Benchmarking Classical Optimization Algorithms

From First Principles to Quantum Applications

Karim El Tohamy

Scientific Computing (CIT644)

Motivation: The Challenge of VQE Landscapes

Context

- Optimization is the computational bottleneck in Variational Quantum Eigensolvers (VQE).
- Real-world landscapes are rarely simple bowls.

The "Ill-Conditioned" Gauntlet

1. **Stiffness:** Narrow, curved valleys.
2. **Saddle Points:** Traps for second-order methods.
3. **Local Traps:** Basins that prevent finding the global minimum.

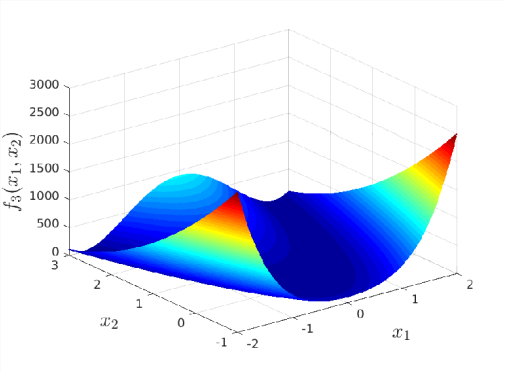


Figure 1: The Rosenbrock "Banana" Valley

Methodology: The "Black Box" Engine

Constraint: Analytic derivatives are assumed unavailable (simulating a QPU).

1. Automatic Numerical Differentiation

We implemented a **Central Finite Difference** engine ($\epsilon = 10^{-8}$).

$$f'(x) \approx \frac{f(x+\epsilon) - f(x-\epsilon)}{2\epsilon}$$

Hessians are computed via recursive finite difference.

2. Validation Strategy

- Benchmarked against **SciPy** (Newton-CG, BFGS, Nelder-Mead).
- Benchmarked against **PyTorch** (Adam).
- Validated on 7 pathological test functions.

Strategy 1: Adam (Adaptive Moment Estimation)

Type: Adaptive First-Order Method

Concept: Uses "Momentum" to push through flat valleys and "Adaptive Learning Rates" to scale steps for each parameter individually.

Mathematical Formulation

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

$$\theta_{t+1} = \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

Execution Steps

1. Calculate Gradient g_t.
2. Update biased moments m_t (momentum) and v_t (velocity).
3. **Bias Correction:** Scale moments to account for initialization at 0.
4. Update parameters.

Implementation Analysis: Adam

Comparison: Custom Adam vs. PyTorch Adam
Difference: Negligible ($< 10^{-8}$).

Conclusion: Our implementation of the bias-correction logic matches the industry standard perfectly.

Feature	Impact
Momentum	Allows Adam to maintain velocity in flat regions where Gradient Descent stalls.
Adaptive Rate	Prevents divergence on steep walls (Rosenbrock) while speeding up on plateaus.

Strategy 2: Newton's Method

Type: Exact Second-Order Method

Concept: Uses the Hessian Matrix (Curvature) to approximate the function as a quadratic bowl and jump directly to the minimum.

Mathematical Formulation

$$\mathbf{x}_{k+1} = \mathbf{x}_k - [\mathbf{H}(\mathbf{x}_k)]^{-1} \nabla f(\mathbf{x}_k)$$

**Implemented via Linear Solve:*

$$\mathbf{H} \Delta \mathbf{x} = -\mathbf{g}$$

Execution Steps

1. Calculate Gradient ∇f.
2. Calculate Hessian $\mathbf{H}$ (Numerical).
3. Solve linear system for step direction $\Delta \mathbf{x}$.
4. Update position: $\mathbf{x}_{new} = \mathbf{x} + \Delta \mathbf{x}$.

Implementation Analysis: Newton vs. SciPy

Comparison: Custom (Exact) vs. SciPy (Truncated)

- **Custom Newton:** Solves the exact linear system ($\mathbf{H}^{-1}\mathbf{g}$).
- **SciPy Newton-CG:** Approximates the step using Conjugate Gradient.

Feature	Custom (Exact)	SciPy (Truncated)
Efficiency	**Faster** per-step (7 iters).	Slower per-step (84 iters).
Scalability	Poor ($O(N^3)$ matrix inversion).	Excellent (Avoids inversion).
Stability	Fragile (Attracted to Saddle Points).	**Robust** (Line Search).

Strategy 3: BFGS (Quasi-Newton)

Type: Quasi-Newton Method

Concept: Avoids the expensive $O(N^3)$ Hessian inversion. It iteratively builds an *approximation* of the Inverse Hessian using gradient history.

Sherman-Morrison Update Rule

$$\mathbf{B}_{k+1} = (I - \rho s y^T)\mathbf{B}_k(I - \rho y s^T) + \rho s s^T$$

where $s = \Delta x$ and $y = \Delta \text{gradient}$.

Execution Steps

1. Calc search direction $p = -Bg$.
2. Take Step ($\alpha = 1.0$).
3. Calculate new gradient.
4. Update Matrix B using the formula.

Implementation Analysis: BFGS

Comparison: Fixed Step vs. Line Search

- **Custom BFGS:** Uses a fixed step size $\alpha = 1.0$ (Textbook definition).
- **SciPy BFGS:** Uses a Backtracking Line Search (Wolfe Conditions).

The Consequence:

- **Custom:** Can take massive, unstable steps ("Explosions") on rugged landscapes before recovering.
- **SciPy:** Guarantees monotonic descent at every step, but requires more function evaluations per iteration.

Strategy 4: Nelder-Mead (Simplex)

Type: Gradient-Free Direct Search

How it Works: It maintains a geometric shape (simplex) of $N + 1$ points. It navigates without derivatives by comparing function values at the vertices.

Geometric Operations

- **Reflect:** Mirror the worst point.
- **Expand:** Stretch if direction is good.
- **Contract/Shrink:** Shrink if trapped.

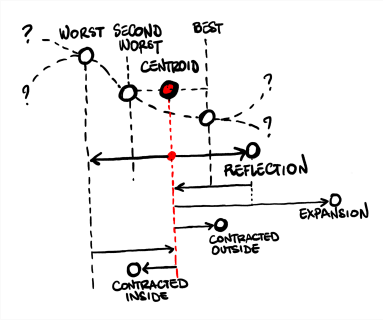


Figure 2: Simplex Transformations

Implementation Analysis: Nelder-Mead

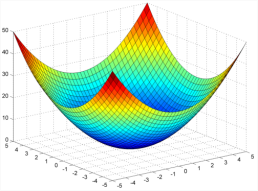
Comparison: Textbook vs. Adaptive

- **Custom NM:** Uses standard coefficients ($\alpha = 1, \gamma = 2, \rho = 0.5, \sigma = 0.5$).
- **SciPy NM:** Uses adaptive coefficients based on problem dimension.

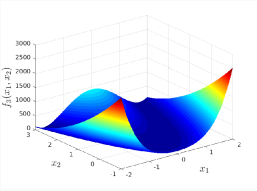
Key Observation: Both implementations performed similarly on our benchmarks, confirming that the core geometric logic is robust. It remains the best fallback when gradients are unavailable.

The Benchmark Suite: Visualizing the Landscapes

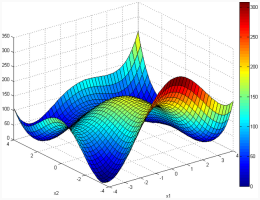
We tested the optimizers on 7 diverse functions to probe specific weaknesses.



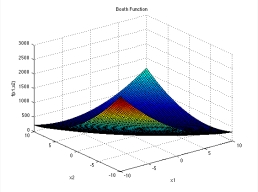
Sphere
(Convex)



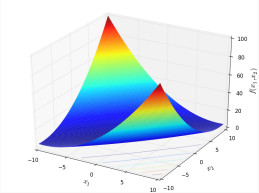
Rosenbrock
(Stiff Valley)



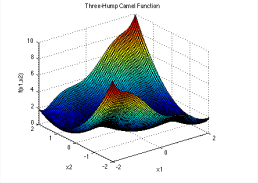
Himmelblau
(Multi-Modal)



Booth
(Plate Valley)



Matyas
(Rotated Bowl)



Three-Hump
(Local Traps)

Benchmark 1: Sphere (Convex Baseline)

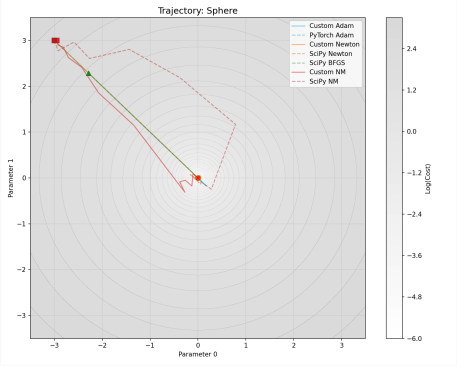


Figure 3: Quadratic vs. Linear Convergence

Analysis

- **Newton/BFGS:** Converged in < 5 iterations (Vertical drop).
- **Adam:** Linear convergence (Slow).

Method	C-Cost	Ref-Cost	C-Iter	R-Iter
Adam	5.62×10^{-9}	8.93×10^{-9}	200	200
Newton	3.94×10^{-14}	$\mathbf{7.88 \times 10^{-46}}$	**2**	**2**
BFGS	7.85×10^{-15}	7.53×10^{-16}	3	3
NM	1.69×10^{-13}	9.46×10^{-10}	58	44

Benchmark 2: Rosenbrock (The Stiffness Problem)

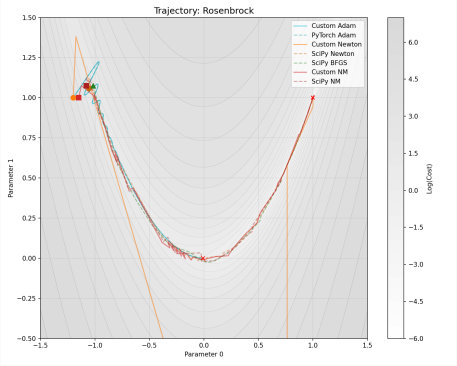


Figure 4: Newton cuts through; Adam follows curve

Analysis

- **Newton:** Solved in **7 iterations**.
- **Adam:** Navigated the valley successfully.

Method	C-Cost	Ref-Cost	C-Iter	R-Iter
Adam	9.98×10^{-1}	$1.02 \times 10^{+0}$	200	200
Newton	1.84×10^{-19}	2.46×10^{-9}	**7**	84
BFGS	$\mathbf{5.64 \times 10^{-20}}$	3.35×10^{-12}	80	31
NM	3.77×10^{-13}	8.18×10^{-10}	113	85

Benchmark 3: Himmelblau (Saddle Points)

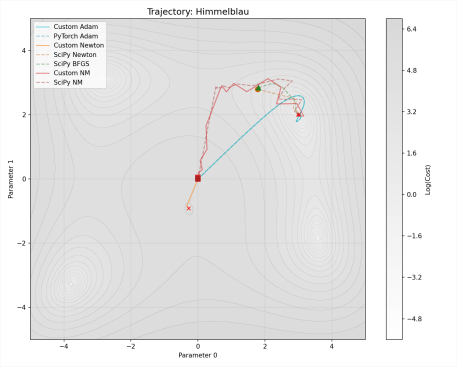


Figure 5: Newton Trapped vs. SciPy Escape

Analysis

- **Custom Newton:** Failed (Cost 182). Attracted to a **Saddle Point**.
- **SciPy:** Succeeded (Trust Regions).

Method	C-Cost	Ref-Cost	C-Iter	R-Iter
Adam	2.40×10^{-7}	2.46×10^{-7}	200	200
Newton	$1.82 \times 10^{+2}$	$\mathbf{4.59 \times 10^{-24}}$	18	**8**
BFGS	2.52×10^{-15}	1.06×10^{-13}	31	10
NM	5.12×10^{-12}	1.43×10^{-8}	69	81

Benchmark 4: Ellipse (Ill-Conditioned)

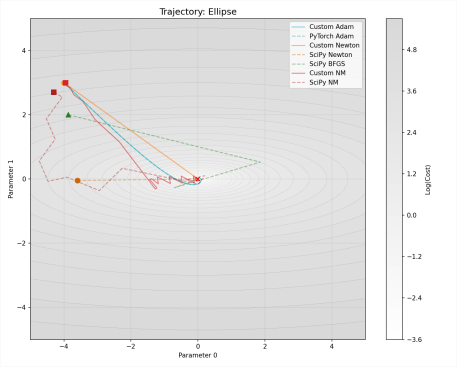


Figure 6: Newton handles scaling perfectly

Analysis

- **Newton:** Perfect compensation for the 10x scaling difference.
- **Adam:** Slower due to first-order nature.

Method	C-Cost	Ref-Cost	C-Iter	R-Iter
Adam	4.60×10^{-8}	6.12×10^{-8}	200	200
Newton	9.95×10^{-41}	$\mathbf{7.12 \times 10^{-47}}$	**3**	**3**
BFGS	1.65×10^{-16}	6.69×10^{-22}	6	6
NM	2.68×10^{-13}	2.75×10^{-9}	67	50

Benchmark 5: Booth (Plate Valley)

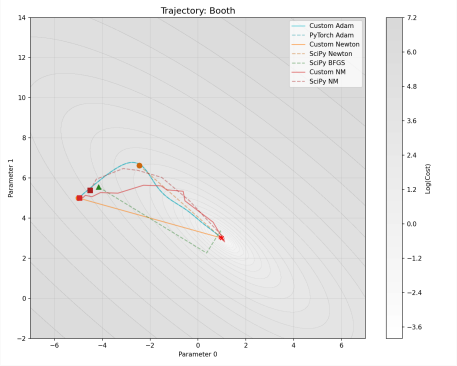


Figure 7: Navigating a flat plate

Analysis

- All optimizers succeeded.
- **Newton:** Again fastest (3 iterations).

Method	C-Cost	Ref-Cost	C-Iter	R-Iter
Adam	4.36×10^{-3}	4.73×10^{-3}	200	200
Newton	4.58×10^{-29}	$\mathbf{3.94 \times 10^{-30}}$	**3**	**3**
BFGS	4.04×10^{-28}	2.12×10^{-14}	7	6
NM	2.90×10^{-13}	1.23×10^{-9}	63	50

Benchmark 6: Matyas (Rotated Bowl)

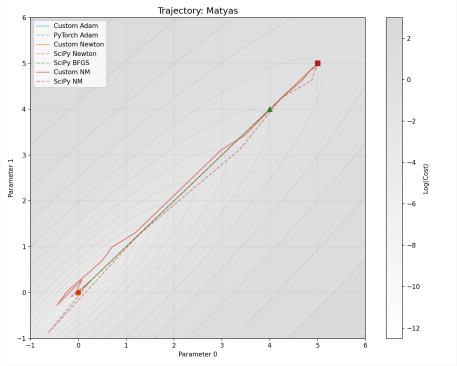


Figure 8: Handling parameter correlation

Analysis

- **Newton:** Handles the *xy* correlation term perfectly.
- **SciPy NM:** Fastest gradient-free method here.

Method	C-Cost	Ref-Cost	C-Iter	R-Iter
Adam	4.80×10^{-10}	1.85×10^{-7}	200	161
Newton	5.97×10^{-12}	$\mathbf{7.06 \times 10^{-37}}$	**2**	3
BFGS	3.11×10^{-27}	6.59×10^{-15}	4	3
NM	1.64×10^{-15}	1.05×10^{-10}	66	42

Benchmark 7: Three-Hump Camel (The Trap)

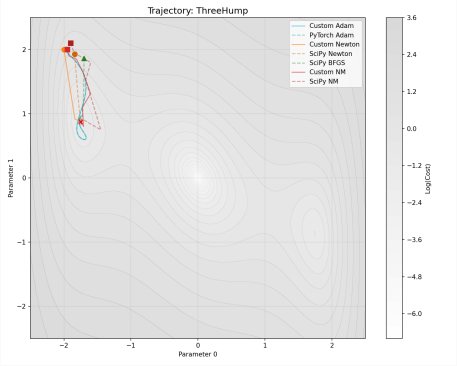


Figure 9: All paths trapped in local minimum

Analysis

- **Result:** Every optimizer failed (Cost 0.298).
- **Reason:** Initialization at $(-2, 2)$ is in the wrong basin.
- **Insight:** Local optimization is initialization-dependent.

Method	C-Cost	Ref-Cost	C-Iter	R-Iter
Adam	0.299×10^{-1}	0.299×10^{-1}	200	200
Newton	0.299×10^{-1}	0.299×10^{-1}	**6**	**6**
BFGS	0.299×10^{-1}	0.299×10^{-1}	10	7
NM	0.299×10^{-1}	0.299×10^{-1}	53	37

The Trade-Off Landscape

Method	Strengths	Weaknesses
Newton	Unbeatable speed on small, convex problems.	Fragile. Attracted to saddle points. Expensive ($O(N^3)$).
BFGS	Fast convergence without full Hessian.	Unstable on rugged landscapes without Line Search.
Adam	**Most Robust.** Handles stiffness and noise excellently.	Slower convergence (200+ iters).
Nelder-Mead	No gradients required. Good for noisy/non-differentiable	Slow convergence. Poor scaling to high dimen-

Conclusion & Future Work

Key Findings

1. **Numerical Differentiation** is a viable strategy for high-precision black-box optimization.
2. **Exact Newton** outperforms truncated approximations on small-scale problems but is fragile.
3. **Initialization Matters:** All local optimizers failed on the Three-Hump Camel trap.

Future Work: Quantum Application

- Replace test functions with **Variational Quantum Circuits**.
- **Conclusion:** Use **Adam** for VQE training due to its demonstrated resilience against non-convex features.