how restricted the ansatz is.
- **Inductive bias in classical ML** — the choice of ansatz is the quantum analogue of the choice of neural network architecture. Both restrict the function class; both inject prior knowledge; both can be over- or under-expressive.
- **Tensor network states** — DMRG, MPS, PEPS — classical analogues of restricted state families with controlled expressivity.
- **Universal quantum computation** — any unitary can in principle be approximated by a sufficiently deep circuit; ansatz design is the art of *not* using a universal one.

Worked Example — Counting the Reachable States

Consider the simplest non-trivial ansatz: a single $R_Y(\theta)$ gate on one qubit, applied to $|0\rangle$.

$$|\psi(\theta)\rangle = R_Y(\theta)|0\rangle = \cos(\theta/2)|0\rangle + \sin(\theta/2)|1\rangle.$$

The set of reachable states is the **great circle** in the Bloch sphere passing through $|0\rangle$, $|+\rangle$, $|1\rangle$, $|-\rangle$. This is a 1-parameter family — a 1D submanifold of a 2D Hilbert space.

Now add a second layer: $R_Z(\phi)R_Y(\theta)|0\rangle$. The reachable set is now the **entire Bloch sphere** — a 2-parameter family covering all pure single-qubit states. Adding a third parameter $R_Y(\alpha)R_Z(\phi)R_Y(\theta)|0\rangle$ gains nothing in terms of *reachable states* (you can already prepare every single-qubit pure state), but it changes the *parameterization geometry* — the same target state can now be reached via multiple paths, which can either help (more escape routes for an optimizer) or hurt (redundant directions in parameter space).

Lesson: **the number of parameters is not the same as the dimension of the reachable set.** Adding parameters past the point of universality only changes geometry, not coverage — and that geometry change can be either friendly or hostile to optimization.

Visualization

Pending. Three-panel diagram. Panel 1: Bloch sphere with a great circle traced through it (single R_Y ansatz, 1-parameter). Panel 2: full Bloch sphere covered (2-parameter ansatz). Panel 3: same Bloch sphere, but with multiple curves all passing through the same point (3-parameter ansatz, redundant). _Caption: "More parameters can mean more reach, or just more parameterization — these are different._"

Exercises

1. For the 1-parameter ansatz $R_Y(\theta)|0\rangle$, what is the maximum value of $\langle Z\rangle$? What is the minimum? Sketch $\langle Z\rangle(\theta)$ over $\theta \in [-\pi, \pi]$.

2. Show that the ansatz $R_X(\theta_1)R_X(\theta_2)|0\rangle$ is *not* universal for single-qubit states (despite having 2 parameters). What parameter geometry is going wrong?
3. (VQA) Sketch the design tradeoff between expressivity, trainability, and hardware compatibility as a 2D Pareto front for a hypothetical "ideal" VQA ansatz family. Where on the front would you put a hardware-efficient ansatz vs a UCCSD ansatz?

Research Extensions

- **Adaptive ansatz construction (ADAPT-VQE)** — grow the ansatz one operator at a time, choosing the next operator based on its energy gradient. Trades depth for expressivity *only when it helps*.
- **Symmetry-preserving ansätze** — restrict the manifold to states obeying problem symmetries (particle number, total spin, parity), shrinking it to the physically relevant subspace.
- **Learned ansätze** — meta-learn an ansatz structure across families of problems using classical ML pipelines.

Cross-links to Other Courses

- **Machine Learning** — model architecture as inductive bias, capacity vs trainability.
- **Differential Geometry** — submanifolds of complex projective space.
- **Quantum Computing** — universal gate sets, gate decomposition, circuit synthesis.
- **Statistics Module 5** — restricted models and their bias-variance tradeoff.

Hardware-Efficient Ansatz

Position in Knowledge Graph

System: Variational Quantum Algorithms **Structural Domain:** Parameterized Circuits **Node:** 2.2

The hardware-efficient ansatz (HEA) is the most widely deployed ansatz family in NISQ-era VQAs. It is also the most widely **maligned**, because most barren-plateau theorems were proved against it. Both facts are true, and both are a consequence of the same design philosophy: **build it out of whatever the hardware can do, and let the optimizer figure out the rest.**

The Construction

A hardware-efficient ansatz has three structural ingredients:

1. **A native gate set.** Whatever single-qubit rotations and two-qubit entanglers the device supports natively. Typical: R_X, R_Y, R_Z for single-qubit; CNOT, CZ, or echoed cross-resonance for two-qubit.
2. **A connectivity-respecting layout.** Two-qubit gates only between qubits that are physically adjacent on the device. No virtual SWAPs unless absolutely necessary.
3. **A repeated layer structure.** Stack L identical (up to parameter values) layers, each of the form *single-qubit rotations on every qubit, followed by entanglers between connected pairs.*

A canonical layer of an n-qubit HEA looks like:

$$U_\ell(\boldsymbol{\theta}_\ell) = \left(\prod_{(i,j)\in E} \mathrm{CNOT}_{ij} \right) \cdot \left(\bigotimes_{q=1}^{n} R_Y(\theta^Y_{\ell,q}) R_Z(\theta^Z_{\ell,q}) \right),$$

where E is the set of native qubit pairs. The full ansatz is $U(\boldsymbol{\theta}) = U_L \cdots U_2 U_1$ acting on $|0\rangle^{\otimes n}$.

For n qubits and L layers with two single-qubit parameters per qubit per layer, the parameter count is $P = 2nL$ — linear in both dimensions. This is small enough to be manageable but large enough that for modest n and L, the ansatz becomes expressively rich.

Why It Was Adopted

When IBM, Google, Rigetti, and IonQ first built devices that could run real circuits, the question facing VQA practitioners was practical: **what ansatz can I actually run on this thing tomorrow?**

The HEA answers that question definitively:

- It uses **only native gates**, so no compilation magic is needed.
- It respects **device connectivity**, so no SWAP overhead.
- It is **shallow** (depth $= O(L)$, independent of n), so it fits inside a coherence window.
- It is **agnostic to the problem**, so the same ansatz template works for any Hamiltonian.

The Kandala et al. (2017) paper that introduced the term demonstrated VQE on H_2, LiH, and BeH_2 using this exact recipe and produced energy estimates that, while not chemically accurate, were close enough to demonstrate the approach. That paper became the template for hundreds of follow-up experiments.

Why It Was Maligned

The McClean et al. (2018) barren plateau paper proved that for sufficiently deep, sufficiently random HEAs on sufficiently many qubits, the variance of the cost gradient $\mathrm{Var}[\partial_i C]$ scales as $O(1/4^n)$. This is exponential in the number of qubits.

In practical terms: a randomly initialized HEA on 50 qubits has a gradient so small that no optimizer, no matter how clever, can detect a descent direction without an exponential number of shots. The cost landscape looks **flat** to any local optimizer. The ansatz is mathematically rich enough to contain the answer, but the optimizer cannot find it.

This is the **barren plateau** phenomenon (Module 4 in full), and it is the central technical reason that HEAs are no longer the default recommendation for large VQAs.

The Tension

The HEA's strengths and weaknesses come from the same source.

It is a **universal approximator** in the limit of large L — meaning the manifold it parameterizes covers all of Hilbert space as the number of layers grows. That universality is what makes it general-purpose. It is also what makes the cost landscape look like a uniformly random function on Hilbert space, which is exactly the setting where barren plateaus are guaranteed.

In short: **the HEA is too good at being expressive, and not good enough at being structured.** A problem-inspired ansatz (node 2.3) gives up some expressivity in exchange for inductive bias that lets the optimizer find a good answer in linear rather than exponential queries.

When To Use It Anyway

Despite the criticism, the HEA remains a defensible choice in several settings:

- **Small qubit count** ($n \leq 8$ or so): the barren plateau scaling doesn't bite hard yet, and the HEA's hardware-friendliness wins.
- **Warm-started initialization**: if you have a good initial guess from a classical pre-computation, you can avoid the random-initialization regime where barren plateaus appear, and the HEA may train fine.
- **Symmetry-restricted variants**: an HEA can be modified to preserve problem symmetries (e.g., particle number), which both shrinks the manifold and can mitigate barren plateaus.
- **Benchmarking baseline**: when introducing a new optimization technique, the HEA is the standard baseline ansatz against which improvements are reported. You can't avoid running on it.

The thesis takes an HEA-with-warm-start as one of its evaluation conditions for exactly the reasons above.

Structural Role

This node is one half of the "what kind of ansatz?" pair (the other half is 2.3). It establishes the hardware-first design culture and surfaces the central pathology — barren plateaus — that all subsequent ansatz design tries to escape from. Module 4 returns to this in full when it proves the barren plateau theorem and analyzes its consequences for HEA-based VQAs.

Relations to Other Concepts

- **2-designs and t-designs** — random circuits drawn from sufficiently deep HEAs approximate Haar-random unitaries; the barren plateau theorem uses this to bound gradient variance.
- **Ising-type ansätze** — HEAs whose entangling layers come from time-evolution under an Ising Hamiltonian; structurally similar.
- **Brick-wall circuits** — a specific HEA layout common in tensor network and quantum simulation literature.
- **QAOA (quantum approximate optimization algorithm)** — a different ansatz philosophy where the layers come from time-evolving the problem and mixer Hamiltonians; structurally distinct from HEAs.

Worked Example — A 4-Qubit HEA

Consider a 4-qubit HEA with linear connectivity (qubits 1-2-3-4) and 3 layers:

- Per layer: 4 qubits $\times$ 2 rotations/qubit = 8 single-qubit parameters.
- Per layer: 3 entanglers (CNOTs between 1-2, 2-3, 3-4).
- Total parameters: $3 \times 8 = 24$.
- Total CNOTs: $3 \times 3 = 9$.
- Circuit depth in two-qubit gates: 3 (one entangling layer per ansatz layer, applied in parallel where possible).

For H_2 at equilibrium geometry, this ansatz is expressive enough to reach within ~1 mHa of the FCI ground state, given a good initialization. For a random initialization, the optimizer typically gets stuck on a plateau and cannot reach chemical accuracy.

The same ansatz template, scaled to 8 qubits with the same layer count, has 48 parameters and 18 CNOTs. At 16 qubits: 96 parameters, 42 CNOTs. The parameter count grows linearly, but the gradient variance shrinks as $1/4^n$ — and somewhere between $n = 12$ and $n = 20$, depending on details, the optimizer simply cannot make progress from random initialization.

Visualization

Pending. A circuit diagram showing 4 horizontal qubit lines, with vertical "rotation columns" alternating with "CNOT staircases" — three layers total. Below: a plot of $\log \mathrm{Var}[\partial_i C]$ *vs* n *showing the* $O(1/4^n)$ *scaling. Caption: "Universal in the limit of large* L *— and that's the problem."*

Exercises

1. Count the parameters in an HEA with $n = 6$ qubits, ring connectivity, $L = 5$ layers, and 3 single-qubit rotations per qubit per layer. What is the CNOT count?
2. For a 2-qubit HEA $U(\theta) = \mathrm{CNOT} \cdot (R_Y(\theta_1) \otimes R_Y(\theta_2)) \cdot \mathrm{CNOT} \cdot (R_Y(\theta_3) \otimes R_Y(\theta_4))|00\rangle$, what is the dimension of the reachable manifold? Is the ansatz universal for 2-qubit pure states? Why or why not?
3. (VQA) The barren plateau theorem requires the ansatz to form an approximate 2-design. Give an intuition for why a random HEA with sufficient depth satisfies this condition, and explain why a problem-inspired ansatz (e.g., UCCSD) does not.

Research Extensions

- **Layer-wise initialization strategies** — train layer 1 to convergence before unfreezing layer 2, etc., to escape the plateau regime.
- **Symmetry-preserving HEAs** — modify the HEA to commute with conserved quantities, restricting to a smaller and more trainable manifold.
- **Hybrid HEA + warm start** — use a classical method (Hartree-Fock, MPS, neural network state) to initialize HEA parameters in a non-random region.

Cross-links to Other Courses

- **Random Matrix Theory** — Haar measure, t-designs, concentration of measure.
- **Statistics Module 4** — variance of estimators in high-dimensional spaces.
- **Quantum Hardware** — native gate sets, qubit connectivity graphs, coherence times.
- **Module 4 (Barren Plateaus)** — the formal pathology that this node introduces informally.

Chemically Inspired Ansatz

Position in Knowledge Graph

System: Variational Quantum Algorithms **Structural Domain:** Parameterized Circuits **Node:** 2.3

If the hardware-efficient ansatz (2.2) is "build whatever the device supports and hope the optimizer can deal," then the chemically-inspired ansatz is the opposite philosophy: **start from physics, write down the state-preparation recipe the chemistry suggests, and only then ask whether you can compile it.**

The canonical example is **Unitary Coupled Cluster with Singles and Doubles (UCCSD)**, the variational quantum analogue of the gold-standard classical post-Hartree-Fock method. A chemically-inspired ansatz pays for its inductive bias with depth — UCCSD circuits are much deeper than HEAs of comparable parameter count — but the bias is so well-chosen that for chemistry problems it can reach the answer where HEAs cannot.

This node generalizes from UCCSD to the broader class of **problem-inspired ansätze** and explains the structural moves that make them work.

The Coupled Cluster Idea

Classical coupled cluster (CC) theory expresses the ground state of a molecule as

$$|\Psi_{\mathrm{CC}}\rangle = e^{T}|\Phi_{\mathrm{HF}}\rangle,$$

where $|\Phi_{\mathrm{HF}}\rangle$ is the Hartree-Fock reference (a single Slater determinant) and T is a "cluster operator" that creates excitations:

$$T = T_1 + T_2 + T_3 + \ldots, \quad T_1 = \sum_{ia} t_i^a a_a^\dagger a_i, \quad T_2 = \sum_{ijab} t_{ij}^{ab} a_a^\dagger a_b^\dagger a_j a_i, \quad \ldots$$

Truncating after T_2 gives **CCSD** — couples singles and doubles excitations only. The amplitudes t_i^a, t_{ij}^{ab} are the variational parameters.

The classical CCSD method is **non-variational** — e^T is not unitary, and the truncation produces an energy estimate that can drop below the true ground state. The quantum analogue fixes this by replacing e^T with the **anti-Hermitian** version:

$$|\Psi_{\mathrm{UCCSD}}\rangle = e^{T-T^\dagger}|\Phi_{\mathrm{HF}}\rangle.$$

Now $T - T^\dagger$ is anti-Hermitian, so $e^{T-T^\dagger}$ is unitary, and the resulting state can be prepared by a quantum circuit. Crucially, the variational principle (1.1) now applies — the energy $\langle\Psi_{\mathrm{UCCSD}}|H|\Psi_{\mathrm{UCCSD}}\rangle$ is a true upper bound on the ground state.

How UCCSD Becomes a Circuit

To run UCCSD on a quantum device, you need to express $e^{T-T^\dagger}$ as a sequence of native gates. The standard recipe:

1. **Map fermionic operators to Pauli operators** via Jordan-Wigner or Bravyi-Kitaev (see node 2.5). Each excitation operator $a_a^\dagger a_i$ becomes a sum of Pauli strings.
2. **Trotterize the exponential.** Because the Pauli strings in $T - T^\dagger$ generally don't commute, you approximate $e^{T-T^\dagger}$ by a Trotter expansion: $\prod_k e^{i\theta_k P_k} + O(\Delta t^2)$.

3. **Implement each $e^{i\theta_k P_k}$ as a gate sequence** — this is a *Pauli rotation* and has a standard decomposition: a basis change to Z, a CNOT staircase, an $R_Z(\theta_k)$, then the inverse staircase and basis change.

For a small molecule like H_2 in a minimal basis (4 spin-orbitals, 2 electrons), the resulting UCCSD ansatz has:

- 2 single excitations and 1 double excitation (after applying spin and number symmetries),
- ~80 two-qubit gates after Jordan-Wigner mapping and Trotterization,
- 3 free parameters.

Compare this to a 4-qubit HEA with 3 layers: 24 parameters, 9 CNOTs. **UCCSD has many fewer parameters but a much deeper circuit.**

Why This Tradeoff Is Often Worth It

The UCCSD parameter manifold is *much smaller* than the full Hilbert space — it lives in the symmetry sector with the right particle number, spin, and (often) point group, and within that sector it covers exactly the states reachable by physically meaningful electron rearrangements.

Three concrete consequences:

1. No barren plateau in the same sense. The barren plateau theorem applies to ansätze that approximate Haar-random unitaries. UCCSD does not — it has heavy structural constraints — and the gradient variance is not exponentially small in qubit number for chemistry-typical instances. The optimizer can find descent directions even from the Hartree-Fock starting point.

2. Natural warm start. The Hartree-Fock state is the all-zero parameter point ($T = 0$ means $e^0 = I$), and Hartree-Fock is already a reasonable approximation to the ground state. The optimizer starts in a region where the gradient is informative and the energy is meaningful. HEAs have no such natural starting point.

3. Chemical interpretability. Each parameter corresponds to a specific physical excitation, so the converged amplitudes can be interpreted (large T_2 amplitudes correspond to strong electron correlation in specific orbital pairs, etc.). This is rare in quantum machine learning and rare in HEAs; it is a feature unique to problem-inspired ansätze.

What You Pay For It

The downsides of UCCSD-style ansätze, in order of increasing severity:

- **Circuit depth.** A Trotterized UCCSD circuit can be 10× deeper than a comparable HEA. On NISQ devices, depth means decoherence, and depth means more total error per evaluation.
- **Trotter error.** The Trotterization introduces $O(\Delta t^2)$ error in the prepared state. This error competes with the optimization error and bounds how close to the true ground state you can get without infinite Trotter steps.
- **Domain specificity.** UCCSD is designed for fermionic chemistry. For spin lattice models, lattice gauge theories, or QML problems, the right "problem-inspired" ansatz is something different (Hamiltonian Variational Ansatz, QAOA-style p-step ansätze, etc.). There is no universal recipe.
- **Compilation complexity.** Mapping $T - T^\dagger$ to Pauli strings and then to gates is non-trivial. Standard libraries (Qiskit Nature, OpenFermion, Pennylane) handle it, but the resulting circuit is fragile to manual modification.

Beyond UCCSD: ADAPT-VQE and Cousins

Even within chemistry, UCCSD is not the end of the story. The biggest practical extension is **ADAPT-VQE** (Grimsley et al. 2019):

> Start with the Hartree-Fock state. At each iteration, compute the gradient of the energy with respect to every operator in a pool (say, all singles + doubles). Add the operator with the largest gradient magnitude to the ansatz. Re-optimize all parameters. Repeat.

ADAPT-VQE *grows* the ansatz one operator at a time, choosing only the operators that actually help. The result is a much shorter circuit than UCCSD — for many molecules, ADAPT reaches chemical accuracy with 10-100× fewer gates. The price is that the structure of the ansatz is not known in advance; it emerges from the optimization itself.

ADAPT is a representative of a broader move: **adaptive ansatz construction**, where the parameterization is co-designed with the optimization rather than fixed at the outset. Module 10 returns to this when it discusses adaptive control views of VQAs.

Structural Role

This node is the second half of the "what kind of ansatz?" dichotomy. Together with 2.2, it spans the design space along the hardware-first-vs-problem-first axis. Subsequent nodes assume you understand both poles and can talk about a given ansatz in those terms.

It also forwards to node 2.5 (Hamiltonian encoding), since the construction of UCCSD-style ansätze depends on the same fermion-to-qubit mapping that the Hamiltonian uses.

Relations to Other Concepts

- **Coupled cluster theory** — the classical method UCCSD is the quantum analogue of.
- **Configuration interaction** — an alternative classical method based on a linear (rather than exponential) ansatz; less compact but variational.
- **Hamiltonian Variational Ansatz (HVA)** — for spin models, the analogue of UCCSD: layers of $e^{i\theta H_k}$ using the problem Hamiltonian itself as the generator.
- **QAOA** — the same idea as HVA applied to combinatorial optimization, with alternating problem and mixer layers.
- **Symmetry-adapted ansätze** — restrict to a particular symmetry sector (particle number, spin, parity) by construction.

Worked Example — H_2 UCCSD

In a minimal STO-3G basis, H_2 has 4 spin-orbitals (2 spatial × 2 spin). After Jordan-Wigner mapping, 4 qubits. Hartree-Fock state: $|1100\rangle$ (two electrons in the lowest spatial orbital, both spins). The relevant excitations:

- Single excitations: $a_2^\dagger a_0$, $a_3^\dagger a_1$ (2 amplitudes).
- Double excitation: $a_2^\dagger a_3^\dagger a_0 a_1$ (1 amplitude).

After symmetry reduction, the double excitation is the dominant degree of freedom — singles cancel for H_2 at equilibrium. A 1-parameter UCCD ansatz suffices for chemical accuracy:

$$|\psi(\theta)\rangle = e^{\theta(a_2^\dagger a_3^\dagger a_0 a_1 - a_0^\dagger a_1^\dagger a_2 a_3)}|1100\rangle.$$

After Jordan-Wigner, this becomes a sum of 8 Pauli strings, and after Trotterization, ~10 CNOTs and 1 R_Z. **1 parameter, ~10 gates, chemical accuracy on H_2.** A 4-qubit HEA needs ~24 parameters and at least 9 CNOTs to reach the same accuracy — and only with the right initialization.

The lesson: **inductive bias is worth a lot of parameters.**

Visualization

Pending. A circuit diagram of the H_2 UCCD ansatz: 4 qubits initialized to $|1100\rangle$, then a single Pauli rotation block. Annotation: "1 parameter, chemistry-aware, hits the answer." Side-by-side comparison with a 4-qubit HEA labeled "24 parameters, fights the optimizer."

Exercises

1. Show that for H_2 in a minimal basis, the singles excitations $a_2^\dagger a_0$ and $a_3^\dagger a_1$ do not contribute to the ground state at equilibrium geometry, by appealing to spin and orbital symmetry.
2. Why is the *anti-Hermitized* exponential $e^{T-T^\dagger}$ used in UCCSD rather than e^T? What property of the resulting state would be lost with the latter?
3. (VQA) ADAPT-VQE chooses operators greedily by gradient magnitude. Construct a toy example where this greedy choice is suboptimal — i.e., the operator with the largest gradient is not the one that should be added next.

Research Extensions

- **k-UpCCGSD** — a generalization of UCCSD that uses generalized singles and doubles (any spin-orbital pair) and allows multiple repetitions.
- **Hamiltonian Variational Ansatz (HVA)** — the spin-model analogue.
- **Operator pool design** — for ADAPT, the choice of pool fundamentally constrains what circuits can be built. Dynamic pools are an active research area.
- **Nonlinear UCC** — extending the ansatz with non-truncated cluster operators in restricted form.

Cross-links to Other Courses

- **Quantum Chemistry** — coupled cluster theory, electronic structure methods.
- **Group Theory / Symmetry** — particle number, spin, point group symmetries in fermionic systems.
- **Module 5 (Regularization)** — UCCSD-style ansätze pair particularly well with the thesis's regularization techniques because their landscapes are more benign.
- **Module 8 (Expressivity vs Trainability)** — UCCSD is an exemplar of "give up some expressivity for much better trainability."

Compilation, Depth, and Width

Position in Knowledge Graph

System: Variational Quantum Algorithms **Structural Domain:** Parameterized Circuits **Node:** 2.4

The mathematical ansatz $U(\boldsymbol{\theta})$ and the physical circuit that runs on a device are not the same thing. Between them sits a **compiler** — a piece of software that translates the abstract gate sequence into one the hardware can actually execute, respecting native gate set, qubit connectivity, scheduling constraints, and calibration.

This node is the bridge from "ansatz on paper" to "circuit on metal." The relevant concepts are **width** (how many qubits), **depth** (how many sequential gate layers), and **gate count** (how many physical operations). On NISQ devices, all three are budgets, and compiling well means staying inside all three at once.

Width

Width is the number of qubits the circuit acts on. On a real device, width is bounded by the number of available, well-calibrated qubits.

Three subtleties:

- **Logical vs physical qubits**: every algorithm-level qubit consumes one device qubit — no error-correction overhead in the NISQ setting. (Fault-tolerant compilation is a different story.)
- **Connectivity**: not all qubit pairs can interact directly. A "100-qubit device" might only support entangling gates between specific neighbors. If your ansatz wants a CNOT between two qubits that aren't connected, the compiler must insert SWAPs to bring them adjacent — and each SWAP is itself a CNOT-heavy operation.
- **Calibration heterogeneity**: not all qubits are equally good. A 27-qubit device might only have 12 qubits with low enough error rates to be worth using on a given day.

Width is the **easiest budget to violate** — your problem either fits or it doesn't — and the **hardest to compress** without physically larger hardware.

Depth

Depth is the length of the longest path through the circuit (the number of gate layers that must execute sequentially). On a real device, depth is bounded by the **coherence time** — the qubits remain in a usable quantum state only for a finite duration before decoherence destroys the information.

A typical NISQ device has:

- Single-qubit gate time: ~20 ns
- Two-qubit gate time: ~200-500 ns
- T1/T2 coherence: 50-200 µs

This gives a depth budget of **a few hundred two-qubit gates** before the state is irrecoverably noisy. For a 100-CNOT circuit, you have already consumed most of the coherence window.

Depth is the most-often-violated budget for chemistry-style ansätze, which is why UCCSD is hard to run on real hardware and ADAPT-VQE (and other depth-saving variants) exist.

Critical distinction: depth vs gate count. A circuit with 1000 single-qubit gates spread across 10 qubits has gate count 1000 but depth only ~100, because gates on different qubits run in parallel. Depth is the time-axis quantity; gate count is the resource quantity. Both matter, for different reasons.

Gate Count

Gate count is the total number of operations. It governs **noise accumulation** — each gate has some error probability p, so the circuit fidelity scales roughly as $(1-p)^{\text{gate count}} \approx e^{-p \cdot \text{gate count}}$.

For $p = 10^{-3}$ (typical NISQ two-qubit error) and a 1000-CNOT circuit, the fidelity is $e^{-1} \approx 0.37$ — meaning roughly 60% of your runs are corrupted by at least one error. For a 100-CNOT circuit, fidelity is $e^{-0.1} \approx 0.90$ — much more workable.

Gate count is what makes "shallow" ansätze attractive: not because they finish faster (a single circuit is fast either way), but because they accumulate less error per shot.

What the Compiler Does

A compiler takes an abstract $U(\boldsymbol{\theta})$ and produces a runnable circuit. Its responsibilities, in roughly the order they happen:

1. Decompose abstract gates into native gates. $R_Y(\theta)$ might not be native; only R_X, R_Z, and CZ might be. The compiler rewrites $R_Y(\theta)$ using the available basis: $R_Y(\theta) = R_Z(-\pi/2)R_X(\theta)R_Z(\pi/2)$.

2. Route logical qubits to physical qubits. The compiler picks an assignment of algorithm qubits to device qubits that minimizes SWAP overhead given the device connectivity graph.

3. Insert SWAPs where needed. For two-qubit gates between non-adjacent qubits, SWAPs move the qubits adjacent. Each SWAP = 3 CNOTs, so this is expensive.

4. Schedule. Decide which gates run in parallel (on different qubits) and which run sequentially. Minimizes depth.

5. Optimize. Combine adjacent rotations, cancel redundant CNOTs, exploit gate identities. Modern compilers use heuristic and SAT-based passes.

6. Calibrate-aware mapping. Pick the best available qubits given today's noise profile.

The compiled circuit can be **2-10× larger** than the abstract one for non-trivial connectivity, especially for chemistry-style ansätze that want all-to-all entanglement on a device with linear or grid connectivity.

The VQA Compilation Subtlety

In a normal quantum algorithm, you compile once, run many times. In a VQA, the *parameters* change between runs but the *structure* doesn't — so a smart VQA compiler compiles the **structure** once and only re-evaluates the parameter angles each iteration.

Modern toolchains (Qiskit, Pennylane, Cirq) all support this "parameterized compilation" pattern: build a circuit template with symbolic parameters, compile to native gates symbolically, then bind concrete parameter values per call. Without this, every optimizer step would pay full compilation cost — potentially seconds — and the inner loop would be unworkable.

This is one of those infrastructure details that doesn't show up in any paper but determines whether a 1000-iteration VQA finishes in an hour or a week.

Why This Matters For Optimizer Choice

Some optimizers ask for many evaluations per iteration (population methods, finite-difference gradient estimation): these are bottlenecked by per-evaluation cost. Others ask for few high-quality evaluations: these are bottlenecked by per-evaluation accuracy.

If your compiled circuit is large enough that every shot is expensive (e.g., it takes 100 ms of wall clock and consumes a chunk of your queue allocation), then **the right optimizer is one that uses few evaluations** — Bayesian optimization, surrogate methods, gradient methods with parameter-shift rather than finite differences.

If your circuit is small and cheap, you can afford a population method or a high-shot-budget gradient method.

The compiled circuit is the constraint that makes this tradeoff concrete. **You can't reason about VQA optimization in isolation from compilation cost** — it sets the per-iteration budget that all of the algorithmic choices have to fit inside.

Structural Role

This is the **operational reality check** node of Module 2. It connects the abstract ansatz definitions of 2.1-2.3 to the physical hardware that nodes 2.7-2.8 will describe in detail. Without this node, the rest of Module 2 reads as pure mathematics; with it, the design tradeoffs for the rest of the course have material weight.

It also feeds Module 9 (hardware noise models): noise rates and gate counts compose into the effective objective function noise profile that the optimizer faces.

Relations to Other Concepts

- **Quantum compilation** — the general subfield. Major tools: Qiskit transpiler, Pennylane transforms, Cirq optimizers, t|ket⟩.
- **Topological compilation** — fault-tolerant compilation for surface codes; not relevant in NISQ.
- **Gate teleportation / measurement-based computation** — alternative compilation models; not yet widespread.
- **Pulse-level compilation** — bypassing gate-level abstractions to compose at the analog pulse level; under active development.

Worked Example — UCCSD on a Linear Topology

Consider a 4-qubit H_2 UCCSD circuit. Abstract form: 1 Pauli rotation $e^{i\theta P}$ where P is an 8-Pauli-string sum acting on all 4 qubits.

After abstract decomposition: - 8 Pauli rotations (one per term in P, Trotterized to first order), - Each Pauli rotation = (basis change) + (CNOT staircase, length 3) + (R_Z) + (inverse staircase) + (inverse basis change), - ~7 CNOTs per term × 8 terms = ~**56 CNOTs**.

On a device with **all-to-all connectivity** (e.g., trapped ion), the compiled circuit is roughly that size. On a device with **linear connectivity** (qubits in a chain, 1-2-3-4), CNOTs between non-adjacent qubits need SWAPs: - A CNOT between qubits 1 and 4 costs 3 + 1 + 3 = 7 native CNOTs (SWAP across, do the gate, SWAP back). - If the staircase requires multiple non-adjacent CNOTs, the SWAP overhead compounds. - Compiled CNOT count: ~**120-180**, depending on routing.

So the same abstract UCCSD ansatz costs 2-3× more on a linear device than on a fully-connected one. This is the operational reason that trapped-ion devices are sometimes more competitive for chemistry than larger but less-connected superconducting devices, even at lower qubit count.

Visualization

Pending. Two-panel diagram. Left: an abstract circuit (4 qubits, a few gates, clean). Right: the same circuit after compilation onto a linear-topology device — many SWAPs inserted, longer overall depth. Caption: "Connectivity is a tax."

Exercises

1. Estimate the compiled depth and CNOT count for a 6-qubit HEA with 3 layers on a square 2×3 grid topology, assuming each layer's entangling pattern is a complete bipartite graph.
2. For a CNOT error rate of $p = 5 \times 10^{-3}$, what is the maximum CNOT count for which the circuit fidelity remains above 0.5?
3. (VQA) The "compile once, bind many" pattern is critical for VQA performance. Sketch a minimal API for parameterized compilation that supports gradient evaluation via the parameter-shift rule without recompilation.

Research Extensions

- **Approximate compilation** — accept some compilation error in exchange for shorter circuits; relevant for ansatz-level approximations.
- **Hardware-aware ansatz synthesis** — automatically construct an ansatz given a target Hamiltonian and a device connectivity graph.
- **Learned compilation** — RL-based compilers that improve on heuristic SWAP insertion.
- **Pulse-level VQAs** — bypass gate-level decomposition by directly optimizing pulse parameters.

Cross-links to Other Courses

- **Compilers (CS)** — instruction selection, register allocation, scheduling — the same algorithmic problems with quantum-specific cost models.
- **Module 9 (Hardware Noise Models)** — gate count maps to noise budget.
- **Quantum Hardware** — coherence times, native gate sets, connectivity graphs.
- **Operations Research** — SWAP minimization is a hard combinatorial problem.

Hamiltonian Encoding

Position in Knowledge Graph

System: Variational Quantum Algorithms **Structural Domain:** Parameterized Circuits **Node:** 2.5

The Hamiltonian H is the **definition of the problem**. But "the Hamiltonian" lives in different mathematical worlds depending on where you are in the pipeline:

- The chemist writes H in second-quantized form, using fermionic creation/annihilation operators.
- The condensed matter physicist writes H in spin variables, using Pauli operators directly.
- The combinatorial optimizer writes H as an Ising or QUBO cost.
- The VQA practitioner needs H as a **sum of Pauli strings on qubits**, because that is what the device can measure.

This node is about the **encoding step** — the translation from problem-native representation to a Pauli decomposition that a quantum device can act on. It is conceptually distinct from the ansatz (which prepares a state) and conceptually distinct from the optimizer (which acts on a scalar). The encoding is the third design choice that determines what the optimizer sees.

The Required Output

What every VQA needs is a representation of H of the form

$$H = \sum_{j=1}^{M} h_j P_j,$$

where each $h_j \in \mathbb{R}$ is a coefficient and each P_j is a tensor product of single-qubit Pauli operators (and identities) on n qubits — i.e., a "Pauli string."

This form is required because it is the only thing a quantum device can directly measure. Each Pauli string can be measured by rotating each qubit into the appropriate basis (Z for I/Z entries, H rotation for X entries, S†H rotation for Y entries) and reading out in the computational basis. The measured eigenvalue is the product of ± 1 outcomes per qubit.

Once H is in this form, the cost function

$$C(\boldsymbol{\theta}) = \langle \psi(\boldsymbol{\theta}) | H | \psi(\boldsymbol{\theta}) \rangle = \sum_j h_j \langle \psi(\boldsymbol{\theta}) | P_j | \psi(\boldsymbol{\theta}) \rangle$$

becomes a weighted sum of Pauli expectation values, each of which can be estimated by repeated measurement (see node 2.7).

The encoding step is therefore: **rewrite the problem Hamiltonian, in whatever form it started, as a Pauli sum.**

Encoding Spin Hamiltonians

For spin systems, the encoding is essentially trivial — the Hamiltonian is **already** a Pauli sum.

The transverse-field Ising model on n sites:

$$H = -J \sum_{\langle i,j \rangle} Z_i Z_j - h \sum_i X_i.$$

This is already in the required form. $M = |\text{edges}| + n$ terms, each a 1- or 2-local Pauli string.

The Heisenberg model:

$$H = J \sum_{\langle i,j \rangle} (X_i X_j + Y_i Y_j + Z_i Z_j).$$

Also already in the required form, with $M = 3|\text{edges}|$ terms.

For these problem classes, encoding is no work and the Pauli sum is short — $M = O(n)$. The qubit count equals the spin count.

Encoding Combinatorial Optimization

For QUBO problems (Quadratic Unconstrained Binary Optimization), the cost function has the form

$$f(\mathbf{x}) = \sum_i a_i x_i + \sum_{i<j} b_{ij} x_i x_j, \quad x_i \in \{0, 1\}.$$

To encode this as a quantum Hamiltonian, substitute $x_i = (1 - Z_i)/2$ so that $x_i = 0 \leftrightarrow Z_i = +1$ and $x_i = 1 \leftrightarrow Z_i = -1$. Expanding gives an Ising-like Hamiltonian:

$$H = c_0 + \sum_i c_i Z_i + \sum_{i<j} c_{ij} Z_i Z_j.$$

This is already a Pauli sum. $M = O(n^2)$ for dense problems, much less for sparse ones. The qubit count equals the number of binary variables. QAOA uses exactly this encoding.

Encoding Fermionic Hamiltonians

This is where the encoding step earns its name. A second-quantized fermionic Hamiltonian for an electronic structure problem looks like

$$H = \sum_{pq} h_{pq} a_p^\dagger a_q + \frac{1}{2} \sum_{pqrs} h_{pqrs} a_p^\dagger a_q^\dagger a_r a_s,$$

where $a_p^\dagger, a_q$ are fermionic creation/annihilation operators and the coefficients come from one- and two-electron integrals over molecular orbitals.

Fermions are not the same as qubits — the operators $a_p^\dagger, a_q$ **anticommute**, while qubit operators on different qubits **commute**. Translating between these algebras is the work of a **fermion-to-qubit mapping**.

There are three standard mappings.

Jordan-Wigner

Map each spin-orbital to one qubit. Define:

$$a_p^\dagger = \frac{1}{2}(X_p - iY_p) \otimes Z_{p-1} \otimes Z_{p-2} \otimes \cdots \otimes Z_0.$$

The trailing Z chain is what enforces fermionic anticommutation: each fermion operator carries a "parity string" through all earlier orbitals. This is the simplest mapping, conceptually: **one fermion per qubit**. Its weakness is that the Pauli strings are *long* — operators acting on widely separated orbitals produce Pauli strings spanning all qubits in between. For sparse fermionic Hamiltonians, JW gives long strings.

Bravyi-Kitaev

Use a more clever data structure (a binary tree of qubits) to encode parity information. Each fermion operator becomes a Pauli string of length $O(\log n)$ instead of $O(n)$. Trade-off: the implementation is more complex, and the resulting Pauli strings are *shorter on average* but spread across non-adjacent qubits, which can hurt on devices with local connectivity.

Parity Encoding

Encode the *partial sums of occupation numbers* rather than the occupations themselves. Useful in some contexts, less common in practice than JW or BK.

The choice between JW and BK matters in practice because it determines:

- Number of Pauli terms in the Hamiltonian (typically $O(n^4)$ for either, but with different prefactors).
- Locality of the Pauli strings (JW: linear; BK: $\log n$).
- Which strings can be measured simultaneously (see node 2.8 on measurement grouping).

Standard libraries (OpenFermion, Qiskit Nature) implement both and let you switch with a single function call.

Symmetry Reductions

After mapping to qubits, you can often **further reduce the qubit count** by exploiting symmetries.

For H_2 in a minimal STO-3G basis: - Naive Jordan-Wigner: 4 spin-orbitals $\to$ 4 qubits. - After exploiting parity (Z2 symmetries arising from spin and electron number conservation): **2 qubits**.

This is "qubit tapering" and is implemented in standard libraries. It reduces both the device cost and the parameter count of the ansatz, often by a factor of 2-4. For larger molecules the savings can be larger.

The cost is conceptual: the resulting 2-qubit Hamiltonian no longer has a clean physical interpretation as "electrons in orbitals." It's just a Pauli sum.

What The Optimizer Sees After Encoding

Once H is encoded, the optimizer sees a single scalar function $C(\boldsymbol{\theta})$ — *the encoding has disappeared.* The choice of mapping, the symmetry reductions, the qubit count: all of these are "compiled away" from the optimizer's perspective.

But the encoding choice has determined:

- **The number of Pauli terms**, which sets the per-evaluation shot cost (one shot budget per term, see node 2.8).
- **The variance structure** of the cost estimator, since variance depends on which Pauli terms are easy to measure jointly.
- **The qubit count**, which sets the depth budget for the ansatz.
- **The barren plateau profile**, indirectly, since fewer qubits means smaller Hilbert space and milder concentration.

So the encoding is invisible to the optimizer but it shapes everything the optimizer experiences. This is one of the places where Module 1's "stochastic objective generator" abstraction (1.8) is **leaky** — the noise structure of the generator is set by encoding decisions made elsewhere, and ignoring those decisions misses a major lever.

Structural Role

This node closes out the "what is the problem definition?" half of Module 2. Together with 2.1-2.3 (ansatz design) it answers: *what does the optimizer minimize, and over what?*

Forwards: - 2.7 (how measurement actually works) consumes Pauli strings as input. - 2.8 (measurement grouping and shot budget) consumes the *number* and *commutation structure* of the Pauli strings. - Module 9 (hardware noise models) consumes encoded Hamiltonians to study how noise propagates into objective error.

Relations to Other Concepts

- **Second quantization** — the standard formalism for many-body fermionic and bosonic systems.
- **Jordan-Wigner transformation** — historically the first fermion-to-qubit map, dating to 1928.
- **Bravyi-Kitaev transformation** — modern, more efficient mapping using parity trees.
- **Z2 symmetries / qubit tapering** — exploit conserved quantities to reduce qubit count.
- **Block encoding** — a different paradigm: encode H as a unitary block within a larger circuit. Used in fault-tolerant algorithms (qubitization), not VQAs.

Worked Example — H_2 End-to-End

Starting representation: second-quantized fermionic Hamiltonian for H_2 in STO-3G basis at 0.74 Å bond length.

After computing one- and two-electron integrals (PySCF or similar), the fermionic Hamiltonian has ~30 nonzero coefficients across ~8 distinct fermionic operators.

After Jordan-Wigner mapping to 4 qubits, this becomes a sum of **15 Pauli strings**:

$$H = -1.05 \cdot I - 0.39 \cdot Z_0 + 0.39 \cdot Z_1 - 0.01 \cdot Z_2 + \cdots + 0.18 \cdot X_0X_1Y_2Y_3 + \ldots$$

(approximate coefficients).

After Z2 symmetry tapering: **5 Pauli strings on 2 qubits**.

The optimizer doesn't know any of this. It sees a function from $\mathbb{R}^P$ to $\mathbb{R}$ and asks for evaluations. But each evaluation costs at least 5 measurement bases $\times$ N shots each. And the optimal ansatz on 2 qubits is much smaller than the optimal ansatz on 4 qubits. **The encoding choices set the cost basis and the design space.**

Visualization

Pending. A flow diagram. Top: "Fermionic Hamiltonian (8 distinct operators)". Middle: "Jordan-Wigner (15 Pauli strings, 4 qubits)". Bottom: "Symmetry-tapered (5 Pauli strings, 2 qubits)". Side annotations: "this affects ansatz design", "this affects measurement cost", "this determines what the optimizer sees".

Exercises

1. Apply the Jordan-Wigner transformation to the fermionic operator $a_2^\dagger a_0$ for a 4-orbital system. Write out the resulting Pauli string explicitly.
2. The Jordan-Wigner mapping gives $O(n)$-length Pauli strings; Bravyi-Kitaev gives $O(\log n)$. Estimate the asymptotic crossover qubit count where BK becomes preferable for measurement cost. (Hint: depends on the connectivity of the device and the dominant cost.)
3. (VQA) For a 2-qubit Pauli sum $H = c_0 II + c_1 ZI + c_2 IZ + c_3 ZZ + c_4 XX$, how many distinct measurement bases are needed to estimate $\langle H \rangle$? Which terms can be measured simultaneously?

Research Extensions

- **Custom fermion-to-qubit mappings** — tailoring the mapping to a specific device topology to minimize compiled circuit depth.
- **Locality-preserving mappings** — encoding Hamiltonians so that nearby orbitals map to nearby qubits.
- **Tapering automation** — automatic discovery of Z2 symmetries in arbitrary fermionic Hamiltonians.
- **Bosonic encodings** — encoding bosonic systems (vibrational modes, photonic networks) into qubits with bounded truncation error.

Cross-links to Other Courses

- **Quantum Chemistry** — second quantization, integrals, basis sets.
- **Group Theory** — Z2 symmetries, conserved quantities.
- **Module 8** — encoding choice affects expressivity.
- **Module 9** — encoding choice affects noise propagation.

Representations of Quantum States

Position in Knowledge Graph

System: Variational Quantum Algorithms **Structural Domain:** Parameterized Circuits **Node:** 2.6

When you simulate, debug, or analyze a VQA before running it on real hardware, you have to choose a **representation** for the quantum state. The same state has many possible mathematical descriptions, and each representation has a different cost-vs-fidelity tradeoff.

This node walks through the four representations that matter for VQA work:

1. **Statevector** — full amplitude vector
2. **Product state** — restricted to factorizable states
3. **Density matrix** — for mixed states and noise
4. **Matrix product state (MPS)** — for low-entanglement states

A working VQA practitioner uses all four at different points in the pipeline. Knowing when to use which is one of the most practically useful skills in the field, and it is rarely taught explicitly.

---

1. Statevector

The most direct representation: a complex vector

$$|\psi\rangle = \sum_{x\in\{0,1\}^n} \psi_x |x\rangle, \quad \psi_x \in \mathbb{C}, \quad \sum_x |\psi_x|^2 = 1.$$

For n qubits, this is a vector of 2^n complex numbers. Memory cost: $2^n \times 16$ bytes (double-precision complex) $= 16$ GB at $n = 30$, 16 TB at $n = 40$. Hard ceiling for personal computers around $n = 28 - 30$; for serious clusters, around $n = 40 - 45$.

Operations are matrix-vector multiplies. Applying a single-qubit gate is $O(2^n)$. Applying an n-qubit unitary directly is $O(4^n)$, though structured unitaries (a circuit of single- and two-qubit gates) can be applied gate-by-gate at $O(2^n)$ per gate.

Use for: debugging small instances, computing exact expectation values, sanity-checking optimizer behavior on toy problems where the ground truth is known.

Don't use for: anything past $n = 30$, or for simulating noise (no room for noise channels in a pure state).

The thesis's experimental setup uses 8-qubit statevector simulation for the VQE benchmarks. At 8 qubits, the statevector is 4 KB and operations are essentially free.

---

2. Product State (Tensor Product of Single-Qubit States)

Restricted to states of the form

$$|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_n\rangle.$$

Memory cost: $2n$ complex numbers (2 amplitudes per qubit). Linear in qubit count.

This representation is exact only for **non-entangled** states. For entangled states it is a strict approximation.

Use for: Hartree-Fock-style starting states (which *are* product states in the right basis), mean-field approximations, sanity-checking that an ansatz can break out of the product subspace.

Don't use for: any state past the first entangling layer of an ansatz. As soon as a CNOT runs, the state generally leaves the product subspace.

The product-state representation matters for VQAs mostly as a **starting point** or **classical baseline**, not as a working representation.

---

3. Density Matrix

For mixed states (statistical ensembles or open-system states), the right representation is

$$\rho = \sum_k p_k |\psi_k\rangle\langle\psi_k|, \quad p_k \geq 0, \quad \sum_k p_k = 1.$$

Memory cost: $2^n \times 2^n$ complex numbers $= 4^n \times 16$ bytes $= 16$ GB at $n = 15$, 16 TB at $n = 20$. Half the qubit ceiling of statevector, because the matrix has square as many entries.

Operations: unitary application is $\rho \to U\rho U^\dagger$, an $O(4^n)$ operation per gate. Noise channels are applied as Kraus sums, which is more general than statevector simulation can express.

Use for: noise simulation (mixed states are the natural language of decoherence), modeling realistic hardware behavior, computing fidelity to a target state.

Don't use for: problems past $n = 18$ or so, or when a pure-state representation suffices.

For the thesis's noise-aware simulations, density matrix representation is the workhorse.

---

4. Matrix Product State (MPS)

A clever approximation: write the amplitude tensor as a product of small matrices.

$$\psi_{x_1 x_2 \ldots x_n} = A^{(1)}_{x_1} A^{(2)}_{x_2} \cdots A^{(n)}_{x_n},$$

where each $A^{(i)}_{x_i}$ is a matrix of size $\chi_{i-1} \times \chi_i$ and χ_i is the **bond dimension** at site i.

For an exact representation of an arbitrary n-qubit state, you need bond dimension up to $2^{n/2}$ — exponential. But for **low-entanglement states**, a small bond dimension ($\chi = 16, 64, 256$) gives an excellent approximation.

The maximum half-chain entanglement entropy that an MPS with bond dimension χ can capture is $\log_2 \chi$. States obeying the **area law** for entanglement (most ground states of local Hamiltonians) can be efficiently represented this way; states with **volume-law entanglement** (random circuits past a certain depth, scrambled states) cannot.

Memory cost: $O(n\chi^2)$ — linear in qubit count, polynomial in bond dimension. Operations: $O(n\chi^3)$ per gate. Tractable for n up to many hundreds when χ is small.

Use for: simulating large 1D-like quantum systems, classical pre-optimization of ansätze, studying how entanglement grows with depth.

Don't use for: problems with strong long-range entanglement, deep random circuits, or states that genuinely live near the volume-law regime.

MPS simulation is what makes "100-qubit VQE" classically tractable for shallow ansätze on chemistry-style problems — and what makes "100-qubit barren plateau study" intractable.

How To Choose

A practical decision tree:

- **Are you debugging a small instance?** → Statevector.
- **Are you studying noise effects?** → Density matrix.
- **Is the state shallow / low-entanglement?** → MPS.
- **Are you computing a Hartree-Fock baseline?** → Product state.
- **Do you need exact answers and n is small enough?** → Statevector.
- **Is n large but the ansatz is shallow?** → MPS, then validate with statevector on a smaller instance.
- **Is the state highly entangled and n is large?** → You're in trouble. This is the regime where classical simulation fails and you actually need the quantum hardware.

The last point is the **quantum advantage frontier**: VQAs are interesting precisely when none of the classical representations can handle the state cheaply.

What The Hardware Returns

None of these representations is what the hardware itself stores. The hardware stores a physical quantum state, and you can only query it through measurement (node 2.7). You **never see** the statevector or density matrix from a real device — you see samples from a distribution induced by it.

This means:

- Debugging on a real device is much harder than debugging in simulation, because you can't inspect intermediate states.
- Comparing simulation to hardware requires statistically reconciling measurement outcomes with simulated probabilities.
- Process tomography (reconstructing the full state) is expensive ($O(4^n)$ measurements) and only practical for very small n.

In practice, VQA development happens in three stages:

1. **Statevector simulation** — verify the ansatz is correct, the optimizer works, the cost converges.
2. **Density matrix simulation** — add a noise model, see how noise affects convergence.
3. **Real hardware** — submit to actual device, deal with the gap between modeled and real noise.

Skipping any stage is asking for trouble.

Structural Role

This node is the **practitioner's toolbox** node of Module 2. It is mostly free of authored opinions and consists of standard background that any VQA worker needs. Forwards to 2.7 (measurement) which describes how the hardware exposes its state, and to 2.9 (ansatz as manifold) which uses the statevector representation to define the geometric picture.

It also feeds Module 4 (barren plateaus): the proofs of the barren plateau theorem use random unitary averages over the statevector representation, and being able to compute these averages depends on being able to write down the state.

Relations to Other Concepts

- **Tensor network states** — the broader family that MPS belongs to (PEPS for 2D, MERA for hierarchical, tree tensor networks for tree-like).

- **Stabilizer formalism** — a different polynomial classical representation, exact for Clifford circuits but unable to represent general states.
- **Neural network quantum states** — encode amplitudes as a neural network; can sometimes outperform MPS for non-1D structure.
- **Decoherence-free subspaces** — restricted state subspaces immune to particular noise types; useful for representation choice when noise structure is known.

Worked Example — Memory For 30 Qubits

How much memory does a 30-qubit state cost in each representation?

- **Statevector:** $2^{30} \times 16$ bytes = **16 GB**. Fits in a high-memory workstation.
- **Density matrix:** $2^{60} \times 16$ bytes = **16 EB**. Completely infeasible.
- **Product state:** 60 complex numbers = **480 bytes**. Trivial, but only correct for unentangled states.
- **MPS, bond dimension 64:** $30 \times 64^2 \times 16$ bytes $\approx$ **2 MB**. Trivial. But only correct if half-chain entanglement entropy is ≤ 6.
- **MPS, bond dimension 1024:** $30 \times 1024^2 \times 16$ bytes $\approx$ **500 MB**. Workable. Captures entanglement entropy up to 10.

Lesson: the order-of-magnitude difference between representations is larger than the order-of-magnitude difference between problem sizes. **Choosing the right representation is more important than buying more RAM.**

Visualization

Pending. A 4-quadrant plot. X-axis: qubit count. Y-axis: log memory cost. Four lines, one per representation, with characteristic crossovers and ceilings. Annotations on each line indicating "use this region."

Exercises

1. Show that a Bell state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ cannot be written as a product state $|\psi_1\rangle \otimes |\psi_2\rangle$ for any choice of $|\psi_1\rangle, |\psi_2\rangle$.
2. What is the bond dimension required to represent the Bell state as an MPS? What about the GHZ state $\frac{1}{\sqrt{2}}(|00\ldots0\rangle + |11\ldots1\rangle)$ on n qubits?
3. (VQA) For a hardware-efficient ansatz with L layers on n qubits, give a heuristic argument for the minimum bond dimension needed to represent the resulting state to fixed accuracy as a function of L and n.

Research Extensions

- **Tree tensor networks for VQA** — alternative to MPS, sometimes better for branched problem structures.
- **Neural quantum states for VQA simulation** — using NQS as a classical baseline against quantum hardware.
- **Approximate density matrix methods** — low-rank density matrix representations for noisy state simulation at higher qubit count.

Cross-links to Other Courses

- **Quantum Information** — pure vs mixed states, density matrix formalism, partial traces.
- **Tensor Networks (separate course)** — MPS, PEPS, MERA, contractions.
- **Numerical Methods** — large-vector arithmetic, sparse matrix tricks.
- **Statistics Module 7** — high-dimensional state representations and their information geometry.

How Measurement Actually Works in a VQA

Position in Knowledge Graph

System: Variational Quantum Algorithms **Structural Domain:** Parameterized Circuits **Node:** 2.7

This node is for the reader who has seen $\langle\psi(\boldsymbol{\theta})|H|\psi(\boldsymbol{\theta})\rangle$ written down a hundred times and is finally ready to ask: **what physically happens, on a real device, in order to produce that scalar?**

The mechanics are not glamorous. They involve a lot of basis changes, a lot of computational-basis readouts, a lot of bit-counting, and a lot of statistical aggregation. But understanding them concretely is what separates a practitioner from a paper-reader, and it is the *why* behind several key facts that show up later in the course (variance scaling, measurement grouping, shot noise, hardware bias).

The Goal

We want to estimate

$$C(\boldsymbol{\theta}) = \langle\psi(\boldsymbol{\theta})|H|\psi(\boldsymbol{\theta})\rangle, \quad H = \sum_{j=1}^{M} h_j P_j.$$

The strategy: estimate each Pauli expectation $\langle P_j\rangle$ separately, then take the linear combination. This works because expectation is linear: $\langle H\rangle = \sum_j h_j \langle P_j\rangle$.

So the entire problem reduces to: **how do you measure a Pauli string P on a state prepared by a circuit?**

Measuring a Single Pauli String: The Recipe

For a Pauli string like $P = X_0 \otimes Z_1 \otimes Y_2 \otimes I_3$ on 4 qubits, the recipe is:

Step 1: Prepare $|\psi\rangle$. Run the ansatz circuit $U(\boldsymbol{\theta})$ on the device, starting from $|0\rangle^{\otimes n}$.

Step 2: Rotate each non-Z, non-I qubit into the Z basis. - For each X entry in P: apply a Hadamard gate H to that qubit. - For each Y entry in P: apply $S^\dagger H$ to that qubit (equivalently $R_X(\pi/2)$). - For each Z entry: do nothing — it's already in the Z basis. - For each I entry: do nothing — measurement of identity is trivial (always $+1$).

After this rotation, the Pauli string has been mapped to a string of Z's and I's, and measuring it in the **computational basis** (which is the Z basis on every qubit) gives the eigenvalue of P.

Step 3: Read out every qubit in the computational basis. The hardware projects each qubit onto $|0\rangle$ or $|1\rangle$, giving a bitstring $b \in \{0,1\}^n$.

Step 4: Compute the eigenvalue from the bitstring. For each qubit position where P has a non-identity entry (X, Y, or Z after rotation, all now Z), the eigenvalue contribution is $(-1)^{b_i}$ ($+1$ for outcome 0, -1 for outcome 1). The full Pauli eigenvalue is the product over all non-identity positions:

$$\text{eigenvalue}(b) = \prod_{i \in \text{support}(P)} (-1)^{b_i}.$$

This is ± 1.

Step 5: Repeat N times and average. Run the circuit N times — each is a "shot" — and average the eigenvalues:

$$\widehat{\langle P \rangle} = \frac{1}{N} \sum_{k=1}^{N} \text{eigenvalue}(b^{(k)}).$$

This is your estimator. It is unbiased: $\mathbb{E}[\widehat{\langle P \rangle}] = \langle P \rangle$. Its variance is bounded by $1/N$ (since each sample is in $[-1, 1]$).

What This Looks Like For The Whole Hamiltonian

Multiplying the protocol across all M Pauli strings:

```
For each Pauli string P_j in H:
    For shot = 1 to N_j:
        Prepare |psi(theta)>
        Rotate to P_j's measurement basis
        Read out the bitstring
        Compute the eigenvalue
    Average to get hat<P_j>

Final cost estimate = sum_j h_j * hat<P_j>
```

Per cost evaluation: $M \times N$ total circuit runs, where M is the number of Pauli strings and N is the per-string shot count. For H_2 (5 strings after tapering), with $N = 1000$ shots each, that's 5000 circuit executions per cost evaluation.

Each evaluation takes wall-clock time on the order of seconds to minutes on a real device, depending on queue depth and circuit length.

Variance and the $1/\sqrt{N}$ Scaling

The estimator $\widehat{\langle P \rangle}$ is a sample mean of ± 1 random variables. By standard statistics:

$$\text{Var}[\widehat{\langle P \rangle}] = \frac{1 - \langle P \rangle^2}{N}, \quad \text{StdDev}[\widehat{\langle P \rangle}] = \sqrt{\frac{1 - \langle P \rangle^2}{N}} \leq \frac{1}{\sqrt{N}}.$$

The standard deviation shrinks as $1/\sqrt{N}$. **This is the fundamental scaling that bounds VQA cost evaluation accuracy.** To halve the error, quadruple the shots.

For the full Hamiltonian, the variance is

$$\text{Var}[\widehat{\langle H \rangle}] = \sum_j \frac{h_j^2 (1 - \langle P_j \rangle^2)}{N_j} \leq \sum_j \frac{h_j^2}{N_j}.$$

If you spend the same N shots on each term, the total cost-estimator variance is bounded by $\frac{1}{N} \sum_j h_j^2$. For molecules, $\sum_j h_j^2$ can be on the order of 1-100 hartree^2, which means even moderate accuracy (millihartree) requires hundreds of thousands of shots per evaluation.

This is a very tight constraint, and it is the reason **measurement grouping** (node 2.8) is so important — it is the only way to amortize measurements across multiple Pauli terms and bring the per-evaluation shot cost into the workable range.

What Hardware Adds On Top

The recipe above assumes ideal hardware. On a real device, several extra effects show up:

1. Gate errors during state preparation. Each gate in $U(\boldsymbol{\theta})$ has some error rate p_g. After 100 gates with $p_g = 10^{-3}$, the prepared state has fidelity $\approx e^{-0.1} \approx 0.9$ to the ideal state. The measured expectation values are biased toward those of the noisy state.

2. Readout error. The "measure in the Z basis" step itself has an error rate, typically 1-5%. A qubit that is actually in $|0\rangle$ is reported as 1 with probability p_{readout}, and vice versa. This adds bias to all expectation values.

3. Crosstalk. Measuring one qubit can subtly disturb the others, creating correlations between supposedly independent measurements.

4. Calibration drift. Gate parameters and readout fidelities change over time. A measurement made at the start of a queue may have slightly different bias than one made an hour later.

All of these turn the ideal recipe into a noisy approximation. The key consequence: **the estimator is biased, not just noisy.** The bias depends on the hardware and changes over time. Unbiased estimation via post-processing (readout error mitigation, zero-noise extrapolation, probabilistic error cancellation) is an active research area.

What The Optimizer Sees

After all of this, the optimizer is handed a single number per evaluation: $\widehat{C}(\boldsymbol{\theta}_k)$. This number is

- biased (by hardware noise and possibly by mitigation procedures),
- noisy (by shot statistics and time-correlated drift),
- delayed (by queue and execution time),
- expensive (each evaluation consumed $M \times N$ circuit runs).

This is the operational meaning of the "stochastic objective generator" abstraction from node 1.8. The abstraction is correct, but **the numbers behind it come from a long pipeline that you have to design to control variance and bias**.

The thesis's experimental setup on simulated hardware fixes the shot count at $N = 1024$ per term and runs for tens of thousands of total circuit executions per optimization run. On real hardware, the equivalent can take hours to a day.

Structural Role

This is the **operational unmasking** node of Module 2. It exposes what $\widehat{C}(\boldsymbol{\theta})$ actually means in terms of physical operations, and connects the abstract Pauli decomposition (2.5) to the noisy scalar the optimizer sees.

Forwards to 2.8 (measurement grouping), which is the optimization step that comes immediately after this raw recipe.

Relations to Other Concepts

- **Quantum measurement (Born rule)** — the underlying physics. $P(b|\psi) = |\langle b|\psi\rangle|^2$.
- **POVMs (positive operator-valued measures)** — the most general measurement formalism; the protocol above is a special case.
- **Direct fidelity estimation** — protocols for estimating state fidelity using random Pauli measurements.

- **Classical shadows** — efficient protocols that estimate many observables from a single measurement set; an active alternative to per-Pauli measurement.
- **Readout error mitigation** — post-processing to remove bias from readout errors.

Worked Example — Measuring $\langle X_0 Z_1 \rangle$ on a Bell State

Suppose $|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ and we want to estimate $\langle X_0 Z_1 \rangle$.

Step 1: prepare the Bell state via $H_0, \mathrm{CNOT}_{0,1}$ on $|00\rangle$.

Step 2: rotate qubit 0 (X position) into Z basis by applying H. Qubit 1 (Z position) needs no rotation.

The state after rotation is:

$$H_0|\psi\rangle = H_0 \cdot \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = \frac{1}{\sqrt{2}}(|+0\rangle + |-1\rangle) = \frac{1}{2}(|00\rangle + |10\rangle + |01\rangle - |11\rangle).$$

Step 3: measure both qubits in computational basis. Each outcome (b_0, b_1) occurs with probability $|\text{amplitude}|^2 = 1/4$.

Step 4: the eigenvalue is $(-1)^{b_0}(-1)^{b_1}$. - $00 \to +1$ (prob 1/4) - $10 \to -1$ (prob 1/4) - $01 \to -1$ (prob 1/4) - $11 \to +1$ (prob 1/4)

Expectation: $\frac{1}{4}(+1 - 1 - 1 + 1) = 0$.

So $\langle X_0 Z_1 \rangle = 0$ for the Bell state. (Sanity check: this is correct — the Bell state has zero expectation for any Pauli string except the four stabilizers $II, ZZ, XX, -YY$.)

Step 5: repeat N times. The sample mean converges to 0 with variance $1/N$.

Visualization

Pending. A circuit diagram with three layers labeled: "(1) state prep — $U(\boldsymbol{\theta})$", "(2) basis change — H, S†H per Pauli letter", "(3) Z-basis readout". A bracket below indicating "this entire circuit runs N times". A separate annotation on the right showing the bitstring → eigenvalue computation.

Exercises

1. For the state $|\psi\rangle = R_Y(\theta)|0\rangle$, describe the measurement protocol for estimating $\langle X \rangle$ and $\langle Z \rangle$. Compute the variance of each estimator at $\theta = \pi/2$ for $N = 100$ shots.
2. Show that $\mathrm{Var}[\widehat{\langle P \rangle}] = (1 - \langle P \rangle^2)/N$. Where in the derivation does the boundedness of the eigenvalues to ± 1 enter?
3. (VQA) The naive cost-estimator variance bound is $\sum_j h_j^2/N$. For a 4-qubit H_2 Hamiltonian with 15 Pauli terms, estimate the shot count per term needed to achieve millihartree accuracy in the cost estimator.

Research Extensions

- **Classical shadows** — measure random unitaries, estimate many observables at once; logarithmic shot scaling in some regimes.
- **Adaptive shot allocation** — dynamically reallocate shots between Pauli terms based on observed variance.

- **Importance-weighted sampling** — query terms with larger $|h_j|$ more often.
- **Bayesian estimators** — use prior knowledge about $\langle P_j \rangle$ to improve early-iteration accuracy.

Cross-links to Other Courses

- **Quantum Measurement** — Born rule, projective measurements, POVMs.
- **Statistics Module 3** — sample mean, variance of estimators, central limit theorem.
- **Module 8** (measurement grouping) — the immediate next step after this node.
- **Module 9** (hardware noise models) — what gets layered on top in practice.

Measurement Grouping and Shot Budget

Position in Knowledge Graph

System: Variational Quantum Algorithms **Structural Domain:** Parameterized Circuits **Node:** 2.8

The naive measurement protocol from node 2.7 — measure each Pauli term in the Hamiltonian separately, with N shots each — is operationally feasible but **wasteful**. For a Hamiltonian with M Pauli terms, it costs $M \times N$ circuit runs per cost evaluation. For chemistry-style Hamiltonians where M can be in the hundreds or thousands, this is prohibitive.

The remedy is **measurement grouping**: identify Pauli terms that can be measured *simultaneously* in the same circuit run, group them, and amortize the shot cost across the group. Doing this well can reduce the per-evaluation cost by an order of magnitude.

This node explains the grouping problem, the standard solutions, and how a finite shot budget gets allocated across groups in practice.

When Two Pauli Strings Can Be Measured Simultaneously

Two Pauli strings can be **measured in the same experiment** if and only if they commute and there exists a single basis change that diagonalizes both. The simplest sufficient condition is **qubit-wise commutativity (QWC)**: two Pauli strings are QWC if at every qubit position, either:

- they have the same Pauli letter (XX, YY, ZZ, II), or
- at least one of them is the identity (XI, IY, ZI, II, etc.).

If two strings are QWC, then the per-qubit basis change for one is compatible with the per-qubit basis change for the other, and a single measurement in the combined basis gives you the bitstring needed to compute eigenvalues for **both**.

Example: $P_1 = X_0 Z_1$ and $P_2 = X_0 I_1$. These are QWC — both have X on qubit 0, and P_2 has identity on qubit 1. A single measurement (rotate qubit 0 to Z basis via H, leave qubit 1 alone, read out both) gives bitstrings from which you compute both $\widehat{\langle P_1 \rangle}$ (using both bits) and $\widehat{\langle P_2 \rangle}$ (using only bit 0).

QWC is the **simplest** notion of joint measurability. There are stronger notions:

- **General commutativity**: two Pauli strings commute as operators, but the diagonalizing basis change is not local (it requires entangling gates in the basis-change layer).
- **Anticommuting partitions**: with extra ancilla qubits, you can measure even mutually anticommuting sets of Paulis jointly.

The thesis's regularization techniques don't depend on grouping strategy specifically, but the shot budget allocations they assume are based on standard QWC grouping.

The Grouping Problem

Given a Hamiltonian $H = \sum_j h_j P_j$, partition the Pauli strings into the **minimum number of QWC groups**. Each group can be measured by a single circuit (with a single basis-change layer); within a group, every shot contributes to estimating every term in the group.

This is exactly the **graph coloring problem** on a "compatibility graph": vertices are Pauli strings, edges connect non-QWC pairs. A valid grouping is a coloring; the minimum grouping is the chromatic number.

Graph coloring is NP-hard in general, but for chemistry Hamiltonians:

- A greedy coloring runs in $O(M^2)$ and produces near-optimal groupings in practice.

- Specialized algorithms (Sorted Insertion, Largest-First Degree-Sorted) consistently reduce M from "thousands" to "tens" for typical chemistry Hamiltonians.

For H_2 (15 Pauli terms after Jordan-Wigner): grouping reduces this to **2 groups**, meaning a 7.5× reduction in circuit count per evaluation. For larger molecules like BeH_2: groupings of ~10-20 from raw counts of 200-500. For very large Hamiltonians (FeMoco, biomolecules): hundreds of groups from tens of thousands of terms.

The reduction is not free — within a group, terms with different supports compete for the same shots, and the variance allocation has to be done carefully — but it is a major operational win.

Variance Within a Group

Suppose a group G contains Pauli terms $P_{j_1}, P_{j_2}, \ldots, P_{j_K}$. A single measurement in the group's basis produces a bitstring from which you compute one eigenvalue per term:

$$\widehat{e}_k = \prod_{i \in \text{support}(P_{j_k})} (-1)^{b_i}, \quad k = 1, \ldots, K.$$

After N_G shots:

$$\widehat{\langle P_{j_k} \rangle} = \frac{1}{N_G} \sum_{s=1}^{N_G} \widehat{e}_k^{(s)}.$$

The crucial point: **all K estimators come from the same shots**. Their variances are still each bounded by $1/N_G$, but they are **correlated** with each other (because they share the underlying measurement outcomes).

The total variance of the cost contribution from the group is:

$$\text{Var}\left[\sum_{k=1}^{K} h_{j_k} \widehat{\langle P_{j_k} \rangle}\right] = \frac{1}{N_G} \text{Var}_{\text{shot}}\left[\sum_k h_{j_k} \widehat{e}_k\right],$$

where the inner variance is over the joint distribution of one shot's worth of measurements in the group basis.

For QWC groups, the inner variance can be substantially smaller than $\sum_k h_{j_k}^2$ (the bound for independent measurements) when the Pauli terms are correlated under the group basis. So measurement grouping does not just save circuit runs — it can also **reduce the per-shot variance contribution**.

Allocating a Shot Budget

Given a total shot budget N_{total} per cost evaluation and a set of G groups, how should you allocate shots across groups?

Two reasonable strategies:

Equal allocation. Give each group $N_G = N_{\text{total}}/G$. Simple, and works fine when the per-group variance contributions are roughly equal.

Variance-weighted allocation. Give each group a number of shots proportional to its variance contribution:

$$N_G = N_{\text{total}} \cdot \frac{\sigma_G}{\sum_{G'} \sigma_{G'}},$$

where σ_G is an estimate of the per-shot standard deviation of the group's contribution. This minimizes the total variance for a fixed total shot count and is the optimal allocation in the sense of Lagrange multipliers. It is sometimes called the "Wu et al." or "Rubin" allocation.

In practice, optimal variance-weighted allocation requires knowing variances in advance, which you don't. A common compromise: use a few hundred shots per group as a "scout" to estimate variances, then allocate the remaining budget proportionally.

The Shot Budget As An Optimization Hyperparameter

The total shot budget N_{total} per evaluation is **part of the optimization design**, not a fixed number imposed by hardware. Tradeoffs:

- **More shots** → less estimator noise → cleaner gradient signal → faster per-iteration progress.
- **Fewer shots** → cheaper evaluations → more iterations per wall-clock budget → more total parameter updates.

Most VQA optimizer choices make implicit assumptions about this tradeoff:

- **SPSA** (Simultaneous Perturbation Stochastic Approximation) tolerates very low shot counts and many iterations.
- **COBYLA** assumes moderate shot counts and uses geometry-based updates.
- **Adam with parameter-shift gradients** wants high enough shot counts that the gradient is reliable.
- **HOPSO** (the thesis's preferred optimizer) tolerates noisy evaluations and benefits from more iterations.

The thesis runs experiments at fixed $N_{\text{total}} = 1024$ shots per evaluation to enable apples-to-apples comparison across optimizers. Real-hardware deployments often use **shot scheduling** — start with low shots (cheap, noisy) for early exploration and ramp up as the optimizer converges (expensive, accurate near the minimum). See node 5.7 for the thesis-level treatment.

What Goes Wrong If You Get This Wrong

Two common failure modes:

1. Insufficient shots per term. With $N = 100$ shots and a Pauli term whose variance is near 1, the estimator standard deviation is ~0.1. For an optimizer trying to detect a gradient component of magnitude 0.05, this is not enough — the noise drowns the signal, and the optimizer wanders. Symptom: cost trajectory looks like a random walk rather than a descent.

2. Imbalanced shot allocation. If you give the 14-Pauli-term H_2 Hamiltonian 100 shots per term but the dominant term has $|h_j| = 10$ and the rest have $|h_j| < 0.1$, then the dominant term contributes $100\times$ more uncertainty to the cost estimate than the rest combined — you have wasted 90% of your shot budget on terms whose noise contribution is negligible. Symptom: variance-weighted reallocation produces a 5-10$\times$ efficiency gain.

Both failure modes are extremely common in beginner VQA work and very rarely diagnosed correctly. Watching the per-term variance contribution to the cost estimate is the single most informative debugging signal.

Structural Role

This is the **operational closure** node of Module 2. Module 2 has now described, end to end, what happens when the optimizer requests a cost evaluation: ansatz design (2.1-2.3) → compilation (2.4) → encoding (2.5) → state representation (2.6) → measurement protocol (2.7) → shot budget and grouping (this node).

Forwards: Module 5 (regularization) uses the shot budget as the resource against which improvements are measured. Module 9 (hardware noise) revisits the variance contributions when realistic noise models are layered in.

Relations to Other Concepts

- **Graph coloring** — the formal problem behind QWC grouping.
- **Importance sampling** — variance-weighted shot allocation is its discrete analogue.
- **Classical shadows** — an alternative to grouping that estimates many observables from random measurements.
- **Adaptive measurement protocols** — schemes that change basis based on previous outcomes.

Worked Example — Grouping for H_2

After Jordan-Wigner, the H_2 Hamiltonian (4 qubits) has 15 Pauli terms.

Let's enumerate a few: - $I_0I_1I_2I_3$ (identity, contributes a constant) - Z_0, Z_1, Z_2, Z_3 (single-qubit Z's) - Z_0Z_1, Z_0Z_2, Z_0Z_3, Z_1Z_2, Z_1Z_3, Z_2Z_3 (ZZ pairs) - $X_0X_1Y_2Y_3$, $X_0Y_1Y_2X_3$, $Y_0X_1X_2Y_3$, $Y_0Y_1X_2X_3$ (XXYY-type four-body)

The 11 Z-only terms are all mutually QWC (they all only have Z's and I's on every qubit) — they form **one group**, measured by reading out all 4 qubits in the Z basis.

The 4 XXYY terms are also mutually QWC: each has the same Pauli letter at each position (after checking carefully — they aren't quite, actually). In practice, they form 1-2 additional groups depending on the exact strings.

Total grouping: **2-3 groups**. From 15 separate measurements down to 2-3 — a roughly 5-7× speedup.

For a 100-shot budget per group (a generous allocation), the total cost per evaluation is 200-300 circuit runs instead of 1500. **That difference is the difference between a workable VQA inner loop and an unworkable one.**

Visualization

Pending. A graph diagram. Vertices: 15 Pauli strings labeled. Edges: connect non-QWC pairs. The vertices are colored to show a 2-coloring (the QWC grouping). Below: a bar chart showing "without grouping: 15 measurements" vs "with grouping: 2 measurements".

Exercises

1. For the Pauli strings $\{X_0Z_1, X_0Y_1, Z_0Z_1, X_0X_1\}$, determine which pairs are QWC. Find the minimum QWC grouping.
2. Show that for a Hamiltonian made entirely of Z-string terms (all Pauli operators are Z or I), every term is QWC with every other term, so the grouping reduces to a single group.
3. (VQA) The greedy QWC grouping algorithm has worst-case approximation ratio bounded by the maximum degree of the conflict graph. Sketch why this matters for chemistry Hamiltonians vs spin-model Hamiltonians.

Research Extensions

- **General-commutativity grouping** — beyond QWC, allowing entangling basis-change layers.
- **Anticommuting set measurements** — using ancilla qubits to measure mutually anticommuting Paulis.
- **Fragment-based methods** — partition the Hamiltonian into structurally meaningful fragments (one-body, two-body, etc.) and measure each fragment optimally.
- **Online shot allocation** — reallocate shots adaptively as the optimization proceeds.

Cross-links to Other Courses

- **Graph Theory** — coloring, chromatic number.
- **Operations Research** — resource allocation under uncertainty.
- **Statistics Module 9** — experimental design, optimal allocation of measurements.
- **Module 5 (Regularization)** — uses the shot budget as the headline cost metric.

The Ansatz as a Manifold

Position in Knowledge Graph

System: Variational Quantum Algorithms **Structural Domain:** Parameterized Circuits **Node:** 2.9

This is the **closing node** of Module 2 and its most geometric. Everything in the module so far has been described in coordinates: parameter vectors, Pauli strings, gate sequences, bitstrings, shot counts. This node steps back from coordinates and looks at the underlying geometric object: **the set of states the ansatz can prepare is a smooth submanifold of complex projective space, and many of the hardest optimization phenomena in VQA are properties of that manifold rather than properties of any particular parameterization.**

This is the geometric viewpoint that makes Module 3 (cost landscapes), Module 4 (barren plateaus), and Module 6 (optimization dynamics) feel coherent rather than ad-hoc. If you internalize this node, the rest of the course is a series of statements about a single geometric object that you can mentally visualize.

The Manifold

For an ansatz $U(\boldsymbol{\theta}) : \mathbb{R}^P \to U(2^n)$ acting on $|0\rangle^{\otimes n}$, the **ansatz manifold** is the image:

$$\mathcal{M} = \{|\psi(\boldsymbol{\theta})\rangle : \boldsymbol{\theta} \in \mathbb{R}^P\} \subset \mathbb{CP}^{2^n-1}.$$

We work in complex projective space $\mathbb{CP}^{2^n-1}$ rather than the unit sphere of $\mathbb{C}^{2^n}$ because global phases don't matter physically — $|\psi\rangle$ and $e^{i\alpha}|\psi\rangle$ are the same state.

Three structural facts about this set:

1. Dimension. The manifold has intrinsic dimension at most P, the parameter count. It can have lower dimension if some parameters are redundant (the parameterization is *singular* somewhere) or if the ansatz contains an exact symmetry that closes off a direction.

2. Smoothness. For circuit ansätze built from analytic gate families (rotations, controlled rotations), the parameterization $\boldsymbol{\theta} \to |\psi(\boldsymbol{\theta})\rangle$ is real-analytic almost everywhere. The manifold is smooth except at isolated singular points.

3. Restriction. $\mathcal{M}$ is a strict, generally low-dimensional subset of $\mathbb{CP}^{2^n-1}$, which itself has dimension $2(2^n - 1)$ — exponential in qubit count. For $n = 10$ qubits, the ambient projective space has real dimension 2046, while a typical 30-parameter ansatz manifold has dimension 30 (or less). **The ansatz traces out a thin 30-dimensional sheet inside a 2046-dimensional space.** Optimization is the act of moving along this sheet looking for a low-energy point.

Tangent Vectors and the Quantum Fisher Information

At a point $|\psi(\boldsymbol{\theta})\rangle$ on the manifold, the tangent space is spanned by the directional derivatives

$$|\partial_i \psi\rangle := \frac{\partial}{\partial \theta_i} |\psi(\boldsymbol{\theta})\rangle.$$

These vectors live in the ambient $\mathbb{C}^{2^n}$ but project (after removing the component parallel to $|\psi\rangle$) into the tangent space of $\mathcal{M}$ at $|\psi(\boldsymbol{\theta})\rangle$.

The natural metric on the manifold is the **Fubini-Study metric**, which on the tangent space takes the form

$$g_{ij}(\boldsymbol{\theta}) = \mathrm{Re}\langle \partial_i \psi | \partial_j \psi \rangle - \langle \partial_i \psi | \psi \rangle \langle \psi | \partial_j \psi \rangle.$$

In quantum information this is known as the **quantum Fisher information matrix** (up to a factor of 4) and it measures how "distinguishable" two infinitesimally separated parameter values are in terms of physical quantum measurements.

The crucial point: this metric is **not** the Euclidean metric δ_{ij} on parameter space. Different parameter directions span tangent vectors of very different lengths in the ambient state space. A "step of size ϵ in parameter space" can correspond to a tiny physical change in the state (if the parameter is in a redundant or weakly-coupled direction) or a huge physical change (if the parameter directly steers the state).

This mismatch is the reason **vanilla gradient descent in parameter space is not optimal** for VQAs. The natural object to descend is the cost gradient pulled back through the inverse metric:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta\, g^{-1}(\boldsymbol{\theta}_k)\, \nabla C(\boldsymbol{\theta}_k).$$

This is the **quantum natural gradient** (Stokes et al. 2020) and it is treated in detail in Module 6. For now, the important takeaway is: **the natural geometry of the ansatz lives on the manifold, not in parameter coordinates.**

Singularities

A point $\boldsymbol{\theta}_0$ is **singular** if the tangent vectors $\{|\partial_i \psi(\boldsymbol{\theta}_0)\rangle\}_{i=1}^{P}$ do not span a P-dimensional subspace — that is, if some directional derivative collapses to zero, or two directions become parallel.

At a singular point: - The local dimension of the manifold drops below P. - The metric g_{ij} becomes singular (determinant zero). - The natural gradient is undefined (the inverse metric blows up). - Multiple distinct $\boldsymbol{\theta}$-paths can lead to the same $|\psi\rangle$, creating a degeneracy.

These singularities are not abstract pathologies — they show up in real ansätze. Typical examples: - **Identity points**: a layer with all rotation angles set to 0 contributes the identity matrix and removes itself from the parameterization. - **Symmetry-equivalent parameters**: two parameters whose effects on the state are related by an exact group action. - **Compositional cancellations**: $R_X(\theta)R_X(-\theta) = I$ creates a 1-parameter family of $(\theta, -\theta)$ configurations all mapping to the same state.

The barren plateau phenomenon (Module 4) is *not* the same as a singularity — it is a global statistical phenomenon — but it can be exacerbated by the manifold passing through near-singular configurations.

Why This Matters For Optimization

Three operationally important consequences of the manifold view:

1. Step size in parameter space $\neq$ step size in state space. A learning rate η that is appropriate in one region of parameter space may be too aggressive or too conservative in another, depending on local manifold geometry. This is why adaptive learning rates (Adam, AdaGrad) often help VQAs — they implicitly normalize for changing local geometry.

2. Curvature drives convergence rate. Just as in classical optimization, the local curvature of the manifold (and the cost surface on it) determines how quickly an optimizer can converge. Highly curved regions need small steps; flat regions allow large steps. Vanilla gradient descent ignores this; second-order methods (natural gradient, Newton-Raphson, BFGS) try to exploit it.

3. The manifold is the same regardless of parameterization. Two ansätze that produce the same set of reachable states have the same manifold — even if their parameterizations are very different. **The**

geometry is intrinsic; the coordinates are convention. This is the fundamental reason that the *choice of parameterization* matters even when it doesn't change what states can be reached: it changes the relationship between parameter directions and tangent directions, and therefore changes which optimizers will be efficient.

The Picture for HEAs and UCCSDs

Two quick intuitive snapshots:

HEA manifold. Roughly Haar-uniform on the symmetry sector reachable by the ansatz. As depth grows, the manifold fills out more and more of the unit sphere in Hilbert space. At the depth where it approximates a 2-design, the manifold is "everywhere dense" in a useful sense — and that's exactly when barren plateaus appear. The manifold is too uniform; almost no direction is special.

UCCSD manifold. Heavily structured by the Hartree-Fock reference and the chosen excitations. Sits in a low-dimensional symmetry sector (fixed particle number, spin). Has natural starting point (the HF state, where all parameters are zero) and natural directions of motion (each excitation amplitude). The manifold is small but well-shaped, and the cost function has identifiable minima rather than uniform plateaus.

The contrast: **HEA gives you a featureless manifold; UCCSD gives you a featured one.** The featured manifold is harder to derive but easier to optimize.

Structural Role

This is the **conceptual capstone** of Module 2. It pulls Module 2's coordinate-level facts (gates, parameters, depths, encodings, shots) up into a coordinate-free geometric picture.

Forwards: - Module 3 (Cost Landscapes) treats the cost function $C : \mathcal{M} \to \mathbb{R}$ as a function on the manifold and studies its critical points and topology. - Module 4 (Barren Plateaus) is a statement about the typical curvature of the cost function on the manifold under random parameterization. - Module 6 (Optimization Dynamics) asks how to update parameters in a way that respects manifold geometry rather than parameter coordinates. - Module 8 (Expressivity vs Trainability) is the formal version of "featureless vs featured manifold."

This is the node where Module 2 stops being about plumbing and starts being about geometry.

Relations to Other Concepts

- **Information geometry** — the general theory of metric structures on parameterized probability/quantum models. Amari and Nagaoka are the standard references.
- **Riemannian manifolds** — the mathematical setting for the natural gradient and second-order methods.
- **Complex projective space** $\mathbb{CP}^{2^n-1}$ — the ambient space in which the ansatz manifold lives.
- **Fubini-Study metric** — the canonical metric on $\mathbb{CP}^n$, and the source of the quantum Fisher information.
- **Stiefel manifold** — alternative parameterization for orthonormal frames, occasionally used in tensor-network analogues of ansatz design.

Worked Example — Two Manifolds With The Same Reach

Consider two single-qubit ansätze:

Ansatz A: $|\psi_A(\theta,\phi)\rangle = R_Z(\phi)R_Y(\theta)|0\rangle$. Two parameters. Reaches every pure single-qubit state. The manifold is the **full Bloch sphere**.

Ansatz B: $|\psi_B(\alpha,\beta,\gamma)\rangle = R_Y(\gamma)R_Z(\beta)R_Y(\alpha)|0\rangle$. Three parameters. Also reaches every pure single-qubit state. The manifold is **also the full Bloch sphere**.

Same manifold. Different parameterizations.

What differs: - **Local metric.** At a generic point, the metric g_{ij} for ansatz A is full rank 2×2. For ansatz B it is rank-2 in a 3×3 matrix — there is one redundant direction at every point. - **Optimization behavior.** A gradient-descent run on ansatz A walks across the Bloch sphere in 2D. The same algorithm on ansatz B has an extra "null direction" (the redundant one) that the optimizer can drift along without changing the state. - **Parameter-space distance.** Two states close together on the Bloch sphere can correspond to parameter vectors that are far apart in either ansatz — but the relationship is different in the two parameterizations.

Both manifolds are the same set of physical states. **The cost landscape differences come entirely from how the parameters cover the manifold.** This is one of the cleanest demonstrations that the choice of parameterization matters even when the reachable set does not.

Visualization

Pending. A diagram showing the Bloch sphere with two coordinate grids overlaid: one from ansatz A (latitude/longitude-like, smooth), one from ansatz B (a 3D parameter space being projected, with a "redundant fiber" at each point shown as a small circle). Caption: "Same manifold, different coordinates, different gradient flows."

Exercises

1. For the 2-qubit ansatz $U(\theta_1,\theta_2) = (R_Y(\theta_1) \otimes R_Y(\theta_2))|00\rangle$, compute the tangent vectors $|\partial_1\psi\rangle$ and $|\partial_2\psi\rangle$. Are they orthogonal? What is the dimension of the resulting manifold?
2. Show that the parameterization $R_X(\theta_1)R_X(\theta_2)|0\rangle$ has a 1-dimensional kernel everywhere — i.e., the linear combination $|\partial_1\psi\rangle - |\partial_2\psi\rangle = 0$. What does this imply about the dimension of the manifold?
3. (VQA) Sketch an argument for why the natural gradient $g^{-1}\nabla C$ is invariant under reparameterizations of the ansatz, while the Euclidean gradient ∇C is not.

Research Extensions

- **Quantum information geometry of variational models** — formalizing the manifold view across ansatz families.
- **Geodesic optimization on ansatz manifolds** — using exact or approximate Riemannian optimizer steps that follow geodesics rather than straight lines in parameter space.
- **Manifold curvature and barren plateaus** — connecting global geometric properties to optimization difficulty.
- **Riemannian regularization** — adding curvature-aware penalties that act intrinsically rather than in parameter coordinates.

Cross-links to Other Courses

- **Differential Geometry** — manifolds, tangent spaces, metrics, curvature.
- **Information Geometry** — Fisher metric, dual connections, exponential families.

- **Module 3 (Cost Landscapes)** — the cost function as a scalar field on the manifold.
- **Module 6 (Optimization Dynamics)** — the natural gradient as the manifold-aware update rule.