Module 3 — Cost Landscapes

Variational Quantum Algorithms

hbar.university

Cost Function Design

Position in Knowledge Graph

System: Variational Quantum Algorithms **Structural Domain:** Cost Landscapes **Node:** 3.1

This is the **opening node** of Module 3 and the place where the question shifts from *what is the ansatz?* to *what is the function we are minimizing on it?*

The cost function in a VQA is **not** uniquely determined by the problem. You start with a problem (find the ground state of H) and you have some freedom to choose what scalar you actually optimize against. This node walks through that freedom, the standard choices, and the consequences each choice has for the resulting landscape.

It is a node about **design decisions you have to make before the optimizer ever sees a number**.

The Default: Energy Expectation

The textbook VQA cost function is the energy expectation:

$$C(\boldsymbol{\theta}) = \langle \psi(\boldsymbol{\theta}) | H | \psi(\boldsymbol{\theta}) \rangle.$$

Why this is the default: - It is bounded below by the true ground state E_0 (variational principle). - It has a meaningful physical interpretation as the expected energy of the prepared state. - It is linear in the Pauli decomposition of H, so it can be computed by Pauli-by-Pauli measurement (node 2.7). - It is the cost the variational principle directly addresses.

For 95% of VQA work — VQE for chemistry, VQE for spin models, ground-state preparation — this is what you optimize.

When The Default Isn't Enough

There are several situations where the energy expectation is a *bad* cost function and you have to design something else.

Excited states

The variational principle gives you the *ground* state. To find an excited state, you need a cost function whose minimum is the excited state of interest. Three approaches:

Subspace-search VQE. Optimize against $\mathrm{Tr}(H\rho_S)$ over states in a fixed subspace, with constraints enforcing orthogonality to lower states.

Variational Quantum Deflation. Add penalty terms to the energy expectation that push the state away from already-found lower states:

$$C(\boldsymbol{\theta}) = \langle\psi(\boldsymbol{\theta})|H|\psi(\boldsymbol{\theta})\rangle + \beta \sum_{i\in\text{found}} |\langle\psi_i|\psi(\boldsymbol{\theta})\rangle|^2.$$

The penalty term raises the cost of any state with overlap with previously found eigenstates.

SSVQE. Train multiple states simultaneously with constraints on their orthogonality.

Each of these is a cost function design choice that changes the landscape topology.

Dynamics simulation

For preparing a state evolved by e^{-iHt} variationally, the cost is not energy but **fidelity to a target state**:

$$C(\boldsymbol{\theta}) = 1 - |\langle\psi_{\text{target}}|\psi(\boldsymbol{\theta})\rangle|^2.$$

Or, computed in another way, the **infidelity in time-evolved measurements**.

These cost functions have qualitatively different landscape structures from the energy expectation — they are bounded above as well as below, they have explicit fixed points at $\boldsymbol{\theta}_0 = \arg\min C$, and they typically have *more* local minima than the energy expectation.

Quantum machine learning

For QML problems (variational classifiers, generative models), the cost function is a problem-specific loss:

- **Cross-entropy** for classification.
- **MSE** for regression.
- **MMD or Wasserstein** for generative models.

The cost is a function of the *measured outputs* of the circuit, which themselves are expectation values, but the relationship between parameters and cost is now mediated by a classical loss function. This adds another layer of nonlinearity (and another source of design freedom).

Constrained optimization

When the problem has symmetry constraints (particle number, total spin), you can enforce them either:

- **Implicitly**, by restricting the ansatz to states obeying the symmetry, OR
- **Explicitly**, by adding penalty terms to the cost: $C(\boldsymbol{\theta}) = \langle H\rangle + \mu\langle(N - N_0)^2\rangle$.

The penalty approach changes the landscape — it creates new minima at the constrained points and lifts the energy of states outside the constraint surface.

Cost Function Design as an Optimization Move

The key insight: **the cost function is itself a hyperparameter.**

Two cost functions that nominally encode the same problem can produce dramatically different optimization behaviors: - different barren plateau profiles, - different numbers of local minima, - different gradient magnitudes, - different convergence rates.

Most VQA papers fix the cost function early and then spend the rest of their effort optimizing the optimizer. A better question is sometimes: **could we have chosen a different cost function whose landscape is friendlier?**

Examples of cost-function-side improvements:

- **Local cost functions** (Cerezo et al. 2021): replace a global cost like $\langle H \rangle$ with a sum of local 2-qubit observables. Provably avoids barren plateaus for shallow ansätze.
- **Variance penalization** (node 5.7 in this course): add a term proportional to the *variance* of the cost estimator, which keeps the optimizer in regions where the cost is more reliably measurable.
- **Cost function re-normalization**: shift and rescale the cost so its dynamic range matches the optimizer's expectations. Surprisingly effective for adaptive learning rate methods.

Each of these is a *cost-side* intervention — modify the function being optimized rather than the algorithm doing the optimizing. It is the dual to optimizer-side interventions and often the more impactful one when it works.

Cost Function Design and Module 5

This connects directly to Module 5's regularization thesis. The thesis distinguishes between two kinds of regularization:

- **Prior injection** — modify the cost function (objective-side intervention).
- **Dynamical modification** — modify the optimizer's update rule (dynamics-side intervention).

The cost function design choices in this node are all examples of prior injection. They change C before the optimizer sees it. Whether this is a good idea — and how it interacts with the optimizer's dynamics — is exactly what Module 5's thesis is about.

So this node is the **upstream** of Module 5's main argument: it is the place where the practitioner consciously chooses what objective they are optimizing, and that choice determines what kind of regularization makes sense.

Structural Role

This is the opening node of Module 3 and the bridge from the **physical** content of Module 2 (how do we get a number out of the device?) to the **landscape** content of Module 3 (what does the resulting function look like?).

Forwards to 3.2 (topology), 3.3 (minima), and to Module 5 (regularization, where cost-function-side modification appears in full).

Relations to Other Concepts

- **Loss function design in ML** — same problem, different domain. Cross-entropy vs MSE vs hinge vs focal loss is the classical analogue.
- **Penalty methods in classical optimization** — Lagrangian relaxation, augmented Lagrangians, barrier functions.
- **Variational principle (1.1)** — guarantees the energy expectation is a defensible default.
- **Surrogate cost functions** — replace the true cost with a cheaper-to-evaluate proxy.

Worked Example — Two Costs For The Same Problem

For H_2, consider two cost functions:

Cost A: Energy expectation.

$$C_A(\boldsymbol{\theta}) = \langle\psi(\boldsymbol{\theta})|H_{H_2}|\psi(\boldsymbol{\theta})\rangle.$$

Range: [-1.85, +0.5] hartree. Minimum at $\boldsymbol{\theta}^*$: -1.137 hartree (FCI). Barren plateau profile (HEA): yes, in the random-init regime.

Cost B: Energy expectation + variance penalty.

$$C_B(\boldsymbol{\theta}) = \langle H\rangle + \lambda \mathrm{Var}[\widehat{\langle H\rangle}].$$

For the same minimizer, the variance is small (eigenstates have zero variance for their corresponding eigenvalue), so the penalty pushes toward eigenstate-like configurations. Range: same as above, plus a positive correction. Minimum: still near the FCI ground state, but the basin is *narrower* and *deeper* — variance vanishes only at eigenstates.

Which is better? Cost A converges faster initially (gradient is larger). Cost B converges slower but gets less stuck in mid-energy plateaus. The right answer depends on the optimizer being used and the problem regime.

This is the kind of tradeoff that cost function design can navigate — and that the rest of Module 3 will give you the tools to reason about.

Visualization

Pending. Two side-by-side cost surfaces. Left: pure energy expectation, with mild gradient and a wide basin. Right: energy + variance penalty, with steeper gradient and narrower basin. Caption: "The cost function is a design choice, not a given."

Exercises

1. For a 1-parameter ansatz $|\psi(\theta)\rangle = R_Y(\theta)|0\rangle$ and Hamiltonian $H = Z$, compute both $\langle H\rangle(\theta)$ and $\mathrm{Var}[\langle H\rangle](\theta)$. Where are the minima of each?
2. The variational principle gives $\langle H\rangle \geq E_0$. Does the same kind of guarantee hold for the variance-penalized cost? Argue carefully.
3. (VQA) Design a cost function whose minimum is the *first excited state* of H, given knowledge of the ground state $|\psi_0\rangle$. What new local minima does your construction introduce?

Research Extensions

- **Adaptive cost reformulation** — reshape the cost during optimization based on observed difficulty.
- **Cost function bandits** — treat the choice of cost function as an arm to be explored.
- **Dual cost functions** — optimize a relaxation, then refine on the original.
- **Cost-shaping for QML** — designing classification losses tuned to circuit ansatz structure.

Cross-links to Other Courses

- **Machine Learning** — loss function design as a first-class concern.
- **Optimization Theory** — penalty methods, Lagrangian duality.
- **Module 5 (Regularization)** — the natural continuation of this node.
- **Module 8 (Expressivity vs Trainability)** — local vs global cost functions and barren plateaus.

Landscape Topology

Position in Knowledge Graph

System: Variational Quantum Algorithms **Structural Domain:** Cost Landscapes **Node:** 3.2

Once you have a cost function (3.1), you can ask: **what does it look like as a function on parameter space?** The answer is a **landscape** — a real-valued function over a (typically high-dimensional) torus, with a topology determined jointly by the ansatz, the Hamiltonian, and the cost choice.

This node introduces the basic topological vocabulary that the rest of Module 3 will use, and characterizes the qualitative features that VQA landscapes exhibit.

It is a vocabulary node, not a theorem node. The point is to give you the words for what you're seeing when you look at a cost surface — *basin*, *ridge*, *saddle*, *plateau*, *period* — and to anchor the intuition that these features are not random but are determined by the structure of the underlying variational problem.

Domain: A Torus, Not a Vector Space

The first topological fact about VQA cost functions: their domain is a **torus**, not Euclidean space.

For an ansatz built from rotation gates $R_P(\theta)$ where each Pauli P has eigenvalues ± 1, the cost function satisfies

$$C(\boldsymbol{\theta} + 2\pi \mathbf{e}_i) = C(\boldsymbol{\theta})$$

for every parameter θ_i. This periodicity makes the natural domain $\mathbb{T}^P = (S^1)^P = [-\pi, \pi]^P$ with edges identified.

Practical consequences:

- **No "boundary" of parameter space.** Every direction wraps around. The optimizer cannot "go to infinity."
- **Parameter-norm regularization is awkward.** L2 penalties $\lambda\|\boldsymbol{\theta}\|^2$ prefer one specific origin, but on a torus there is no privileged origin — every point is equivalent topologically.
- **Multiple paths between any two points.** On a torus, you can go around the long way or the short way, and an optimizer with momentum can sometimes exploit this.
- **Topology contributes minima.** A function on a torus has, by Morse theory, a minimum number of critical points determined by the topology — at minimum, one minimum, one maximum, and (for $P \geq 2$) several saddle points. **You cannot have a torus with only one critical point.**

The torus is a standard mathematical setting for periodic optimization but is rarely emphasized in VQA papers. It quietly determines a lot of what is and isn't possible.

Five Qualitative Features

Every VQA landscape exhibits some mixture of the following features.

1. Basins

A **basin** is a region around a local minimum where the gradient points (statistically) toward the minimum. Optimizers that find a basin will descend to its bottom and stay there unless they can escape.

VQA landscapes typically have **many basins**, ranging from very shallow (small region of attraction, mild depth) to deep (large region, low energy). The structure of basins is the most important thing for an optimizer's behavior, more important than the global minimum location.

2. Ridges and Saddles

A **saddle point** is a critical point where the Hessian has both positive and negative eigenvalues. The cost decreases in some directions and increases in others. Gradient descent slows down at saddles but eventually escapes them through the descent directions; SGD can escape faster due to noise.

A **ridge** is a region of high cost separating two basins. Crossing a ridge requires either going around it (energy-conserving path) or briefly increasing the cost (energy-violating path). Most local optimizers cannot cross ridges; this is what makes them get stuck in suboptimal basins.

For typical VQA ansätze, the ratio of saddle points to local minima can be quite high — much of the "structure" of a high-dimensional landscape is saddle-shaped, and avoiding saddles is a real concern.

3. Plateaus

A **plateau** is a region where the gradient is small in all directions. Plateaus come in two flavors:

- **Topological plateaus** are regions of genuinely small gradient — flat regions of the function. They occur near critical points and in "valleys" between minima.
- **Concentration plateaus** (or barren plateaus, Module 4) are regions where the gradient is exponentially small *in expectation* — a statistical property of how the cost function is constructed, not a feature of any particular point.

Both look the same to a local optimizer: the gradient is too small to drive movement. Distinguishing them requires global information that the optimizer doesn't have.

4. Periodic Repeats

Because the domain is a torus, every feature appears infinitely many times (one per fundamental cell). A local minimum at $\boldsymbol{\theta}^*$ is also at $\boldsymbol{\theta}^* + 2\pi\mathbf{e}_i$ for every i. This means an optimizer with periodic awareness can move between fundamental cells "for free" — sometimes useful for escaping a plateau by jumping a half-period.

In practice, most optimizers are unaware of the periodic structure and treat parameter space as Euclidean. This is a small operational loss.

5. Symmetry-Equivalent Critical Points

If the ansatz has a symmetry — e.g., parameters θ_i and θ_j play interchangeable roles for some pair (i, j) — then critical points come in equivalence classes under that symmetry. A local minimum at (θ_1^*, θ_2^*) is also a local minimum at (θ_2^*, θ_1^*). This is treated in detail in node 3.4.

What VQA Landscapes Look Like in Aggregate

Combining all of the above, the typical VQA landscape has the following statistical character:

- **Many local minima.** An ansatz with P parameters typically has $O(P)$ or more local minima for chemistry-style cost functions.
- **Many more saddles.** Saddle points outnumber minima by a substantial factor in high dimensions.
- **Mostly flat in expectation.** For random initializations of expressive ansätze, the *average* gradient magnitude is small compared to the global cost range.
- **Non-trivial basin sizes.** The basins of attraction of local minima vary in size from "the global minimum with a basin filling most of parameter space" to "tiny isolated minima with negligible attraction."
- **Symmetry-related structure.** Many critical points are equivalent under exchange or sign symmetries.

This is qualitatively similar to neural network loss landscapes (as studied in deep learning theory), and many of the heuristics from that field transfer. But the *detailed* structure is determined by the ansatz design and the Hamiltonian, and it cannot be assumed from the analogy alone.

Why This Matters For Optimizer Choice

The features above interact differently with different optimizers:

- **Gradient descent** does well in smooth basins, fails on plateaus and saddles, cannot cross ridges.
- **Adam / momentum methods** can build up enough velocity to escape mild saddles and climb small ridges.
- **Population-based methods** (HOPSO, evolutionary strategies) maintain diverse points across multiple basins simultaneously.
- **Bayesian optimization** is good at navigating landscapes with few minima but expensive evaluations; struggles with high-dimensional plateau regions.
- **SPSA** is robust to noise and can escape some plateaus through random perturbation.

The thesis's argument that the choice of optimizer is the dominant variable in many VQA experiments (node 1.9) is in part a statement about how different optimizers handle the topology features above. **Two optimizers running on the same landscape can have qualitatively different trajectories because they engage with the topology differently.**

Structural Role

This node provides the **vocabulary** for the rest of Module 3. The next nodes elaborate specific features:

- 3.3: local vs global minima in detail.
- 3.4: how symmetries shape the landscape.
- 3.5: gradients and curvature, the local structure.
- 3.6: concentration of measure, the statistical structure.
- 3.7: shot noise, the stochastic structure.
- 3.8: parameter-shift rule, the gradient computation.

You'll come back to this node's vocabulary repeatedly. It is the lens.

Relations to Other Concepts

- **Morse theory** — the topology of smooth functions on manifolds. Counts critical points by index and relates them to the manifold's homology.
- **Loss landscape topology in deep learning** — analogous study of the loss landscapes of neural networks. Exhibits many of the same features.
- **Random landscape theory** — physical models of random Gaussian functions on high-dimensional spaces. Predicts saddle/minimum ratios in high dimension.
- **Spin glass landscapes** — the canonical model of a "rugged" landscape with many critical points.

Worked Example — A 2D VQE Landscape

For $H = Z_0 + Z_1 + 0.5X_0X_1$ on 2 qubits with ansatz $|\psi(\theta_1, \theta_2)\rangle = R_Y(\theta_1) \otimes R_Y(\theta_2)|00\rangle$:

$$C(\theta_1, \theta_2) = \cos\theta_1 + \cos\theta_2 + 0.5 \sin\theta_1 \sin\theta_2.$$

This landscape on the 2-torus $[-\pi, \pi]^2$ has:

- A **global minimum** near $(\theta_1, \theta_2) = (\pi - \arctan(0.25), \pi - \arctan(0.25))$ where both Z's are -1 and the XX correlation aligns.
- A **second local minimum** related by the symmetry $(\theta_1, \theta_2) \to (-\theta_1, -\theta_2)$.
- **Two saddle points** near $(0, \pi)$ and $(\pi, 0)$ where the Z's disagree.
- **A maximum** near $(0, 0)$ where everything is $+1$.
- **Periodic copies** of all of the above shifted by 2π in each direction.

You can visualize this surface as a 2D heatmap and see all of these features clearly. **This is the structure your optimizer is navigating, even when you can't visualize it.**

Visualization

Pending. A 2D heatmap of the worked example, showing minima (dark), maxima (light), saddles (mid), with periodic boundaries. Annotated with the topological vocabulary words: basin, ridge, saddle, plateau.

Exercises

1. For the 2D landscape $C(\theta_1, \theta_2) = \cos\theta_1 + \cos\theta_2$, enumerate all critical points on the fundamental cell $[-\pi, \pi]^2$ and classify them as min/max/saddle.
2. Show that on a 1-torus S^1, any continuous function must have at least one minimum and one maximum (this is a special case of Morse inequalities). Generalize to the 2-torus.
3. (VQA) Sketch how a basin's *size* differs from its *depth*, and explain why an optimizer might prefer a wide-shallow basin over a narrow-deep one even when the latter has lower energy.

Research Extensions

- **Topological data analysis of cost landscapes** — using persistent homology to characterize landscape structure across ansatz families.
- **Critical point counting for variational ansätze** — analytic results for specific ansatz classes.
- **Connectivity of basins** — when are two basins connected by descent paths in the cost-restricted manifold?
- **Mode connectivity** — borrowing ideas from neural network landscapes about connected sets of low-cost points.

Cross-links to Other Courses

- **Topology / Morse Theory** — critical points, indices, homology.
- **Statistical Physics** — random landscapes, spin glasses, replica calculations.
- **Machine Learning** — loss landscape geometry of deep networks.
- **Module 4 (Barren Plateaus)** — concentration plateaus in detail.

Local and Global Minima

Position in Knowledge Graph

System: Variational Quantum Algorithms **Structural Domain:** Cost Landscapes **Node:** 3.3

This node zooms in on the most important topological feature of a VQA landscape — its **minima** — and dissects what they mean for optimization.

The mantra "we want to find the global minimum" is correct in the abstract but operationally misleading in VQA practice. For most realistic VQA instances, the global minimum is **unreachable** by local optimization, and the practical question is **which local minimum can the optimizer find, and how good is it?**

Definitions

A **critical point** $\boldsymbol{\theta}^*$ of $C : \mathbb{T}^P \to \mathbb{R}$ satisfies $\nabla C(\boldsymbol{\theta}^*) = 0$.

A critical point is a **local minimum** if there exists a neighborhood U of $\boldsymbol{\theta}^*$ such that $C(\boldsymbol{\theta}^*) \leq C(\boldsymbol{\theta})$ for all $\boldsymbol{\theta} \in U$.

A critical point is a **strict local minimum** if the inequality is strict for $\boldsymbol{\theta} \neq \boldsymbol{\theta}^*$.

A **global minimum** is the lowest local minimum on the entire domain.

The **basin of attraction** of a local minimum $\boldsymbol{\theta}^*$ is the set of starting points $\boldsymbol{\theta}_0$ from which gradient descent (with appropriately small step size) converges to $\boldsymbol{\theta}^*$.

How Many Minima?

A typical VQA cost function on P parameters has **many** local minima — usually more than P, often growing polynomially or even exponentially in P.

This is true for two related reasons:

1. Symmetry. Ansatz symmetries create equivalence classes of minima. If the ansatz has a permutation symmetry over k parameters, there are at least $k!$ symmetry-equivalent minima for any non-symmetric configuration. Permutation-symmetric HEAs at moderate qubit count can have hundreds of equivalent minima.

2. Compositional cancellation. Layers in a deep ansatz can compose to (approximately) the identity in many distinct ways, creating local minima at parameter configurations whose effective state is similar. $R_X(\theta_1)R_X(\theta_2)$ has a 1-parameter family of (θ_1, θ_2) configurations that all give the same single-qubit unitary, and any cost function on this combined parameterization inherits a 1-parameter degeneracy.

The number of minima grows roughly with the number of "ways" the ansatz can prepare each state, which for expressive ansätze is large.

How Good Are The Local Minima?

A more important question than "how many?" is **how good are they?**

For UCCSD-style chemically-motivated ansätze, the result is often striking: **almost all local minima are within chemical accuracy of the global minimum.** That is, any optimizer that finds *any* local minimum is essentially "done." This is one of the key practical advantages of problem-inspired ansätze.

For HEA-style hardware-motivated ansätze, the result is much worse: local minima can range from "barely better than random" to "near the global minimum," with the typical value far above the global minimum and the gap growing with depth. HEAs do have many local minima, but most of them are mediocre.

This is the **quality of local optima** problem, and it is one of the main reasons HEAs are usually outperformed by problem-inspired ansätze on chemistry instances even when both ansätze are equally trainable. The HEA *can* find a low-cost point, but the typical low-cost point it lands in is much higher than the typical low-cost point of UCCSD.

Why You (Usually) Can't Find The Global Minimum

For high-dimensional non-convex problems, finding the global minimum is generally NP-hard. For VQA problems specifically, several additional obstacles apply:

1. Exponentially many local minima. Even if you could count them, you can't afford to visit them all.

2. Saddles between basins. Local optimizers cannot cross saddles upward. To get from one basin to another, you need either restart-based exploration or a global method.

3. Noisy evaluations. You don't actually know whether a point is a local minimum or just a region with small gradient. Distinguishing requires more shots.

4. Limited query budget. Even global methods need many queries to explore a high-dimensional space, and each query is expensive.

In practice, VQA practitioners do **multi-start optimization**: run gradient descent (or another local method) from many random initializations, take the best result. This is not guaranteed to find the global minimum but typically finds a good one. Algorithms like HOPSO and CMA-ES are essentially principled multi-start strategies that share information between restarts.

The "Reachability vs Findability" Distinction

A critical conceptual move: **a state being reachable on the manifold is not the same as a state being findable by the optimizer.**

For an ansatz that contains the true ground state in its image: - **Reachable**: there exists $\boldsymbol{\theta}^*$ such that $|\psi(\boldsymbol{\theta}^*)\rangle$ equals the ground state. - **Findable**: starting from a typical initialization, the optimizer converges to $\boldsymbol{\theta}^*$ (or to a parameter giving a state with energy near the ground state) within the available query budget.

The first is a property of the ansatz manifold (Module 2.9). The second is a property of the *combination* of the ansatz, the cost function, the optimizer, and the initialization.

A barren plateau is the extreme case of "reachable but not findable" — the manifold contains the answer, but the cost gradient is so small everywhere that no local optimizer can detect a descent direction.

This is also exactly the conceptual move that node 1.9 (separation of objective and dynamics) demands: reachability is an objective property, findability is a composition property.

What An Optimizer Actually Settles On

In practice, when a VQA optimizer "converges," it has found one of the following:

1. **A local minimum** of the noisy cost function (which may or may not be a local minimum of the noiseless cost).

2. **A noise-equilibrium point** where the gradient signal is below the noise floor and the optimizer's updates cancel out on average.
3. **A point in a long flat region** where the gradient is small but not zero, and the optimizer is moving slowly.

Distinguishing these is hard. A common heuristic is to look at the variance of the cost over the last K iterations: if it is dominated by the optimizer's step size, you are at a noise equilibrium; if it is dominated by shot noise, you are at a stable point.

Practical Implications

Three operational rules of thumb that follow from the above:

1. Always do multi-start. Single-start optimization wastes the structure of the landscape — you can pay the cost of multiple optimizations in parallel and at least know which basin you've found.

2. Don't trust convergence as quality. A "converged" run might be in a mediocre local minimum. Always compare the converged value against a known reference (Hartree-Fock, MP2, classical CCSD).

3. Pay attention to optimizer-specific basin preferences. Different optimizers find different local minima from the same starting point, and their preferences can be exploited (or accidentally encode bias).

These are not theorems. They are tactical observations from practice. Most published VQA results follow them implicitly. Most failed VQA experiments did not.

Structural Role

This node sharpens the topology vocabulary of node 3.2 by zooming in on minima specifically. It is also the conceptual setup for Module 4 (barren plateaus): the failure mode introduced there is "all local minima become unreachable in expectation," which is a more extreme form of the problem characterized here.

Relations to Other Concepts

- **Loss landscape geometry of neural networks** — many of the same phenomena (many minima, mostly equivalent for overparameterized networks; basin connectivity).
- **Random energy models** — statistical physics models for random landscapes that predict the distribution of local minima energies.
- **Combinatorial optimization** — global vs local minima as a problem class.
- **Multi-start optimization** — the standard practical technique for landscape exploration.

Worked Example — Counting Minima In An HEA

Consider a 2-qubit HEA with 1 layer: $|\psi(\boldsymbol{\theta})\rangle = \text{CNOT} \cdot (R_Y(\theta_1) \otimes R_Y(\theta_2)) \cdot \text{CNOT} \cdot (R_Y(\theta_3) \otimes R_Y(\theta_4))|00\rangle$.

For a Hamiltonian $H = Z_0 + Z_1$, the cost function $C(\boldsymbol{\theta})$ is a 4D function. Numerical exploration shows:

- **At least 16 local minima** in the fundamental cell $[-\pi, \pi]^4$, arising from sign-flip symmetries: $(\theta_i \to -\theta_i + 2\pi)$ gives an equivalent state up to a global phase, and 4 parameters $\times$ 2 sign choices $= 2^4 = 16$.
- **The global minimum** is at $(0, 0, \pi, \pi)$ or its sign-flip equivalents, where both qubits are flipped to $|11\rangle$ and $\langle Z_0 + Z_1\rangle = -2$.

- **All 16 minima have the same energy** (because they all map to the same state up to a global phase).
- **Saddles between them** correspond to half-flipped configurations with energy 0 or near-0.

So this 4-parameter ansatz has 16 equivalent global minima and a complicated structure of saddles between them. A gradient-descent optimizer starting from a random init will find one of the 16 — which one depends on the basin of attraction the init falls into — but since they're all equivalent, "finding the global minimum" is trivial here.

For a more interesting Hamiltonian (e.g., $H = Z_0 + Z_1 + 0.3X_0X_1$), the symmetry is broken slightly and the 16 minima split into 16 *near-equivalent* minima at slightly different energies. Then the choice of optimizer starts to matter: a finer optimizer will find a slightly lower one, but they're all close.

This is the pattern: **HEAs generate many minima by symmetry, and which one you find usually doesn't matter much — until you turn off the symmetry, at which point the structure suddenly becomes important.**

Visualization

Pending. A 2D slice of the 4D HEA landscape, showing 4 of the 16 minima (one quadrant of the sign-flip group). Color: cost. Annotations: basins, saddles, the global value. Caption: "4 of 16 equivalent minima."

Exercises

1. For the cost function $C(\theta_1, \theta_2) = -\cos\theta_1 - \cos\theta_2$ on $[-\pi, \pi]^2$, find all local minima. How many are there? How many are global?
2. Show that the cost function in Exercise 1 has the same number of local maxima as local minima, by Morse-theoretic reasoning.
3. (VQA) Describe a multi-start optimization protocol for a 20-parameter VQA. How many restarts would you do? How would you decide when to stop?

Research Extensions

- **Counting critical points of variational ansätze** — analytic upper and lower bounds.
- **Energy distribution of local minima** — what is the typical gap between the lowest local minimum found and the global minimum?
- **Basin volume estimation** — for a given local minimum, how big is its basin? Determines how easy it is to find by random initialization.
- **Saddle-aware optimizers** — methods that explicitly seek out and cross saddles.

Cross-links to Other Courses

- **Optimization Theory** — local vs global minima, convergence guarantees.
- **Statistical Physics** — random landscape models, Kac-Rice formula for critical point counting.
- **Module 4 (Barren Plateaus)** — the "all minima unreachable" failure mode.
- **Module 6 (Optimization Dynamics)** — how different optimizers handle multi-minima landscapes.

Symmetry in Landscapes

Position in Knowledge Graph

System: Variational Quantum Algorithms **Structural Domain:** Cost Landscapes **Node:** 3.4

Symmetry in a VQA cost landscape is both a **gift** and a **curse**, depending on how you look at it.

The **gift**: a symmetric landscape has equivalent critical points related by group actions. Once you find one, you've effectively found them all. Symmetries can be exploited to restrict the search space.

The **curse**: symmetric landscapes have *redundant* parameter directions. Two distinct parameter vectors can produce the same physical state, which means the parameter-space gradient has a kernel — there are directions in parameter space that don't change anything, but the optimizer still tries to move along them, wasting effort.

This node walks through both sides of the symmetry coin and explains how to think about symmetry-aware optimization.

Three Sources of Symmetry

There are three distinct kinds of symmetry that show up in VQA cost functions, and they have very different consequences.

1. Hamiltonian Symmetries

If the Hamiltonian H commutes with some unitary G, then the eigenstates of H split into invariant subspaces under G, and the ground state can be chosen to lie in a specific symmetry sector.

Examples: - **Particle number symmetry**: $[H, \hat{N}] = 0$ for fermionic Hamiltonians. - **Total spin / spin projection**: for spin-conserving Hamiltonians. - **Translation invariance**: for lattice Hamiltonians on translation-invariant lattices. - **Parity** (Z2 symmetries): for many quantum chemistry Hamiltonians after Jordan-Wigner. - **Time-reversal**: for Hamiltonians made of real combinations of Pauli operators.

Consequences for the cost landscape: - The ground state lies in one specific symmetry sector. If the ansatz prepares states outside that sector, those states have higher energy than they "should" — the cost function has local minima outside the relevant sector that the optimizer might mistakenly find. - Symmetry-preserving ansätze (constructed to commute with G) restrict the search to the right sector, removing those spurious minima. - Symmetry-violating ansätze must "discover" the symmetry through optimization, which is generally slow and unreliable.

2. Ansatz Symmetries

If the ansatz $U(\boldsymbol{\theta})$ has a structural symmetry — e.g., parameters θ_i and θ_j are interchangeable — then the cost function inherits a **discrete symmetry group action** on parameter space.

Examples: - **Permutation symmetry of parallel layers**: in an HEA where every layer is identical, swapping the parameters of layer 3 with layer 5 produces a different parameter vector with the same cost. - **Sign-flip symmetry**: $R_Y(\theta)$ and $R_Y(-\theta + 2\pi)$ prepare states differing by a phase. This can be a 1- or 2-element symmetry depending on the surrounding circuit. - **Qubit permutation symmetry**: when the device topology and ansatz are symmetric under qubit relabeling.

Consequences for the cost landscape: - Critical points come in equivalence classes. A local minimum at $\boldsymbol{\theta}^*$ is also a local minimum at every $g \cdot \boldsymbol{\theta}^*$ for $g \in G$. - The number of critical points is multiplied by $|G|$. - Symmetry-related basins are equally good — the optimizer is indifferent between them.

This is the source of the "16 equivalent minima" phenomenon described in node 3.3.

3. Parameterization Redundancy (Continuous Gauge)

Some ansatz parameterizations have **continuous redundancies** — directions in parameter space that don't change the prepared state at all.

Example: $R_X(\theta_1)R_X(\theta_2)|0\rangle$ depends only on the sum $\theta_1 + \theta_2$, not on the individual values. The direction $(\theta_1, \theta_2) \to (\theta_1 + s, \theta_2 - s)$ is a 1-parameter gauge orbit — no physical motion, just relabeling.

Consequences for the cost landscape: - The cost function has flat directions along the gauge orbits. - The Hessian has a kernel; the metric g_{ij} is degenerate (rank deficient). - Any local minimum is part of an entire submanifold of equivalent minima of dimension equal to the gauge orbit dimension. - The natural gradient is undefined in the gauge directions.

These continuous redundancies are pathological — they waste parameters and slow optimization without producing any genuine minimum structure. Good ansatz design avoids them where possible.

How To Use Symmetry

The good move is: **identify symmetries and exploit them to shrink the search space.**

For Hamiltonian symmetries:

- **Restrict the ansatz** to states obeying the symmetry. For particle number, this means using particle-number-preserving gates. For parity, this means using only even Pauli rotations.
- **Encode the symmetry into the cost function** as a penalty. Add $\mu(\langle G\rangle - g_0)^2$ to the energy expectation, where g_0 is the desired symmetry charge. Pushes the optimizer toward the correct sector.
- **Project the state** onto the symmetric subspace before measurement. Equivalent to ground-state preparation in a smaller Hilbert space.

For ansatz symmetries:

- **Quotient out the symmetry** — use a parameterization where redundant directions are explicitly removed. Sometimes possible by clever ansatz design.
- **Symmetry-aware initialization** — start the optimizer at a representative point of each equivalence class to ensure broad exploration.
- **Symmetry-aware aggregation** — when reporting "the converged parameters," report the equivalence class rather than a specific representative.

For continuous gauge:

- **Eliminate the gauge** by reparameterizing. $R_X(\theta_1)R_X(\theta_2) \to R_X(\theta_1 + \theta_2)$.
- If you can't eliminate it (gauges arising from circuit composition aren't always removable), use the **natural gradient** (Module 6) which automatically projects out gauge directions.

How Symmetry Misleads

The bad move is to **ignore symmetry**. Three failure modes:

1. Optimizer wastes effort on gauge. A vanilla gradient descent on a gauge-redundant parameterization spends real shots evaluating the gradient in gauge directions, where it always reads zero plus noise. The optimizer is "moving" in those directions but the state isn't.

2. Symmetry-broken minima look like progress. If the cost function has a Hamiltonian symmetry but the ansatz doesn't preserve it, the optimizer will find local minima at symmetry-broken states. These look like "good progress" (the cost is decreasing) but the converged state has the wrong quantum numbers — it is not the physical ground state.

3. Spurious basins from broken parameterization symmetries. If your ansatz has a discrete symmetry the cost function does not, then the symmetry-related "minima" are not actually equivalent — they have slightly different costs because of the asymmetry — and the optimizer can get confused about which one to commit to.

The right diagnostic for all of these is to **inspect the converged state**, not just the converged cost. Compute the expectation values of the symmetry generators. If they don't match the expected charges, you have a symmetry-related bug.

Symmetry and Module 5

Module 5's regularization framework has a clean interaction with symmetry:

- **Prior injection (objective-side)** can encode symmetry as a penalty term added to the cost.
- **Dynamical modification (dynamics-side)** can encode symmetry as a constraint on the optimizer's update rule (project the gradient onto the symmetric subspace).

These produce *different* converged states even when nominally enforcing the same symmetry. The thesis's argument that these two regularization styles have different semantics applies in the symmetry case as well — perhaps especially clearly there.

Structural Role

This is the **structural awareness** node of Module 3. It rounds out the topology discussion (3.2-3.3) with the observation that not all topology is independent — structural relations between minima exist and can be exploited.

Forwards: every symmetry-aware ansatz design choice in Module 2.3 (chemically-inspired ansätze) is consistent with this node.

Relations to Other Concepts

- **Group theory** — the mathematical setting for discrete symmetries.
- **Gauge theory** — the formal treatment of continuous parameterization redundancies.
- **Equivariant neural networks** — the classical ML analogue of symmetry-preserving ansätze.
- **Symmetry breaking in physics** — the process by which a symmetric Hamiltonian acquires a non-symmetric ground state. Different phenomenon, often confused.
- **Quotient space optimization** — optimizing on the quotient by a symmetry group.

Worked Example — Sign-Flip Symmetry of R_Y

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

For Hamiltonian $H = X$, the cost is

$$C(\theta) = \langle\psi|X|\psi\rangle = 2\cos(\theta/2)\sin(\theta/2) = \sin(\theta).$$

This has minima at $\theta = -\pi/2$ and $\theta = 3\pi/2$, and these differ by 2π — they are the **same point on the torus** S^1. One minimum, no symmetry.

Now consider $H = Z$. The cost is

$$C(\theta) = \cos(\theta).$$

Minimum at $\theta = \pi$. Maximum at $\theta = 0$. The function has reflection symmetry $\theta \to -\theta$, but $-\pi = \pi$ on the torus, so the symmetry has just one fixed point and no equivalence class structure. Again one minimum.

Now consider $H = X^2 + Z^2 = 2I$ (a useless Hamiltonian, but pedagogically clean): the cost is constant. Every point is equivalent. The "symmetry" is the entire group acting trivially.

These three examples show that the *manifest* symmetry of the cost function depends jointly on the ansatz and the Hamiltonian. **Symmetry isn't a property of the ansatz alone or the Hamiltonian alone; it's a property of their composition.**

Visualization

Pending. A 2D parameter space colored by cost, with discrete symmetry orbits highlighted (e.g., 4-fold permutation orbits). Multiple equivalent minima shown connected by group action arrows. Caption: "Symmetry × cost function = orbit structure."

Exercises

1. The Hamiltonian $H = Z_0 + Z_1$ is invariant under qubit swap. The ansatz $R_Y(\theta_1) \otimes R_Y(\theta_2)|00\rangle$ is also invariant under qubit swap. Show that the cost function has a $\mathbb{Z}_2$ symmetry, and characterize the orbits.
2. Design a symmetry-violating perturbation to the above Hamiltonian and show that the symmetry of the cost function is broken.
3. (VQA) Explain why a symmetry-preserving ansatz can sometimes have **fewer** local minima than a symmetry-breaking one. Is fewer minima always good?

Research Extensions

- **Equivariant VQAs** — ansätze whose action commutes with a target symmetry group by construction.
- **Symmetry-protected initial states** — restricting the reference state to lie in a specific symmetry sector.
- **Symmetry detection algorithms** — automatic discovery of symmetries in arbitrary Hamiltonians.
- **Quotient-space optimization** — optimizing on the parameter space modulo the symmetry group.

Cross-links to Other Courses

- **Group Theory** — discrete and continuous symmetries.
- **Differential Geometry** — quotient manifolds, principal bundles.
- **Equivariant ML** — neural networks invariant under group actions.
- **Module 2.3** — chemically-inspired ansätze are symmetry-aware by construction.

Landscape Geometry and Gradients

Position in Knowledge Graph

System: Variational Quantum Algorithms **Structural Domain:** Cost Landscapes **Node:** 3.5

This node is about the **local** structure of a VQA cost landscape: gradient magnitude, curvature, the metric, and how these properties change as you move around parameter space.

Local geometry is what every optimizer "sees" — gradient descent uses the gradient, second-order methods use the Hessian, natural gradient uses the metric. Understanding local geometry is therefore the bridge between the topological vocabulary of nodes 3.2-3.4 and the operational behavior of an optimizer in Modules 5-7.

The Gradient

The cost function gradient is

$$\nabla C(\boldsymbol{\theta}) = \left(\frac{\partial C}{\partial \theta_1}, \ldots, \frac{\partial C}{\partial \theta_P} \right).$$

For a VQA with cost $C(\boldsymbol{\theta}) = \langle \psi(\boldsymbol{\theta}) | H | \psi(\boldsymbol{\theta}) \rangle$, each component is

$$\frac{\partial C}{\partial \theta_i} = \langle \partial_i \psi | H | \psi \rangle + \langle \psi | H | \partial_i \psi \rangle = 2 \, \mathrm{Re} \langle \partial_i \psi | H | \psi \rangle,$$

where $|\partial_i \psi\rangle$ is the partial derivative of the state with respect to θ_i.

For circuit ansätze built from Pauli rotations $R_P(\theta) = e^{-i\theta P/2}$, this can be evaluated **exactly** using the parameter-shift rule (node 3.8):

$$\frac{\partial C}{\partial \theta_i} = \frac{1}{2} \left[C(\boldsymbol{\theta} + \tfrac{\pi}{2} \mathbf{e}_i) - C(\boldsymbol{\theta} - \tfrac{\pi}{2} \mathbf{e}_i) \right].$$

This identity is what makes gradient-based VQA optimization viable — exact gradients can be computed by *evaluating the cost function* at shifted parameter values, no finite-difference approximation needed.

The trade is that each gradient component costs two cost evaluations. For a P-parameter ansatz, computing the full gradient costs $2P$ cost evaluations — and each cost evaluation is itself M measurement bases $\times$ N shots.

Gradient Magnitude

How big is $\|\nabla C(\boldsymbol{\theta})\|$ at a typical point?

For shallow problem-inspired ansätze (UCCSD, HVA, ADAPT-VQE on chemistry instances), the gradient magnitude at the Hartree-Fock starting point is usually substantial — large enough to drive meaningful descent.

For random initializations of expressive HEAs, the gradient magnitude is *exponentially small in qubit count* — this is the barren plateau result, which Module 4 will treat in full. At $n = 20$ qubits, the typical gradient magnitude in a random HEA region can be 10^{-6} or smaller, far below the shot-noise floor of any reasonable budget.

At critical points, gradient is exactly zero by definition. Near a critical point, the gradient grows linearly with distance from the critical point, with the rate set by the Hessian eigenvalues.

The practical question is: **is the gradient distinguishable from shot noise?** Shot noise sets a floor on detectable gradient magnitudes:

$$\text{detectable iff} \quad \|\nabla C\| \gtrsim \frac{\sigma_{\text{shot}}}{\sqrt{\text{shots/component}}}.$$

When the gradient magnitude drops below this threshold, the optimizer effectively cannot distinguish "no signal" from "small signal" and progress stalls. This is the **shot noise floor** that Module 5 spends a lot of effort working around.

The Hessian

The second-order structure is captured by the Hessian:

$$\mathbf{H}_{ij}(\boldsymbol{\theta}) = \frac{\partial^2 C}{\partial\theta_i \partial\theta_j}.$$

For Pauli-rotation ansätze, the Hessian can also be computed via parameter shifts (a "double-shift rule"), but each component costs four cost evaluations, so the full $P \times P$ Hessian costs $O(P^2)$ cost evaluations. For $P = 50$, that's 10,000 cost evaluations per Hessian — generally prohibitive.

The Hessian eigenvalues at a critical point classify it:

- **All eigenvalues positive**: local minimum.
- **All negative**: local maximum.
- **Mixed signs**: saddle point. The number of negative eigenvalues is the **Morse index** of the saddle.
- **Some zero eigenvalues**: degenerate critical point. Often signals a symmetry or gauge direction.

The **largest eigenvalue** of the Hessian is the **local Lipschitz constant** of the gradient and bounds how large a gradient-descent step you can take without overshooting.

The **condition number** of the Hessian (ratio of largest to smallest eigenvalue) determines convergence rate of gradient descent — large condition numbers mean slow convergence in the worst direction.

The Quantum Fisher Metric

The natural metric on the ansatz manifold is the **quantum Fisher information matrix** (QFIM):

$$g_{ij}(\boldsymbol{\theta}) = 4 \cdot \mathrm{Re}\left[\langle\partial_i\psi|\partial_j\psi\rangle - \langle\partial_i\psi|\psi\rangle\langle\psi|\partial_j\psi\rangle\right].$$

This is the **Fubini-Study metric** pulled back through the parameterization, scaled to make the Cramér-Rao bound clean.

The QFIM measures how distinguishable nearby parameter values are in terms of any possible quantum measurement. Two parameter directions with small g_{ii} produce nearly indistinguishable states; two with large g_{ii} produce very different states.

Crucially, the QFIM is **distinct from the Hessian of the cost function**. The Hessian depends on the cost function (and therefore on the Hamiltonian); the QFIM depends only on the ansatz.

The natural gradient update rule (Module 6) uses the QFIM to rescale the cost gradient:

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

This produces updates that are *invariant under reparameterization*: changing the way you label parameters doesn't change the trajectory in state space. Vanilla gradient descent does not have this property and can be wildly different under different parameterizations of the same ansatz.

Local Curvature and Step Size

The relationship between gradient magnitude and Hessian curvature determines the appropriate step size for gradient descent.

For a quadratic approximation $C(\boldsymbol{\theta}_k + \mathbf{d}) \approx C(\boldsymbol{\theta}_k) + \nabla C^T \mathbf{d} + \frac{1}{2}\mathbf{d}^T \mathbf{H} \mathbf{d}$, the optimal step in the gradient direction is

$$\eta^* = \frac{\|\nabla C\|^2}{\nabla C^T \mathbf{H} \nabla C}.$$

If you take a smaller step you make less progress; if you take a larger step you may overshoot. Adaptive learning rate methods (Adam, RMSProp) approximate this implicitly by scaling each component by an estimate of recent gradient variance, which is correlated with curvature.

In VQA practice, **the Hessian varies significantly across parameter space**, so a single global step size is rarely optimal. This is one reason adaptive methods often outperform vanilla gradient descent on VQAs.

Why Local Geometry Doesn't Tell You Everything

A subtle but important point: **local geometry is not sufficient to navigate a non-convex landscape.**

Even with perfect gradient and Hessian information at every point, a local optimizer: - Cannot tell whether the current basin is the global one. - Cannot find escape routes to better basins. - Cannot distinguish a long valley from a saddle. - Cannot detect concentration plateaus (Module 4) until it's already in one.

This is why **global** information — coming from population methods, surrogate models, or symbolic analysis — is often needed for hard VQA instances. Local geometry is the optimizer's microscope; it tells you the immediate slope but not the overall topology.

The thesis's choice of HOPSO as a representative population-based optimizer (Module 4-5 of the thesis) is motivated in part by this observation: HOPSO maintains a population of points that collectively sample the global structure, while each individual member follows local gradient information.

Structural Role

This node is the **local-structure** node of Module 3. It connects the topological vocabulary (3.2-3.4) to the gradient-computation machinery (3.8) and forwards to the statistical phenomena (3.6, 3.7) and barren plateaus (Module 4).

It is also the place where the QFIM is introduced informally; Module 6 will give it the full formal treatment as the source of the natural gradient.

Relations to Other Concepts

- **Calculus of variations** — first- and second-order necessary conditions for optimality.
- **Information geometry** — the Fisher metric and its quantum analogue.
- **Riemannian optimization** — optimizing on manifolds with non-trivial metrics.
- **Lipschitz constants** — the bound on gradient change that controls step size.
- **Newton's method** — second-order optimization that uses the inverse Hessian.

Worked Example — Gradient Of The Single-Qubit $H = Z$ Cost

For $|\psi(\theta)\rangle = R_Y(\theta)|0\rangle$ and $H = Z$:

$$C(\theta) = \cos(\theta), \quad C'(\theta) = -\sin(\theta), \quad C''(\theta) = -\cos(\theta).$$

At $\theta = 0$: $C = 1$, $C' = 0$, $C'' = -1$. Local maximum. At $\theta = \pi$: $C = -1$, $C' = 0$, $C'' = 1$. Local minimum (also global). At $\theta = \pi/2$: $C = 0$, $C' = -1$, $C'' = 0$. Inflection point.

Gradient magnitude at $\theta = \pi/2$ is 1, the largest possible. At $\theta = 0$ and $\theta = \pi$ it is 0.

The Hessian eigenvalue at $\theta = \pi$ is $+1$, so the local Lipschitz constant of the gradient is 1, and gradient descent with step size $\eta < 1$ is guaranteed to converge near this minimum.

For shot-noisy estimation with $N = 100$ shots and $\mathrm{Var}[\widehat{Z}] = 1 - \cos^2(\theta) = \sin^2(\theta)$, the noise floor is $\sin(\theta)/\sqrt{100} = 0.1\sin(\theta)$. At θ near π, this floor approaches 0 — the noise vanishes near the eigenstate. But the gradient also vanishes there, so the SNR doesn't necessarily improve.

This is a clean toy example of the relationship between gradient magnitude, curvature, and shot noise. **The optimizer is happy near the minimum because both signal and noise are small in absolute terms — what matters is their ratio.**

Visualization

Pending. A 1D plot of $C(\theta) = \cos(\theta)$ over $[-\pi, \pi]$, with overlay annotations: gradient arrow at $\theta = \pi/2$, curvature label at $\theta = \pi$, noise band around the curve. Caption: "Gradient and curvature, the optimizer's microscope."

Exercises

1. For $C(\theta_1, \theta_2) = \cos(\theta_1)\cos(\theta_2)$, compute the Hessian at the four critical points $(0,0), (0,\pi), (\pi,0), (\pi,\pi)$. Classify each.
2. For the same function, compute the QFIM g_{ij} for the parameterization $R_Y(\theta_1) \otimes R_Y(\theta_2)|00\rangle$. Compare to the Hessian of C. Are they the same? Why or why not?
3. (VQA) Estimate the typical gradient magnitude in a random HEA with 10 qubits and 5 layers from the Pauli-shift formula (without computing the full barren plateau theorem). Compare to the shot noise floor at $N = 1024$ shots per evaluation.

Research Extensions

- **Curvature-aware schedules** — using local Hessian information to adapt the step size online.
- **Approximate natural gradient** — block-diagonal or low-rank approximations to the QFIM that are cheap to compute.

- **Hessian estimation via parameter shifts** — efficient protocols that don't require $O(P^2)$ circuit runs.
- **Landscape topography from local probes** — reconstructing the global topology from local gradient/Hessian samples.

Cross-links to Other Courses

- **Calculus / Differential Geometry** — gradients, Hessians, Riemannian metrics.
- **Information Geometry** — Fisher information, quantum Fisher information.
- **Numerical Optimization** — Newton's method, quasi-Newton, trust regions.
- **Module 6 (Optimization Dynamics)** — the natural gradient and other geometric methods.

Concentration of Measure

Position in Knowledge Graph

System: Variational Quantum Algorithms **Structural Domain:** Cost Landscapes **Node:** 3.6

This node is the **statistical** view of VQA cost landscapes. Where node 3.5 looked at the gradient at a single point, this node asks: **what does the gradient look like across all points, on average?**

The answer reveals one of the fundamental obstacles to scalable VQA optimization: in high-dimensional Hilbert spaces, almost all states look the same to almost all observables. This phenomenon is called **concentration of measure** and it is the deeper mathematical reason behind barren plateaus, the difficulty of random initialization, and several other VQA pathologies.

This node introduces concentration without yet doing the full barren plateau theorem (Module 4). The point here is to make the *general* phenomenon intuitive, so that the *specific* result in Module 4 lands as an instance of a broader pattern.

What Concentration Means

Concentration of measure is the observation that, in high-dimensional spaces with appropriate random structure, smooth functions take values very close to their mean almost everywhere.

The classical version: if $f : S^{n-1} \to \mathbb{R}$ is a 1-Lipschitz function on the unit sphere in $\mathbb{R}^n$, then for a uniformly random point $\mathbf{x}$ on the sphere,

$$\mathbb{P}\big[|f(\mathbf{x}) - \mathbb{E}[f]| > t\big] \leq 2\exp(-cnt^2).$$

For large n, this probability is very small unless t is very small. **Almost all of the sphere is mapped to a tiny window around the mean.**

This is counterintuitive because in low dimensions ($n = 2, 3$) it doesn't happen much — a sphere in 3D has plenty of room for a function to take varied values. But as n grows, the volume of the sphere increasingly concentrates near any equator, and any smooth function "averages out" over that volume.

The Quantum Version

For an n-qubit Hilbert space, the unit sphere is $S^{2\cdot 2^n - 1}$ — a sphere of dimension $2 \cdot 2^n - 1$, exponentially large in n.

If you draw a random pure state $|\psi\rangle$ from the **Haar measure** (the unique unitarily invariant probability measure on this sphere), then for any observable O with eigenvalue range $[\lambda_{\min}, \lambda_{\max}]$:

$$\mathbb{P}_{\psi\sim\text{Haar}}\big[|\langle\psi|O|\psi\rangle - \text{Tr}(O)/2^n| > t\big] \leq 2\exp(-c \cdot 2^n \cdot t^2/(\lambda_{\max} - \lambda_{\min})^2).$$

That is: **a Haar-random quantum state has expectation value almost exactly equal to** $\text{Tr}(O)/2^n$ **for almost any observable**, with concentration that gets exponentially tight as the qubit count grows.

For traceless observables like Pauli strings, $\text{Tr}(O)/2^n = 0$, so:

> A Haar-random state on n qubits has $\langle P\rangle \approx 0$ for every Pauli string P, with the typical deviation scaling as $2^{-n/2}$.

This is the *raw* concentration phenomenon for quantum states.

How Concentration Becomes a Cost Landscape Problem

When you put concentration together with the structure of an expressive ansatz, you get:

Step 1: A sufficiently deep, sufficiently random ansatz produces an approximate **2-design** — its parameter distribution induces a state distribution that approximates Haar measure on the relevant subspace.

Step 2: Therefore, for a typical random parameter vector $\boldsymbol{\theta}$, the expectation value $\langle\psi(\boldsymbol{\theta})|H|\psi(\boldsymbol{\theta})\rangle$ is concentrated near its mean $\mathrm{Tr}(H)/2^n$, with deviation $O(2^{-n/2})$.

Step 3: Variations in the parameter vector also produce variations in the cost — but those variations are also subject to concentration. The directional derivative of the cost is *also* a random variable concentrated near zero.

Step 4: Quantitatively, the gradient variance scales as $O(2^{-2n})$ (and the gradient norm as $O(2^{-n})$). This is the **barren plateau result** in its standard form.

Step 5: An optimizer drawing samples from this landscape sees gradients that are below the shot-noise floor for any reasonable budget, and progress stalls.

The barren plateau is concentration of measure made operationally lethal.

What Doesn't Concentrate

Not every ansatz produces concentrated gradients. The key escape routes are:

1. Restricted manifolds. If the ansatz only reaches a small subspace of Hilbert space (e.g., particle-number-conserving states for a fermionic Hamiltonian), the relevant measure is on that subspace, not the full sphere. The exponent in the concentration bound becomes |subspace| rather than 2^n, and concentration is much milder. UCCSD avoids barren plateaus precisely for this reason.

2. Local cost functions. If the cost is built from observables acting on $O(\log n)$ qubits, the concentration scales differently — the relevant Hilbert space is the small subsystem, and the concentration penalty is mild. Cerezo et al. (2021) proved that local cost functions admit gradients of order $\mathrm{poly}(1/n)$ rather than $O(2^{-n})$.

3. Shallow circuits. A circuit shallower than $O(\log n)$ cannot scramble enough to approximate a 2-design. Its state distribution is *structured*, not Haar-uniform, and the gradient does not concentrate as severely. Module 4 examines the threshold depth precisely.

4. Non-random initialization. If you start the optimizer at a non-random parameter (e.g., a Hartree-Fock-equivalent point or a warm start from a classical surrogate), you can avoid the random regime entirely. The barren plateau is a property of the *typical* point, not every point.

These escape routes are not always available, but when they are, they are the right way to design around the problem.

A Useful Mental Picture

Imagine an exponentially large sphere in some abstract space. On that sphere, almost all of the area is concentrated near any "equator" — a great-circle slice.

Now imagine you parameterize a curve on the sphere (the ansatz manifold) and drop a random point. With overwhelming probability, that point is near the equator. Worse, the curve's tangent at that point is nearly parallel to the equator, so moving along the curve doesn't move you away from the equator.

If your cost function is $f(\mathbf{x}) =$ signed distance from the equator, then: - The cost at a random point is near zero (concentration). - The gradient of the cost in the direction of curve motion is near zero (because the tangent is parallel to the equator). - All of this gets exponentially worse as the dimension grows.

That is the geometric picture of a barren plateau. **It is not that the answer isn't there. It is that the answer is overwhelmed by the volume of "everything else."**

Concentration In Other Disciplines

The same phenomenon appears in many other fields under different names:

- **Random matrix theory**: spectral measures of random matrices concentrate around a deterministic limit (Wigner semicircle, Marchenko-Pastur).
- **Statistical mechanics**: thermodynamic quantities concentrate near their ensemble averages in the thermodynamic limit.
- **High-dimensional statistics**: empirical estimators concentrate around population values.
- **Deep learning**: the loss landscape of deep neural networks at random initialization exhibits concentration phenomena that affect trainability ("vanishing gradients").

The pattern is universal: **as dimension grows, randomness produces uniformity**. VQAs are subject to this in a particularly sharp way because Hilbert space dimension grows exponentially with qubit count.

Why This Is The Deep Reason Things Are Hard

If you internalize concentration of measure as the underlying phenomenon, you understand barren plateaus, the difficulty of random initialization, the importance of inductive bias, and the tradeoff between expressivity and trainability — all as instances of a single mathematical fact.

This is the kind of reframing that the thesis's "stochastic objective generator" abstraction (1.8) was setting up: the optimizer sees a noisy scalar function whose statistical properties are determined by deep structural facts about the ansatz and the Hamiltonian. **Concentration of measure is one of the most important of those facts.**

It is also why "just use a more clever optimizer" cannot fix a barren plateau: the concentration is in the *function*, not in the optimizer. No amount of optimizer cleverness can detect a signal that isn't there. The fix has to come from the ansatz side or the cost side.

Structural Role

This node is the **statistical foundation** for Module 4 (barren plateaus), and an important conceptual node in its own right. It also explains, *in advance*, why several of Module 5's regularization techniques work — they push the optimizer into regions of parameter space where the concentration is mild.

Relations to Other Concepts

- **Lévy's lemma** — the canonical concentration inequality for spheres.
- **Talagrand's inequality** — the metric-space generalization.
- **Haar measure on $U(d)$** — the unique unitarily invariant probability measure on the unitary group.
- **t-designs** — a discrete approximation to Haar measure that captures the lowest t moments.
- **Cramér-Rao bound** — the information-theoretic lower bound that concentration phenomena saturate.

Worked Example — Concentration On A 4-Qubit Sphere

Take $n = 4$ qubits, so the relevant unit sphere has dimension $2 \cdot 16 - 1 = 31$. Draw a Haar-random state $|\psi\rangle$ and compute $\langle Z_0 \rangle$.

The mean over Haar is zero ($\mathrm{Tr}(Z_0)/16 = 0$). The variance over Haar is

$$\mathrm{Var}_{\mathrm{Haar}}[\langle Z_0 \rangle] = \frac{1}{2^n + 1} = \frac{1}{17} \approx 0.059.$$

So the standard deviation is $\sqrt{0.059} \approx 0.24$. For $n = 10$, the variance becomes $1/1025 \approx 10^{-3}$ and the std dev is 0.03. For $n = 20$, variance 10^{-6}, std dev 10^{-3}. For $n = 30$, variance 10^{-9}, std dev 3×10^{-5}.

At 30 qubits, a Haar-random state is essentially indistinguishable from one with $\langle Z_0 \rangle = 0$ using fewer than 10^9 shots. **The signal of any single Pauli expectation drowns in the volume of Hilbert space.**

This is the operational meaning of concentration: as the system size grows, you need exponentially many measurements to extract any structural information about a typical random state.

Visualization

Pending. Two-panel plot. Left: histogram of $\langle Z \rangle$ over many Haar-random states for $n = 2, 5, 10, 20$ qubits, showing the distribution narrowing dramatically. Right: log-log plot of $\mathrm{StdDev}[\langle Z \rangle]$ *vs n, showing exponential decay. Caption: "Concentration of measure: more qubits, less variation."*

Exercises

1. Show that for $n = 1$ (single qubit), the variance of $\langle Z \rangle$ over Haar-random states is $1/3$. (Hint: parameterize the state on the Bloch sphere and integrate.)
2. The concentration bound predicts variance $\sim 1/2^n$. Verify numerically by drawing Haar-random states for $n = 4, 6, 8$ qubits and computing the empirical variance of $\langle Z_0 \rangle$.
3. (VQA) Why does *local* cost (using only an observable on $O(\log n)$ qubits) avoid the worst of the concentration bound? Sketch the argument.

Research Extensions

- **Concentration in t-designs** — sharper concentration results for the actual distributions induced by random circuits.
- **Anti-concentration ansätze** — ansatz designs that provably avoid the concentration regime.
- **Initialization strategies** — distributions over parameters that avoid Haar-typical states.
- **Concentration in the presence of noise** — does noise help or hurt the concentration phenomenon?

Cross-links to Other Courses

- **Probability Theory** — concentration inequalities, Lévy's lemma.
- **Random Matrix Theory** — spectral concentration.
- **Information Theory** — typicality, asymptotic equipartition.
- **Module 4 (Barren Plateaus)** — the specific application to VQA gradients.

Shot Noise as Cost Stochasticity

Position in Knowledge Graph

System: Variational Quantum Algorithms **Structural Domain:** Cost Landscapes **Node:** 3.7

So far in Module 3, the cost function has been treated as a *deterministic* scalar function on parameter space. This node introduces the **stochastic** view: in practice the optimizer never sees $C(\boldsymbol{\theta})$ — it sees a noisy estimate $\widehat{C}(\boldsymbol{\theta})$ corrupted by shot noise.

The relationship between C and $\widehat{C}$ is the relationship between **what the landscape really is** and **what the optimizer thinks it is**, and a lot of VQA practice consists of managing the gap between the two.

This node lays out the structure of that gap.

The Two Cost Functions

Let's be explicit.

True cost:

$$C(\boldsymbol{\theta}) = \langle\psi(\boldsymbol{\theta})|H|\psi(\boldsymbol{\theta})\rangle.$$

A deterministic function of $\boldsymbol{\theta}$. What you'd compute from a perfect statevector simulation.

Estimated cost:

$$\widehat{C}(\boldsymbol{\theta}) = \sum_j h_j \widehat{\langle P_j\rangle}, \quad \widehat{\langle P_j\rangle} = \frac{1}{N_j}\sum_{s=1}^{N_j} \widehat{e}_j^{(s)}.$$

A random variable. What the optimizer actually receives. $\mathbb{E}[\widehat{C}(\boldsymbol{\theta})] = C(\boldsymbol{\theta})$ (unbiased estimator, in the absence of hardware noise).

The optimizer's job is to find $\arg\min_{\boldsymbol{\theta}} C(\boldsymbol{\theta})$, but its only access to C is through repeated samples of $\widehat{C}$. This is a stochastic optimization problem in the textbook sense.

The Variance Structure

The variance of $\widehat{C}(\boldsymbol{\theta})$ is

$$\mathrm{Var}[\widehat{C}(\boldsymbol{\theta})] = \sum_j \frac{h_j^2(1-\langle P_j\rangle^2)}{N_j}.$$

This formula has several important features:

1. Inverse in shot count. Doubling N_j halves the variance contribution from term j.

2. Proportional to coefficient squared. Terms with large $|h_j|$ contribute more variance per shot.

3. State-dependent. The factor $(1-\langle P_j\rangle^2)$ means that variance is *minimized* when $\langle P_j\rangle = \pm 1$ (the state is an eigenstate of P_j) and *maximized* when $\langle P_j\rangle = 0$. **The variance depends on $\boldsymbol{\theta}$.**

The third point is the most important: **the noise on $\widehat{C}$ is not uniform across parameter space**. Some regions are intrinsically noisier than others, and the optimizer experiences a *changing noise profile* as it moves around.

For chemistry Hamiltonians where the dominant Pauli terms have $h_j \sim 1$, the typical variance contribution per term is order 1 over N shots, and total variance is $\sum_j h_j^2/N$ — easily 0.01-1 for realistic settings. That's

a per-evaluation standard deviation of 0.1-1 hartree, which is large compared to chemical accuracy (~ 0.001 hartree).

Where The Variance Hides The Structure

When the variance of $\widehat{C}$ is large compared to the *local variation* of C, the optimizer cannot distinguish "small motion in $\boldsymbol{\theta}$" from "no motion in $\boldsymbol{\theta}$."

Define the **signal-to-noise ratio** at a point:

$$\mathrm{SNR}(\boldsymbol{\theta}) = \frac{\|\nabla C(\boldsymbol{\theta})\|}{\sqrt{\mathrm{Var}[\widehat{C}(\boldsymbol{\theta})]}}.$$

When SNR is large, the optimizer sees the gradient clearly and can descend. When SNR is near 1, the optimizer's view is dominated by noise and progress is slow. When SNR is much less than 1, the optimizer is essentially doing a random walk.

The dangerous regime is **near critical points and on plateaus**, where the gradient is small but the variance is not. This is exactly where you most want to make small careful steps, and exactly where you can't tell whether your steps are doing anything.

This is the **shot noise floor** that limits VQA optimization in practice. It is not addressed by being smarter; it requires either more shots, or a smaller variance contribution at the relevant points, or better signal extraction.

How Shot Noise Differs From Other Optimization Noise

Shot noise has properties that make it different from generic stochastic optimization noise:

1. State-dependent variance. Unlike the constant Gaussian noise often assumed in stochastic optimization theory, shot noise variance depends on the parameter point. Standard convergence analyses (e.g., Robbins-Monro) assume bounded variance and may not directly apply.

2. Bounded estimator outputs. Each shot gives a value in $[-1, +1]$ for each Pauli, so $\widehat{C}$ has bounded support. This is mostly a feature — sub-Gaussian concentration applies — but it means certain heavy-tailed-noise techniques are not relevant.

3. Correlated across the gradient. When you compute $\nabla\widehat{C}$ by parameter-shift, the same shots are used for shifted evaluations within a single component but independent shots are used across components. This produces a specific (block-diagonal) covariance structure that some optimizers can exploit and others can't.

4. Adaptive in principle. You can choose to spend more shots in some regions of parameter space than others — variance is not a fixed property of the problem, it is a budget allocation. Module 5's "shot-adaptive weighting" technique exploits this.

5. Time-stationary. On simulated hardware, shot noise is i.i.d. across iterations. On real hardware, it can drift due to calibration changes. Most analysis assumes the i.i.d. case.

These features are why VQA optimization is its own subfield rather than a special case of generic stochastic optimization.

Cost-Side vs Optimizer-Side Mitigation

The shot noise problem can be attacked from either side:

Cost-side: reduce the variance of $\widehat{C}$ at fixed shot budget.

- Better measurement grouping (node 2.8) → lower per-shot variance.
- Variance-weighted shot allocation → put shots where they matter most.
- Local cost functions → replace high-variance global observables with low-variance local ones.
- Importance sampling — query terms with large $|h_j|$ more often.

Optimizer-side: make the optimizer robust to whatever noise is there.

- Larger batch sizes (more shots per evaluation) → lower per-evaluation variance.
- Momentum / averaging methods (Adam, EMA gradients) → average over noise across iterations.
- Trust region methods — only commit to a step when the noise is small enough to be confident.
- Population methods (HOPSO) → collectively average over many points.

The thesis argues that *both* sides should be considered, and that the regularization techniques in Module 5 are best understood as the dynamics-side counterpart to the cost-side techniques here.

What Shot Noise Looks Like To The Optimizer

A toy mental model: imagine the true cost surface, then add a random Gaussian field of standard deviation $\sigma(\boldsymbol{\theta})$ on top of it. The Gaussian field is independent at each evaluation but has the same variance structure across evaluations (assuming i.i.d. noise).

What the optimizer sees at any single iteration is: - The true gradient direction, plus - A random gradient component proportional to the local noise level, plus - Anything from the optimizer's own update rule (momentum, etc.).

If the random component dominates the true gradient, the optimizer is essentially walking randomly. If the true gradient dominates, the optimizer is doing approximate gradient descent.

The transition between these regimes happens around SNR = 1, and the location of that transition in parameter space depends on the local landscape and the shot budget. **The optimizer crosses in and out of "useful" and "useless" regions during a single optimization run.**

Structural Role

This is the **stochasticity** node of Module 3 and the bridge to gradient computation (3.8) and to Module 5 (where the regularization framework is built around managing this exact problem).

It is also one of the cleanest places to see the "stochastic objective generator" abstraction (1.8) at work: the optimizer sees a noisy scalar oracle whose properties are determined by upstream physics. The shot noise is the dominant source of that noise on simulated hardware.

Relations to Other Concepts

- **Stochastic optimization theory** — Robbins-Monro, Kiefer-Wolfowitz, SGD, SPSA.
- **Sub-Gaussian random variables** — the formal class to which bounded shot noise belongs.
- **Variance reduction techniques** — control variates, stratified sampling, importance sampling.
- **Bayesian optimization** — explicitly models the stochastic oracle and uses Bayesian inference to decide where to query.
- **Robust optimization** — optimizing in the presence of bounded but unknown perturbations.

Worked Example — Variance At Three Parameter Points

For $|\psi(\theta)\rangle = R_Y(\theta)|0\rangle$ and $H = Z + 0.5X$:

$$C(\theta) = \cos(\theta) + 0.5\sin(\theta).$$

At $\theta = 0$ (state $|0\rangle$): $\langle Z\rangle = 1$, $\langle X\rangle = 0$. Cost = 1. Variance contributions: $(1-1) + 0.25(1-0) = 0.25$. Standard deviation = $0.5/\sqrt{N}$.

At $\theta = \pi$ (state $|1\rangle$): $\langle Z\rangle = -1$, $\langle X\rangle = 0$. Cost = -1. Variance: $0 + 0.25 = 0.25$. Same as above.

At $\theta = \pi/2$ (state $|+\rangle$): $\langle Z\rangle = 0$, $\langle X\rangle = 1$. Cost = 0.5. Variance: $(1-0) + 0.25(1-1) = 1$. Standard deviation = $1/\sqrt{N}$.

So the noise on $\widehat{C}$ varies by factor of 2 across parameter space. At eigenstate-like points (where one of the Pauli terms is saturated), variance is lower. At superposition points, variance is higher.

For an optimizer trying to detect a small gradient near $\theta = \pi/2$, the noise floor is *higher* there than near the minimum, even though the gradient is *also* higher. The SNR may or may not be favorable depending on the exact geometry.

This is the kind of effect that gets averaged away in textbook analyses but matters operationally.

Visualization

Pending. A 1D plot of $C(\theta)$ over $[-\pi, \pi]$, with a noise band of width $\sigma(\theta)$ around the curve, showing variable thickness. Annotation: "noise is wider at superposition points, narrower near eigenstates."

Exercises

1. For $H = Z_0 Z_1$ on 2 qubits and a state $|\psi\rangle = \cos(\theta/2)|00\rangle + \sin(\theta/2)|11\rangle$, compute $\langle ZZ\rangle(\theta)$ and $\mathrm{Var}[\langle ZZ\rangle](\theta)$. Where is the variance maximized?
2. Show that if a cost function is the expectation value of a single observable bounded in $[-1, +1]$, then $\mathrm{Var}[\widehat{C}] \leq 1/N$ where N is the shot count, regardless of the state.
3. (VQA) For an optimizer running with adaptive shot allocation, design a heuristic that increases shot count when SNR is low. What metric would you use to decide?

Research Extensions

- **Variance-aware shot allocation** — dynamically increasing shots in high-noise regions.
- **Importance-weighted estimators** — biasing measurements toward terms with large coefficients.
- **Variance penalization (Module 5.7)** — regularizing the cost to prefer low-variance regions.
- **Bandit-style stopping rules** — terminating evaluation when confidence intervals are tight enough.

Cross-links to Other Courses

- **Statistics** — sample variance, central limit theorem, sub-Gaussian concentration.
- **Stochastic Optimization** — Robbins-Monro, SPSA, Polyak averaging.

- **Module 5 (Regularization)** — the dynamics-side approach to managing this noise.
- **Module 9 (Hardware Noise Models)** — noise sources beyond shot noise.

The Parameter-Shift Rule

Position in Knowledge Graph

System: Variational Quantum Algorithms **Structural Domain:** Cost Landscapes **Node:** 3.8

This is the **closing node** of Module 3 and the place where the gradient computation question becomes concrete.

The parameter-shift rule is the small mathematical fact that makes gradient-based VQA optimization possible: **for ansätze built from Pauli rotations, the gradient of the cost function with respect to any parameter can be computed exactly by evaluating the cost function at two shifted parameter values**, with no need for finite differences or symbolic differentiation through the circuit.

This is a more remarkable result than it sounds. Without it, you would have to estimate gradients from finite differences (which compound shot noise) or differentiate through the entire quantum circuit symbolically (which is infeasible at scale). Parameter-shift gives you exact gradients with shot-statistics-only error — a clean separation that makes the rest of VQA optimization theory tractable.

The Statement

For a parameterized circuit where parameter θ_i appears in a single Pauli rotation gate $R_P(\theta_i) = e^{-i\theta_i P/2}$ (with P a Hermitian operator with eigenvalues ± 1), the gradient of any expectation value $C(\boldsymbol{\theta}) = \langle O \rangle_{\boldsymbol{\theta}}$ is

$$\frac{\partial C}{\partial \theta_i} = \frac{1}{2}\left[C\left(\boldsymbol{\theta} + \frac{\pi}{2}\mathbf{e}_i\right) - C\left(\boldsymbol{\theta} - \frac{\pi}{2}\mathbf{e}_i\right)\right].$$

This is **exact**, not an approximation. There is no $O(\Delta t^2)$ error like in finite differences. The only error in the estimated gradient is shot noise from estimating the two cost values.

The shifts are by exactly $\pi/2$ — not a small step, but a *symmetric* large step that exploits the trigonometric structure of the underlying functional dependence.

The Derivation

Why does this work?

The key fact is that for $P^2 = I$ (which holds for any Pauli string), the rotation gate has the simple expansion

$$R_P(\theta) = \cos(\theta/2)I - i\sin(\theta/2)P.$$

So if the cost function depends on θ_i only through this gate, it has the functional form

$$C(\theta_i) = a + b\cos(\theta_i) + c\sin(\theta_i)$$

for some θ_i-independent constants a, b, c. (Substituting in the rotation expansion and computing $\langle\psi(\theta_i)|O|\psi(\theta_i)\rangle$ gives precisely this form — the cosine and sine come from squaring the rotation amplitudes.)

A function of this form has derivative

$$C'(\theta_i) = -b\sin(\theta_i) + c\cos(\theta_i).$$

Now compute $C(\theta_i + \pi/2) - C(\theta_i - \pi/2)$:

$$C(\theta_i + \pi/2) = a + b\cos(\theta_i + \pi/2) + c\sin(\theta_i + \pi/2) = a - b\sin(\theta_i) + c\cos(\theta_i).$$
$$C(\theta_i - \pi/2) = a - b\cos(-\theta_i + \pi/2) + c\sin(-\theta_i + \pi/2)... = a + b\sin(\theta_i) - c\cos(\theta_i)...$$

(Working it out more carefully:)

$$C(\theta_i + \pi/2) - C(\theta_i - \pi/2) = 2(-b\sin(\theta_i) + c\cos(\theta_i)) = 2C'(\theta_i).$$

So $C'(\theta_i) = \frac{1}{2}[C(\theta_i + \pi/2) - C(\theta_i - \pi/2)]$. **The shift derivative is the exact derivative because the underlying function is a sinusoid.**

The derivation only used: 1. $P^2 = I$ (for the rotation expansion to be sinusoidal). 2. θ_i appears in exactly one place in the circuit.

Both conditions are true for standard Pauli-rotation ansätze, which is why parameter-shift works for them.

Where Parameter-Shift Doesn't Apply

Parameter-shift in its basic form has assumptions, and there are ansätze where it doesn't work.

1. Gates whose generators have more than two distinct eigenvalues. A gate of the form $e^{-i\theta G}$ where G has eigenvalues $\{\lambda_1, \lambda_2, \lambda_3, \dots\}$ gives a cost function that is a sum of multiple sinusoids with different frequencies, not a single sinusoid. The naive 2-shift rule doesn't apply, and you need a *generalized parameter-shift rule* with more shifts to disentangle the frequencies.

For example: a controlled rotation gate has a non-Pauli generator. The generalized rule for it uses 4 shifts.

2. Same parameter appearing in multiple gates. If θ_i appears in two different rotation gates, the function depends on θ_i through a sum of two sinusoids and the 2-shift rule again doesn't suffice.

3. Hardware-specific gate decompositions. If the device decomposes a "logical" $R_P(\theta)$ into a different physical gate sequence (e.g., $e^{-i\theta P/2} \to$ several pulses with θ-dependent durations), parameter-shift may not apply at the pulse level.

4. Non-unitary noise channels. If the circuit includes intermediate measurements or non-unitary operations, the simple algebraic structure breaks down.

For practical VQA work with standard ansätze (HEA, UCCSD, QAOA, HVA), parameter-shift in its basic form covers nearly all cases. The generalized rules are well-understood for the cases it doesn't.

Cost Comparison: Parameter-Shift vs Finite Difference

Parameter-shift: 2 cost evaluations per gradient component. Exact gradient (up to shot noise).

Forward finite difference: 2 cost evaluations per component (one shifted, one baseline). Approximation error $O(\Delta t)$, where Δt is the shift.

Central finite difference: 2 cost evaluations per component. Approximation error $O(\Delta t^2)$.

The **circuit-evaluation cost is the same** (2 evaluations per component) for parameter-shift and for either finite difference variant. The difference is in the **error**: parameter-shift has zero discretization error, while finite difference has $O(\Delta t)$ or $O(\Delta t^2)$ error.

For finite difference, you face the standard tradeoff: - Small Δt: low discretization error but small numerator ($C(\theta + \Delta t) - C(\theta)$ is small) which gets dominated by shot noise. - Large Δt: better signal-to-noise ratio in

the numerator but bigger discretization error. - Optimal Δt is somewhere in the middle and depends on the unknown gradient magnitude and noise level.

Parameter-shift sidesteps this entirely. The shift $\pi/2$ is large enough to give a well-conditioned numerator and produces zero discretization error regardless of the local function behavior. This is the operational reason it dominates finite difference for VQA gradient estimation.

What This Means For The Optimizer

A parameter-shift gradient evaluation for a P-parameter ansatz costs $2P$ cost function evaluations. At $P = 50$, that's 100 evaluations per gradient — and at, say, $M = 10$ Pauli measurement groups $\times$ $N = 1000$ shots per group, each cost evaluation is 10,000 circuit runs, so a single full gradient is **1 million circuit runs**.

This is *much* cheaper than computing the Hessian ($O(P^2)$ gradients $= O(P^3)$ circuit runs) but still substantial.

Three optimizer strategies emerge:

1. Full-gradient methods (BFGS, vanilla gradient descent): pay the $2P$ cost per iteration, take an informed step.

2. Subsampled-gradient methods (SPSA, random coordinate descent): pay only $O(1)$ or $O(\log P)$ cost per iteration but get a noisier gradient estimate. SPSA is famous for getting a good descent direction from just *one* parameter-shift pair, regardless of how many parameters there are.

3. Gradient-free methods (COBYLA, Bayesian optimization, HOPSO): skip parameter-shift entirely and work only from cost evaluations. Each iteration is cheap but informed updates require more iterations.

The choice between these is driven largely by per-evaluation cost — and per-evaluation cost is set by the parameter-shift rule's $2P$ overhead. **The parameter-shift rule is what makes gradient-based VQA optimization possible, and its cost is what makes gradient-free alternatives competitive.**

Closing Module 3

This is the last node of Module 3. With the parameter-shift rule in hand, you have:

- A vocabulary for the topology of cost landscapes (3.2-3.4).
- An understanding of local geometry and gradients (3.5).
- The statistical view (concentration, 3.6).
- The stochastic view (shot noise, 3.7).
- The operational mechanism for gradient computation (3.8, this node).

Module 4 will use all of these to prove and analyze the barren plateau theorem. Module 5 will use them to motivate regularization. Module 6 will use them to design optimization update rules. Modules 7-10 will return to them as needed.

The cost landscape is now fully described.

Structural Role

This is the operational closure of Module 3. It also forwards directly to Module 6, which builds optimizers that use parameter-shift gradients as their primary input.

Relations to Other Concepts

- **Numerical differentiation** — finite differences as the classical alternative to parameter-shift.
- **Automatic differentiation** — computational graph differentiation for general functions; parameter-shift is essentially a closed-form auto-diff for unitary circuits.
- **Generalized parameter-shift rules** — extensions to gates with non-Pauli generators (Wierichs et al. 2022).
- **Stochastic parameter shift** — variants that randomize the shift to avoid certain pathologies.

Worked Example — Verifying Parameter-Shift On A Single Qubit

For $|\psi(\theta)\rangle = R_Y(\theta)|0\rangle$ and $H = Z$, the cost is $C(\theta) = \cos(\theta)$, so $C'(\theta) = -\sin(\theta)$.

Let's verify the parameter-shift rule at $\theta = \pi/4$.

By calculus: $C'(\pi/4) = -\sin(\pi/4) = -1/\sqrt{2} \approx -0.707$.

By parameter-shift:

$$C(\pi/4 + \pi/2) - C(\pi/4 - \pi/2) = \cos(3\pi/4) - \cos(-\pi/4) = -1/\sqrt{2} - 1/\sqrt{2} = -2/\sqrt{2} = -\sqrt{2}.$$

$$\frac{1}{2}[C(\pi/4 + \pi/2) - C(\pi/4 - \pi/2)] = -\sqrt{2}/2 = -1/\sqrt{2} \approx -0.707.$$

Match. The parameter-shift rule gives exactly the analytic derivative. No finite-difference error, no truncation. Just two cost evaluations and an arithmetic combination.

For a real device, each of those two cost evaluations would be a full noisy estimation: prepare $R_Y(3\pi/4)|0\rangle$, measure Z, N shots; then prepare $R_Y(-\pi/4)|0\rangle$, measure Z, N shots; subtract the sample means; divide by 2. The result is an unbiased estimator of the exact gradient with variance $\sim 1/N$.

Visualization

Pending. A 1D plot of $C(\theta) = \cos(\theta)$ with two shifted points $(\theta - \pi/2, C(\theta - \pi/2))$ and $(\theta + \pi/2, C(\theta + \pi/2))$ marked, and a tangent line at θ whose slope is the parameter-shift gradient. Caption: "Two evaluations, exact slope."

Exercises

1. Verify the parameter-shift rule for the cost function $C(\theta) = \langle X \rangle = \sin(\theta)$ at $\theta = \pi/3$.
2. Show that for a function of the form $f(\theta) = a + b\cos(\theta) + c\sin(\theta) + d\cos(2\theta) + e\sin(2\theta)$, the simple 2-shift rule gives an *incorrect* derivative. Sketch the generalized rule (using 4 shifts) that recovers the correct derivative.
3. (VQA) For a 50-parameter ansatz, full-gradient evaluation costs 100 cost evaluations per iteration. SPSA estimates a gradient direction from just 2 cost evaluations per iteration. Sketch the conditions under which SPSA's per-iteration noise is better-traded against the full-gradient cost.

Research Extensions

- **Pulse-level parameter-shift** — extending parameter-shift to ansätze defined at the pulse level rather than the gate level.

- **Reduced-shot parameter-shift** — protocols that use fewer shots per gradient component by reusing measurements.
- **Stochastic parameter-shift** — randomized shifts that improve some statistical properties.
- **Higher-order shift rules** — direct estimation of Hessians via 4-shift rules.

Cross-links to Other Courses

- **Numerical Methods** — finite differences, automatic differentiation.
- **Calculus** — exact derivatives via trigonometric identities.
- **Module 6 (Optimization Dynamics)** — how this gradient is used by optimizers.
- **Module 1 (Node 1.4)** — earlier introduction of the parameter-shift rule from the architecture side.