Gradient Estimation

Variational Quantum Algorithms · Module 1 — Hybrid Quantum Classical Systems

Node 1.4

Position in Knowledge Graph

System: Variational Quantum Algorithms **Structural Domain:** Hybrid Quantum-Classical Systems **Node:** 1.4

This node tackles the next question after "how do you evaluate $C(\boldsymbol{\theta})$": **how do you evaluate $\nabla C(\boldsymbol{\theta})$?**

The answer turns out to be remarkably clean for a wide class of circuits. It is called the **parameter-shift rule**, and it is one of the few genuinely beautiful results in VQA pedagogy. If you remember one technical result from Module 1 beyond the variational principle, remember this one.

The Problem

Gradient-based optimizers (SGD, Adam, L-BFGS, natural gradient) need $\nabla C(\boldsymbol{\theta})$. On a classical computer with a differentiable cost, you get this for free via automatic differentiation. On a quantum computer, autodiff does not work: the "forward pass" is a physical quantum circuit, not a Python trace.

So you are stuck estimating $\partial C/\partial\theta_i$ from black-box evaluations of C. There are two competing strategies.

The Bad Strategy: Finite Differences

The classical numerical-analysis move: pick a small h, compute

$$\frac{\partial C}{\partial \theta_i} \approx \frac{C(\boldsymbol{\theta} + h\mathbf{e}_i) - C(\boldsymbol{\theta} - h\mathbf{e}_i)}{2h}.$$

This is bad for VQA for two reasons.

First, **bias vs variance tradeoff.** Small h reduces the truncation error but amplifies shot noise: the numerator is a small difference of two noisy quantities, and the variance of the estimator scales as $1/h^2$. Large h reduces noise amplification but increases truncation bias. You cannot win both.

Second, **it does not have to be approximate.** VQAs have a structural property — periodicity of Pauli-rotation gates — that allows gradients to be computed **exactly** (up to shot noise) with two function evaluations. The parameter-shift rule.

The Good Strategy: The Parameter-Shift Rule

Claim. If the cost function has the form

$$C(\boldsymbol{\theta}) = \langle 0|U^\dagger(\boldsymbol{\theta}) H U(\boldsymbol{\theta})|0\rangle,$$

and the parameter θ_i enters through a single-qubit Pauli rotation $R_P(\theta_i) = \exp(-i\theta_i P/2)$, then

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

This is exact. No Taylor expansion, no small-h limit. The only noise in the estimate comes from shot noise in the two function evaluations, not from truncation.

Why It Works — The Derivation Sketch

For a Pauli rotation $R_P(\theta) = \cos(\theta/2)I - i\sin(\theta/2)P$ (recall $P^2 = I$), a direct calculation gives

$$\frac{d}{d\theta}R_P(\theta) = -\frac{i}{2}P \cdot R_P(\theta).$$

Plugging into $C(\theta) = \langle 0|R_P^\dagger(\theta) A R_P(\theta)|0\rangle$ (for some effective observable A that absorbs everything else) and using the commutator identity $[P, A] = \frac{1}{i}\big(R_P(\pi/2)AR_P(-\pi/2) - R_P(-\pi/2)AR_P(\pi/2)\big)$, the derivative collapses to a difference of two cost evaluations at shifted angles.

The full derivation is in Mitarai et al. 2018 and Schuld et al. 2019. The key ingredient is that P has only two distinct eigenvalues (± 1), which constrains the Fourier decomposition of $C(\theta)$ to contain only the two frequencies ± 1. The shift rule is effectively a **trigonometric interpolation identity** evaluated at sample points that the structure of Pauli rotations makes exact.

The Cost

Two circuit evaluations per parameter. For a circuit with P parameters, one full gradient costs $2P$ evaluations. Each evaluation itself costs M measurement circuits (one per commuting Pauli group) times N shots.

So the per-gradient shot budget is roughly $2PMN$. For a 40-parameter circuit, 50 Pauli terms, 1024 shots: **4.1 million shots per gradient**. This is why shot efficiency matters.

Alternatives When Parameter-Shift Is Too Expensive

SPSA (Simultaneous Perturbation Stochastic Approximation). Estimates the whole gradient in two circuit evaluations, independent of P. The catch: the estimate is high-variance, biased against individual components, and works only in expectation. But for very high-dimensional circuits where $2P$ evaluations is prohibitive, SPSA is the canonical choice.

Gradient-free methods. COBYLA, Nelder-Mead, and similar methods avoid gradients entirely. They scale poorly with P but are robust to noise. Often competitive for small circuits.

Natural gradient. Uses a Fisher-information-weighted gradient direction. Requires extra circuit evaluations to estimate the quantum Fisher information matrix (see Module 6), but converges faster per iteration in many problem classes.

Structural Role

This node is where the **parameter-shift rule** enters the course, and it is a pivotal result. It is what makes gradient-based optimization viable for VQA at all. Every time node 1.5 or later refers to "a gradient step," the gradient is being estimated by the shift rule under the hood.

It is also the technical detail that node 1.6 (VQA as system identification) depends on: the shift rule gives the optimizer a noisy but unbiased gradient oracle, which is exactly what first-order sysid methods need.

Relations to Other Concepts

- **Automatic differentiation** — the classical-ML tool the shift rule is trying to replace.
- **Finite-difference methods** — the naive alternative. Worse variance, worse bias, same cost.
- **Simultaneous Perturbation Stochastic Approximation (SPSA)** — Spall 1992; independent of parameter count.
- **Hadamard test** — another quantum primitive for estimating derivatives, via controlled gates. Higher circuit-depth overhead than the shift rule.
- **Quantum natural gradient** — Stokes et al. 2020; uses Fubini-Study metric.

Worked Example — Parameter-Shift on a One-Qubit RY Circuit

Consider

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

With $H = Z$, the cost is $C(\theta) = \cos\theta$ and the exact gradient is $-\sin\theta$.

Apply the shift rule:

$$\frac{\partial C}{\partial \theta} \stackrel{?}{=} \frac{1}{2}\big[\cos(\theta + \pi/2) - \cos(\theta - \pi/2)\big] = \frac{1}{2}\big[-\sin\theta - \sin\theta\big] = -\sin\theta. \checkmark$$

Exact, not approximate. Every time you use the shift rule in VQE or QAOA, this is what is happening under the hood.

Visualization

*Pending. Diagram: cost function $C(\theta)$ plotted over one period. Two evaluation points shown at $\theta \pm \pi/2$. Arrow between them, labeled with the shift-rule formula. Annotate that the slope of the chord between these two points **equals** the tangent slope at θ — exactly.*

Exercises

1. Verify the parameter-shift rule for $R_X(\theta)$ acting on $|0\rangle$ with observable $H = Z$. Compute both sides of the identity and show they agree.
2. Derive the variance of the shift-rule gradient estimator assuming each evaluation has shot variance σ^2. Compare to the variance of a central finite difference with step h.
3. (VQA) For a Hamiltonian with $M = 100$ Pauli terms and a circuit with $P = 20$ parameters, estimate the shot cost of one full parameter-shift gradient at 1024 shots per measurement. Compare to SPSA.

Research Extensions

- **Higher-order parameter shifts** — for gates whose generators have more than two distinct eigenvalues, the shift rule generalizes to more than two evaluation points (Wierichs et al. 2022).
- **Stochastic parameter-shift** — sampling parameters to estimate gradients; trades bias for variance.
- **Differentiable quantum circuits** — autodiff frameworks (PennyLane, TorchQuantum) that compose shift-rule gradients with classical autodiff.

Cross-links to Other Courses

- **Calculus / Numerical Analysis** — finite differences, truncation error, bias-variance tradeoffs.
- **Cost Landscapes (Module 3)** — gradient information as a probe of landscape geometry.
- **Optimization Dynamics (Module 6)** — how gradient estimators interact with update rules.

Variational Quantum Algorithms · Module 1 — Hybrid Quantum Classical Systems · Node 1.4
Concepts: gradient-descent, parameterized-circuits, expectation-value
Prerequisites: 1.3
Enables: 1.5, 1.6

hbar.university — own.your.cognition