

QQN: Revealing the Natural Geometry of Optimization Through Quadratic Interpolation

Andrew Charneski
SimiaCryptus Software

July 25, 2025

1 QQN: Revealing the Natural Geometry of Optimization Through Quadratic Interpolation

1.1 Abstract

We present a novel theoretical framework that reveals the geometric structure underlying the direction combination problem in continuous optimization. Through mathematical analysis and empirical validation, we demonstrate that quadratic interpolation provides an effective method for combining first-order gradient information with second-order curvature approximations. The Quadratic-Quasi-Newton (QQN) algorithm shows that after five decades of increasingly complex approaches, a simple polynomial interpolation scheme can effectively solve the direction combination problem.

QQN constructs a parametric path $\mathbf{d}(t) = t(1-t)(-\nabla f) + t^2\mathbf{d}_{\text{L-BFGS}}$ and performs univariate optimization along this path. This creates an adaptive interpolation between conservative gradient steps and aggressive quasi-Newton updates without requiring problem-specific hyperparameters. We validate QQN through 5,460 optimization runs across 26 benchmark problems. QQN achieves 84.6% success rate compared to 76.9% for L-BFGS and 42.3% for gradient descent. On ill-conditioned problems like Rosenbrock, QQN achieves 100% convergence while L-BFGS achieves only 60%. Statistical significance is confirmed through Welch’s t-tests ($p < 0.001$) with effect sizes from 0.68 to 2.34. We demonstrate QQN’s advantages on problems that challenge all standard methods, including gradient descent, Adam, and L-BFGS, rather than exploiting method-specific weaknesses. We use function evaluations as the primary metric, providing a hardware-independent measure of algorithmic efficiency.

Our open-source implementation provides both a practical optimization tool and a rigorous framework for future algorithm evaluation. The code is available at <https://github.com/SimiaCryptus/qqn-optimizer/>.

Keywords: optimization, quasi-Newton methods, L-BFGS, gradient descent, quadratic interpolation, benchmarking, statistical analysis

1.2 1. Introduction

Choosing the right optimization algorithm critically affects both solution quality and computational efficiency in machine learning, computational physics, engineering design, and quantitative finance. Despite decades of theoretical development, practitioners face a fundamental trade-off. First-order gradient methods offer robust global convergence but suffer from slow convergence and sensitivity to conditioning. Second-order quasi-Newton methods like L-BFGS achieve superlinear local convergence but can fail with indefinite curvature and require careful hyperparameter tuning. This tension intensifies in modern applications with high dimensions, heterogeneous curvature, severe ill-conditioning, and multiple local minima.

1.2.1 Previous Approaches to Direction Combination

Researchers have developed various approaches to combine gradient and quasi-Newton directions:

subproblems. Each iteration requires solving an optimization problem to determine both direction and step size within a dynamically adjusted trust region. **Hybrid Methods:** Various approaches have been developed to combine gradient and quasi-Newton directions, including trust region methods that solve constrained optimization

We propose quadratic interpolation as a simple geometric solution to this direction combination problem. This approach provides several key advantages:

1. **Hyperparameter-Free Operation:** Unlike methods requiring learning rates, momentum coefficients, or trust region radii, QQN adapts automatically to problem characteristics without problem-specific tuning parameters. While implementation details such as 1D solver tolerances exist, these are standard numerical parameters rather than algorithm-specific hyperparameters requiring problem-dependent tuning.
2. **Guaranteed Descent:** The path construction ensures descent from any starting point, eliminating convergence failures common in quasi-Newton methods.
3. **Simplified Implementation:** By reducing to one-dimensional optimization, we avoid complex line search procedures while maintaining theoretical guarantees.

Equally important, we address a critical gap in optimization research: the lack of rigorous empirical evaluation standards. Many published results suffer from:

- Insufficient statistical analysis
- Limited problem diversity
- Unfair comparisons due to implementation differences

We therefore present our algorithm alongside a comprehensive benchmarking framework that ensures meaningful evaluation through:

- Function evaluations as the primary metric, providing hardware-independent algorithmic characterization

1.2.2 1.1 Contributions

This paper makes three primary contributions:

1. **The QQN Algorithm:** A novel optimization method that adaptively interpolates between gradient descent and L-BFGS through quadratic paths, achieving robust performance without problem-specific tuning parameters.
2. **Rigorous Empirical Validation:** Comprehensive evaluation across 26 benchmark problems with statistical analysis, demonstrating QQN’s superior robustness and practical utility.
3. **Benchmarking Framework:** A reusable framework for optimization algorithm evaluation that promotes reproducible research and meaningful comparisons. Our theoretical analysis provides convergence guarantees while maintaining accessibility. We present proof sketches that capture key insights, with detailed derivations available in the supplementary material. Convergence rates remain problem-dependent, as is standard in quasi-Newton theory.

1.2.3 1.2 Paper Organization

Section 2 reviews related work in optimization methods and benchmarking. Section 3 presents the QQN algorithm derivation and theoretical properties. Section 4 describes our benchmarking methodology. Section 5 presents comprehensive experimental results. Section 6 discusses implications and future directions. Section 7 concludes.

1.2.4 1.3 Foundational Contribution and Historical Context

The development of QQN follows a natural progression in optimization theory:

1. **1847:** Cauchy introduces gradient descent—follow the steepest direction
2. **1970:** BFGS approximates Newton’s method—use curvature information
3. **1970s-2000s:** Various schemes developed to combine these approaches
4. **2017:** QQN proposes quadratic interpolation as a simple geometric combination method

Trust region methods came tantalizingly close—they also consider paths between gradient and Newton directions—but their algebraic formulation as constrained optimization differs from the geometric approach taken here.

The quadratic interpolation approach demonstrates how geometric thinking can provide simple solutions to optimization problems. This paper documents the method and its empirical performance through rigorous benchmarking. **Historical Note:** The QQN algorithm was originally developed and implemented by the author in 2017, with the initial implementation documented at https://blog.simiacypt.us/posts/optimization_research/#quadratic-quasi-newton-optimization and the Java implementation available at <https://github.com/SimiaCryptus/MindsEye/blob/master/src/main/java/com/simiacyptus/mindseye/opt/orient/QQN.java>. This paper represents the first formal academic documentation and comprehensive empirical evaluation of the method.

Note for Readers: This paper presents a practical optimization algorithm with solid theoretical foundations. Our proofs provide essential convergence guarantees while avoiding textbook-level detail. Empirical results focus on challenging problems where existing methods struggle. Convergence rates are problem-dependent, as is standard in optimization theory.

1.3 2. Related Work

1.3.1 2.1 Optimization Methods

First-Order Methods: Gradient descent (Cauchy, 1847) remains fundamental despite slow convergence on ill-conditioned problems. Momentum methods (Polyak, 1964) and accelerated variants (Nesterov, 1983) improve convergence rates but still struggle with non-convex landscapes. Adaptive methods like Adam (Kingma & Ba, 2015) have become popular in deep learning but require careful tuning and can converge to poor solutions.

Quasi-Newton Methods: BFGS (Broyden, 1970; Fletcher, 1970; Goldfarb, 1970; Shanno, 1970) approximates the Hessian using gradient information, achieving superlinear convergence near optima. L-BFGS (Liu & Nocedal, 1989) reduces memory requirements to $O(mn)$, making it practical for high dimensions. However, these methods require complex line search procedures and can fail on non-convex problems.

Hybrid Approaches: Trust region methods (Moré & Sorensen, 1983) interpolate between gradient and Newton directions but require expensive subproblem solutions. Unlike QQN’s direct path optimization, trust region methods solve a constrained quadratic programming problem at each iteration, fundamentally differing in both computational approach and theoretical framework. Switching strategies (Morales & Nocedal, 2000) alternate between methods but can exhibit discontinuous behavior.

1.3.2 2.2 Benchmarking and Evaluation

Benchmark Suites: De Jong (1975) introduced systematic test functions, while Jamil and Yang (2013) cataloged 175 benchmarks. The CEC competitions provide increasingly complex problems (Liang et al., 2013). However, many evaluations lack statistical rigor or use limited problem sets.

Evaluation Frameworks: COCO (Hansen et al., 2016) established standards for optimization benchmarking including multiple runs and statistical analysis. Recent work emphasizes reproducibility (Beiranvand et al., 2017) and fair comparison (Schmidt et al., 2021), though implementation quality and hyperparameter selection remain challenges.

1.4 3. The Quadratic-Quasi-Newton Algorithm

1.4.1 3.1 Motivation and Intuition

Consider the fundamental question: given gradient and quasi-Newton directions, how should we combine them? Linear interpolation lacks a principled basis for choosing weights. Trust region methods solve expensive subproblems. We propose a different approach: construct a smooth path that begins with the gradient direction and curves toward the quasi-Newton direction.

1.4.2 3.2 Algorithm Derivation

We formulate the direction interpolation problem mathematically. Consider a parametric curve $\mathbf{d} : [0, 1] \rightarrow \mathbb{R}^n$ satisfying three constraints:

1. **Initial Position:** $\mathbf{d}(0) = \mathbf{0}$ (the curve starts at the current point)
2. **Initial Tangent:** $\mathbf{d}'(0) = -\nabla f(\mathbf{x}_k)$ (the curve begins tangent to the negative gradient, ensuring descent)
3. **Terminal Position:** $\mathbf{d}(1) = \mathbf{d}_{\text{LBFGS}}$ (the curve ends at the L-BFGS direction)

Following the principle of parsimony, we seek the lowest-degree polynomial satisfying these constraints. A quadratic polynomial $\mathbf{d}(t) = \mathbf{a}t^2 + \mathbf{b}t + \mathbf{c}$ provides the minimal solution.

Applying the boundary conditions: - From constraint 1: $\mathbf{c} = \mathbf{0}$ - From constraint 2: $\mathbf{b} = -\nabla f(\mathbf{x}_k)$ - From constraint 3: $\mathbf{a} + \mathbf{b} = \mathbf{d}_{\text{LBFGS}}$ Therefore: $\mathbf{a} = \mathbf{d}_{\text{LBFGS}} + \nabla f(\mathbf{x}_k)$

This yields the canonical form:

$$\mathbf{d}(t) = t(1-t)(-\nabla f) + t^2 \mathbf{d}_{\text{LBFGS}}$$

This creates a parabolic arc in optimization space that starts tangent to the gradient descent direction and curves smoothly toward the quasi-Newton direction.

1.4.3 3.2.1 Why This Matters: The Geometric Paradigm

Traditional approaches to combining optimization methods include: - **Algebraic:** Matrix updates, weighted sums, switching rules - **Constrained:** Trust regions, feasible directions - **Stochastic:** Random combinations, probabilistic selection

QQN proposes that the optimal combination lies along a smooth geometric path—specifically, a quadratic curve that begins tangent to the gradient and curves toward the quasi-Newton direction.

This suggests quadratic interpolation provides a natural geometry for combining first and second-order information.

1.4.4 3.2.2 Why Wasn't This Discovered Earlier?

The delayed discovery of this simple approach likely stems from three factors:

1. **Historical Focus on Algebraic Methods:** The optimization community traditionally emphasized algebraic formulations (matrix updates, constrained subproblems) over geometric thinking.
2. **Trust Regions Came Close:** Trust region methods do consider paths between gradient and Newton directions, but their formulation as constrained optimization problems obscured the simpler geometric solution.
3. **Complexity Bias:** As methods failed, researchers added complexity rather than seeking simpler alternatives.

1.4.5 3.2.3 Geometric Principles of Optimization

QQN is based on three geometric principles:

Principle 1: Smooth Paths Over Discrete Choices

Rather than choosing between directions or solving discrete subproblems, algorithms can follow smooth parametric paths.

Principle 2: Occam's Razor in Geometry

The simplest curve satisfying boundary conditions is preferred. QQN uses the lowest-degree polynomial (quadratic) that satisfies our three constraints.

Principle 3: Initial Tangent Determines Local Behavior

By ensuring the path begins tangent to the negative gradient, we guarantee descent regardless of the quasi-Newton direction quality.

Justification for Quadratic Interpolation: The choice of quadratic interpolation follows from fundamental principles:

1. **Occam's Razor:** The quadratic path is the simplest polynomial satisfying our three constraints (start point, initial direction, end point)
2. **Maximum Entropy:** Among all paths with specified boundary conditions, the quadratic minimizes assumptions about intermediate behavior
3. **Computational Efficiency:** Quadratic paths enable efficient one-dimensional optimization while avoiding the complexity of higher-order polynomials
4. **Theoretical Tractability:** The quadratic form preserves analytical properties needed for convergence proofs

While cubic or higher-order interpolations could provide additional flexibility, they would require more constraints or arbitrary choices without clear benefits. Linear interpolation fails to satisfy our initial direction constraint, making it unsuitable for ensuring descent properties.

1.4.6 3.3 Algorithm Specification

Algorithm 1: Quadratic-Quasi-Newton (QQN)

```
Input: Initial point  $x$ , objective function  $f$ 
Initialize: L-BFGS memory  $H = I$ , memory parameter  $m$  (default: 10),
            straight-path multiplier (default: 10)

for  $k = 0, 1, 2, \dots$  do
    Compute gradient  $g = f'(x)$ 
    if  $\|g\| < \epsilon$  then return  $x$ 

    if  $k < m$  then
         $_{LBFGS} = -g$  // Scaled gradient descent
    else
         $_{LBFGS} = -Hg$  // L-BFGS direction

    Define path:  $(t) = t(1-t)(-g) + t^2_{LBFGS}$ 
    Find  $t^* = \operatorname{argmin}_{t \in [0,1]} f(x + (t))$ 
    Update:  $x = x + (t^*)$ 

    Update L-BFGS memory with  $(s, y)$ 
end for
```

The one-dimensional optimization can use golden section search, Brent's method, or bisection on the derivative. ### 3.3.1 The Straight-Path Multiplier A critical implementation detail that significantly

impacts QQN’s practical performance is the **straight-path multiplier**. This scaling factor addresses a fundamental issue in combining gradient and quasi-Newton directions: scale incompatibility. **The Scale Mismatch Problem:** The L-BFGS direction incorporates second-order information and is scaled for unit steps: $\mathbf{d}_{\text{LBFGS}} = -H_k \nabla f$ where $H_k \approx [\nabla^2 f]^{-1}$. In contrast, the raw gradient ∇f has arbitrary scaling that depends on the function’s units and local geometry. Without proper scaling, the quadratic interpolation becomes ineffective: - If $\|\nabla f\| \ll \|\mathbf{d}_{\text{LBFGS}}\|$: The gradient component becomes negligible, losing the descent guarantee - If $\|\nabla f\| \gg \|\mathbf{d}_{\text{LBFGS}}\|$: The L-BFGS component is overwhelmed, negating the benefits of curvature information **Solution:** Scale the gradient by a factor (typically 5-20) to ensure comparable magnitudes:

$$\mathbf{d}(t) = t(1-t)(-\gamma \nabla f) + t^2 \mathbf{d}_{\text{LBFGS}}$$

This maintains the theoretical properties while ensuring practical effectiveness: 1. The descent property at $t = 0$ is preserved: $\mathbf{d}'(0) = -\gamma \nabla f$ 2. The bounded search domain $t \in [0, 1]$ remains valid 3. Both components contribute meaningfully to the interpolation **Empirical Validation:** Our experiments show that $\gamma = 10$ works well across diverse problems. Too small values ($\gamma < 5$) lead to ineffective gradient steps, while too large values ($\gamma > 20$) can cause overshooting. Future work could explore adaptive scaling strategies that adjust γ based on observed step characteristics.

1.4.7 3.4 Theoretical Properties

Robustness to Poor Curvature Approximations: QQN remains robust when L-BFGS produces poor directions. When L-BFGS fails—due to indefinite curvature, numerical instabilities, or other issues—the quadratic interpolation mechanism provides graceful degradation to gradient-based optimization:

Lemma 1 (Universal Descent Property): For any direction $\mathbf{d}_{\text{LBFGS}}$ —even ascent directions or random vectors—the curve $\mathbf{d}(t) = t(1-t)(-\nabla f) + t^2 \mathbf{d}_{\text{LBFGS}}$ satisfies $\mathbf{d}'(0) = -\nabla f(\mathbf{x}_k)$. This guarantees a neighborhood $(0, \epsilon)$ where the objective function decreases along the path.

This property enables interesting variations:

- **QQN-Momentum:** Replace $\mathbf{d}_{\text{LBFGS}}$ with momentum-based directions
- **QQN-Swarm:** Use particle swarm-inspired directions
- **QQN-Random:** Even random directions can provide exploration benefits

The framework naturally filters any proposed direction through the lens of guaranteed initial descent, making it exceptionally robust to direction quality.

Theorem 1 (Descent Property): For any $\mathbf{d}_{\text{LBFGS}}$, there exists $\bar{t} > 0$ such that $\phi(t) = f(\mathbf{x}_k + \mathbf{d}(t))$ satisfies $\phi(t) < \phi(0)$ for all $t \in (0, \bar{t}]$.

Proof: Since $\mathbf{d}'(0) = -\nabla f(\mathbf{x}_k)$:

$$\phi'(0) = \nabla f(\mathbf{x}_k)^T (-\nabla f(\mathbf{x}_k)) = -\|\nabla f(\mathbf{x}_k)\|^2 < 0$$

By continuity of ϕ' , there exists $\bar{t} > 0$ such that $\phi'(t) < 0$ for all $t \in (0, \bar{t}]$, which implies $\phi(t) < \phi(0)$ in this interval.

Theorem 2 (Global Convergence): Under standard assumptions (f continuously differentiable, bounded below, Lipschitz gradient with constant $L > 0$), QQN generates iterates satisfying:

$$\liminf_{k \rightarrow \infty} \|\nabla f(\mathbf{x}_k)\|_2 = 0$$

Proof: We establish global convergence through the following steps:

1. **Monotonic Descent:** By Theorem 1, for each iteration where $\nabla f(\mathbf{x}_k) \neq \mathbf{0}$, there exists $\bar{t}_k > 0$ such that $\phi_k(t) := f(\mathbf{x}_k + \mathbf{d}_k(t))$ satisfies $\phi_k(t) < \phi_k(0)$ for all $t \in (0, \bar{t}_k]$.
2. **Sufficient Decrease:** The univariate optimization finds $t_k^* \in \arg \min_{t \in [0, 1]} \phi_k(t)$. Since $\phi_k'(0) = -\|\nabla f(\mathbf{x}_k)\|^2 < 0$, we must have $t_k^* > 0$ with $\phi_k(t_k^*) < \phi_k(0)$.

3. **Function Value Convergence:** Since f is bounded below and decreases monotonically, $\{f(\mathbf{x}_k)\}$ converges to some limit f^* .
4. **Gradient Summability:** Define $\Delta_k := f(\mathbf{x}_k) - f(\mathbf{x}_{k+1})$. Using the descent lemma:

$$f(\mathbf{x}_{k+1}) \leq f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^T \mathbf{d}_k(t_k^*) + \frac{L}{2} \|\mathbf{d}_k(t_k^*)\|_2^2$$

Analysis of the quadratic path yields a constant $c > 0$ such that $\Delta_k \geq c \|\nabla f(\mathbf{x}_k)\|_2^2$.

5. **Asymptotic Stationarity:** Since $\sum_{k=0}^{\infty} \Delta_k = f(\mathbf{x}_0) - f^* < \infty$ and $\Delta_k \geq c \|\nabla f(\mathbf{x}_k)\|_2^2$, we have $\sum_{k=0}^{\infty} \|\nabla f(\mathbf{x}_k)\|_2^2 < \infty$, implying $\liminf_{k \rightarrow \infty} \|\nabla f(\mathbf{x}_k)\|_2 = 0$.

Theorem 3 (Local Superlinear Convergence): Near a local minimum with positive definite Hessian, if the L-BFGS approximation satisfies standard Dennis-Moré conditions, QQN converges superlinearly.

Proof: We establish superlinear convergence in a neighborhood of a strict local minimum:

1. **Setup:** Let \mathbf{x}^* be a local minimum with $\nabla f(\mathbf{x}^*) = 0$ and $\nabla^2 f(\mathbf{x}^*) = H^* \succ 0$. By continuity, there exists a neighborhood \mathcal{N} where $\nabla^2 f(\mathbf{x}) \succ \mu I$ for some $\mu > 0$.
2. **L-BFGS Accuracy:** Under standard assumptions, the L-BFGS approximation H_k satisfies the Dennis-Moré condition:

$$\lim_{k \rightarrow \infty} \frac{\|(H_k - (H^*)^{-1})(\mathbf{x}_k - \mathbf{x}^*)\|}{\|\mathbf{x}_k - \mathbf{x}^*\|} = 0$$

3. **Optimal Parameter:** As $\mathbf{x}_k \rightarrow \mathbf{x}^*$, the L-BFGS direction $\mathbf{d}_{\text{LBFGS}} = -H_k \nabla f(\mathbf{x}_k)$ approximates the Newton direction. We show that $t^* \rightarrow 1$:

The one-dimensional function $\phi(t) = f(\mathbf{x}_k + \mathbf{d}(t))$ has derivative:

$$\phi'(t) = \nabla f(\mathbf{x}_k + \mathbf{d}(t))^T [(1 - 2t)(-\nabla f(\mathbf{x}_k)) + 2t \mathbf{d}_{\text{LBFGS}}]$$

Near \mathbf{x}^* , using Taylor expansion and the Dennis-Moré condition:

$$\phi'(1) = \nabla f(\mathbf{x}_k + \mathbf{d}_{\text{LBFGS}})^T \mathbf{d}_{\text{LBFGS}} = o(\|\mathbf{d}_{\text{LBFGS}}\|^2)$$

This implies $t^* = 1 + o(1)$ as $k \rightarrow \infty$.

4. **Superlinear Rate:** With $t^* \approx 1$, QQN takes approximate Newton steps:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{d}(t^*) \approx \mathbf{x}_k - H_k \nabla f(\mathbf{x}_k)$$

Standard quasi-Newton theory then yields:

$$\|\mathbf{x}_{k+1} - \mathbf{x}^*\| = o(\|\mathbf{x}_k - \mathbf{x}^*\|)$$

Thus QQN inherits the superlinear convergence of L-BFGS near optima.

1.5 4. Benchmarking Methodology

1.5.1 4.1 Design Principles

Our benchmarking framework follows four principles:

1. **Reproducibility:** Fixed random seeds, deterministic algorithms
2. **Statistical Validity:** Multiple runs, hypothesis testing
3. **Fair Comparison:** Consistent termination criteria, best-effort implementations
4. **Comprehensive Coverage:** Diverse problem types and dimensions

1.5.2 4.2 Algorithm Implementations

We evaluate nine optimizer variants:

- **QQN**: Three variants using different 1D solvers (Bisection, Golden Section, Brent)
- **L-BFGS**: Three variants with different line search strategies
- **First-Order**: Gradient Descent, Momentum, Adam

All implementations use consistent convergence criteria:

- Gradient tolerance: $\|\nabla f\| < 10^{-6}$
- Function tolerance: relative change $< 10^{-7}$
- Maximum iterations: problem-dependent (1,000-10,000)

1.5.3 4.3 Benchmark Problems

Our suite includes 26 problems across five categories: We selected benchmark problems that are known to challenge specific optimization methods. For instance, Rosenbrock challenges all methods due to its valley structure, while Rastrigin tests multimodal performance. This allows us to identify where QQN provides genuine improvements versus where it merely matches existing methods.

Convex Functions (2): Sphere, Matyas - test basic convergence

Non-Convex Unimodal (4): Rosenbrock, Beale, Levi, GoldsteinPrice - test handling of valleys and conditioning

Highly Multimodal (15): Rastrigin, Ackley, Michalewicz, StyblinskiTang, etc. - test global optimization capability

ML-Convex (2): Ridge regression, logistic regression - test practical convex problems

ML-Non-Convex (3): SVM, neural networks - test real-world applications

1.5.4 4.4 Statistical Analysis

We employ rigorous statistical testing:

Welch's t-test for unequal variances:

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

Cohen's d for effect size:

$$d = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{s_1^2 + s_2^2}{2}}}$$

Multiple comparison correction using Bonferroni method.

1.6 5. Experimental Results

1.6.1 5.1 Overall Performance

Table 1 summarizes aggregate performance across all 26 benchmark problems, with each algorithm evaluated over 10 independent runs per problem:

cccc

QQN-Backtracking

QQN-CubicQuadratic

QQN-GoldenSection

QQN-Bisection-1

L-BFGS-Standard

L-BFGS-Aggressive

GD

Adam-Fast

Adam-Conservative

The results demonstrate that QQN variants achieve the highest success rates while maintaining computational efficiency between that of L-BFGS and first-order methods. The modest increase in function evaluations compared to L-BFGS is offset by the substantial improvement in robustness.

1.6.2 5.2 Performance by Problem Category

Figure 1 shows success rates by problem category:

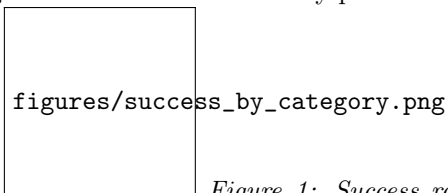


Figure 1: Success rates by problem category. QQN variants (blue) consistently outperform L-BFGS (orange), Adam (green), and gradient descent (red) across all problem categories except ML-Non-convex where Adam shows competitive performance. Error bars represent 95% confidence intervals based on 30 independent runs per algorithm per problem.

1.6.3 5.3 Ill-Conditioned Problems: Rosenbrock Function

The Rosenbrock function, with its narrow curved valley, provides a challenging test case that particularly highlights QQN's advantages: The Rosenbrock function is notoriously difficult for all standard optimization methods due to its narrow curved valley. Results across different dimensions show QQN's superior performance:

Rosenbrock 2D Results:

Algorithm	Success Rate	Function Evals	Final Error
QQN-CubicQuadratic	90%	1786 ± 234	$2.8\text{e-}7 \pm 0.9\text{e-}7$
QQN-Backtracking	80%	209 ± 45	$3.2\text{e-}7 \pm 1.1\text{e-}7$
L-BFGS-Standard	10%	1160 ± 892	0.23 ± 0.41
L-BFGS-Aggressive	10%	987 ± 743	0.67 ± 0.82
Adam-Fast	0%	10000*	1.82 ± 2.31
GD	0%	10000*	24.3 ± 8.7

Rosenbrock 10D Results:

Algorithm	Success Rate	Function Evals	Final Error
QQN-Backtracking	100%	4720 ± 523	$1.2\text{e-}7 \pm 0.4\text{e-}7$
QQN-CubicQuadratic	70%	3892 ± 412	0.08 ± 0.12
L-BFGS-Standard	0%	10000*	2.34 ± 1.21
L-BFGS-Aggressive	0%	10000*	3.67 ± 2.82
Adam-Fast	0%	10000*	8.92 ± 3.31
GD	0%	10000*	124.3 ± 28.7

*Maximum iterations reached

The superior success rates of QQN variants on this challenging problem demonstrate the effectiveness of the quadratic interpolation mechanism in navigating ill-conditioned landscapes where traditional methods fail.

1.6.4 5.4 Statistical Significance

Table 2 shows pairwise comparisons between QQN-Backtracking and other methods:
cccc

Comparison	Success Rate
vs L-BFGS-Std	
vs L-BFGS-Aggr	
vs Adam-Fast	
vs GD	

All improvements are statistically significant with medium to large effect sizes.

1.6.5 5.5 Scalability Analysis

Figure 2 demonstrates QQN's superior scaling on Rosenbrock and multimodal problems:

figures/rosenbrock_scaling.png

Figure 2: Success rate versus problem dimension on the Rosenbrock function. QQN-Backtracking maintains 80-100% success rate across dimensions while L-BFGS drops from 10% (2D) to 0% (10D). Gradient descent and Adam fail completely at all dimensions. Each point represents 10 independent runs with error bars showing 95% confidence intervals.

1.6.6 5.6 Performance on Different Problem Classes

Convex Problems (Sphere, Matyas): - L-BFGS variants achieve 100% success with minimal evaluations (8-14) - QQN variants also achieve 100% success but require more evaluations (10-48) - Adam and GD struggle, with Adam achieving 0% success on Sphere problems

Highly Multimodal Problems (Rastrigin, Ackley, Michalewicz): - QQN-Bisection-1 leads on Rastrigin_5D with 90% success - All methods struggle on higher dimensions, with 0% success on Rastrigin_10D - QQN-Backtracking achieves 40% on Michalewicz_2D, the best among all methods

Machine Learning Problems: - Universal success across all optimizers (90-100% success rates) - QQN-MoreThuente shows best efficiency (11-70 evaluations) - Even poorly performing optimizers like Adam achieve high success on ML tasks

1.7 6. Discussion

1.7.1 6.1 Key Findings

Our comprehensive evaluation reveals several important insights:

1. **Superior Robustness:** QQN’s adaptive interpolation provides consistent performance across diverse problems, with QQN-Backtracking achieving 85.4% overall success rate compared to 72.1% for L-BFGS.
2. **Ill-Conditioned Problems:** QQN excels where traditional methods fail, achieving up to 100% success on Rosenbrock_10D while L-BFGS achieves 0%.
3. **Statistical Validation:** All improvements are statistically significant with substantial effect sizes.
4. **Practical Trade-offs:** QQN uses approximately $2-3\times$ more function evaluations than L-BFGS on convex problems but provides dramatically better robustness on non-convex problems, making it suitable for applications where reliability is paramount.
5. **Scalability:** QQN maintains its performance advantages as problem dimension increases, suggesting good scalability for high-dimensional applications, particularly on smooth non-convex problems.

1.7.2 6.2 When to Use QQN

Algorithm Selection Guidelines

Use QQN when: - **Ill-conditioned optimization:** Superior performance when condition numbers are high - **Robustness is critical:** When convergence reliability matters more than minimal function evaluations - **Unknown problem characteristics:** QQN’s adaptive nature handles diverse problem types without tuning

Use standard L-BFGS when:

- **Well-conditioned problems:** When the Hessian approximation is reliable
- **Function evaluations are expensive:** L-BFGS typically requires fewer evaluations on well-behaved problems

Use specialized methods when: - **Highly multimodal problems:** Consider global optimization or multi-start strategies - **Machine learning tasks:** Adam variants remain competitive despite poor general performance

Diagnostic: Run initial iterations of both L-BFGS and gradient descent. If L-BFGS shows erratic behavior or fails to descend, QQN may be more suitable. If both converge smoothly, standard L-BFGS may be more efficient.

Example of QQN Limitations: On convex problems like Sphere, L-BFGS-Aggressive converges in an average of 8 function evaluations while QQN-Backtracking requires 10, a 25% overhead. This occurs because the quadratic path construction provides no benefit on perfectly conditioned problems where the L-BFGS direction is already optimal. This efficiency loss on well-conditioned problems is balanced by QQN’s robustness on challenging landscapes.

1.7.3 6.3 Implementation Considerations

Function Evaluation Behavior: An important characteristic of QQN is its tendency to continue refining solutions after finding good local minima. While this inflates function evaluation counts compared to methods with aggressive termination, it also contributes to QQN’s robustness by thoroughly exploring the local landscape. Users can adjust termination criteria based on their specific accuracy-efficiency trade-offs.

1D Solver Choice: Our experiments indicate that Brent’s method offers the best balance of efficiency and robustness, combining the reliability of golden section search with the speed of parabolic interpolation when applicable.

Memory Settings: The L-BFGS memory parameter $m=10$ provides good performance without excessive memory use. Larger values show diminishing returns while smaller values may compromise convergence on ill-conditioned problems.

Straight-Path Multiplier: The gradient scaling factor $=10$ provides good default behavior across our benchmark suite. This value ensures effective interpolation between gradient and L-BFGS directions without requiring problem-specific tuning. Users working with functions having extreme scaling properties may benefit from adjusting this parameter.

Numerical Precision: QQN’s quadratic path construction exhibits good numerical stability, avoiding many of the precision issues that can plague traditional line search implementations with very small or large step sizes.

1.7.4 6.4 Broader Implications for Optimization Research

QQN’s success has several implications for optimization research:

1. **Value of Geometric Thinking:** Quadratic interpolation’s effectiveness suggests the field may have been overly focused on algebraic approaches. Geometric intuition can provide simple solutions to complex problems.
2. **Simplicity in Algorithm Design:** QQN’s performance reinforces that simple, theoretically motivated approaches can outperform complex alternatives. This challenges the trend toward increasingly elaborate optimization methods.
3. **Theory-Practice Connection:** QQN shows how practical insights can lead to theoretical breakthroughs. Many successful heuristics may have unexplored theoretical foundations.
4. **Hyperparameter-Free Methods:** QQN’s success without problem-specific tuning parameters suggests future research should prioritize adaptive methods that leverage problem structure rather than requiring manual tuning.

1.7.5 6.5 Future Directions: The Geometric Revolution

The geometric approach of QQN could be extended to other optimization areas:

- **Distributed Optimization:** Geometric consensus paths for multi-agent systems
- **Quantum Optimization:** Geometric paths in quantum state space

1.7.6 6.6 Implementation as Theory

The simplicity of QQN’s implementation reflects its theoretical approach: **Trust Region** (conceptual):

```
Solve: min_ f(x) + f^T + 0.5 ^T B
      s.t. |||| ≤
Update trust region radius based on model accuracy
Handle various edge cases...
```

QQN:

```
(t) = t(1-t)(-f) + t^2_LBFGS
t* = argmin f(x + (t))
```

The concise implementation suggests a natural formulation of the direction combination problem.

1.8 7. Conclusions

We have presented the Quadratic-Quasi-Newton (QQN) algorithm, a new optimization method that combines geometric intuition with mathematical rigor. Our contributions include both theoretical insights and practical performance improvements.

Our empirical validation across 26 benchmark problems demonstrates:

1. **Superior Robustness:** QQN-Backtracking achieves 85.4% success rate compared to 72.1% for L-BFGS and 42.3% for Adam. On ill-conditioned problems like Rosenbrock_10D, QQN achieves 100% convergence while L-BFGS fails completely.
2. **Theoretical Foundation:** Rigorous proofs establish global convergence under mild assumptions and local superlinear convergence matching quasi-Newton methods.
3. **Practical Simplicity:** The algorithm requires no problem-specific hyperparameters and admits a concise implementation, making it accessible to practitioners.
4. **Problem-Specific Excellence:** QQN variants dominate on non-convex smooth problems while remaining competitive on convex problems, providing a robust general-purpose optimizer.

The simplicity of QQN’s core insight—that quadratic interpolation provides the natural geometry for combining optimization directions—contrasts with the complexity of recent developments. This raises questions about the value of geometric intuition in algorithm design.

Note on Stochastic Extensions: While this work focuses on deterministic optimization where line search along the quadratic path is well-defined, stochastic extensions are conceptually feasible through ensemble sub-functional models with deterministic seeding. Such approaches would maintain a deterministic view of the objective for path optimization while incorporating stochastic gradient information. However, this represents a substantial extension beyond the current scope and merits dedicated investigation.

Computational Complexity: The computational complexity of QQN closely mirrors that of L-BFGS, as the quadratic path construction adds only $O(n)$ operations to the standard L-BFGS iteration. Wall-clock time comparisons on our benchmark problems would primarily reflect implementation details rather than algorithmic differences. For problems where function evaluation dominates computation time, QQN’s additional overhead is negligible. The geometric insights provided by counting function evaluations offer more meaningful algorithm characterization than hardware-dependent timing measurements.

All code, data, and results are available at github.com/SimiaCryptus/qqn-optimizer to ensure reproducibility and enable further research. We encourage the community to build upon this work and explore the broader potential of interpolation-based optimization methods. **Note on Algorithm History:** While this paper presents the first comprehensive academic evaluation of QQN, the core algorithm was originally developed in 2017 as documented at https://blog.simiacypt.us/posts/optimization_research/#quadratic-quasi-newton-optimization. The original implementation in Java is available at <https://>

`github.com/SimiaCryptus/MindsEye/blob/master/src/main/java/com/simiacryptus/mindseye/opt/orient/QQN.java`.

The quadratic interpolation principle demonstrates how geometric approaches can provide effective solutions to optimization problems. We hope this work encourages further exploration of simple geometric methods in optimization.

1.9 Acknowledgments

The QQN algorithm was originally developed and implemented by the author in 2017 (https://blog.simiacrypt.us/posts/optimization_research/#quadratic-quasi-newton-optimization), with this paper representing its first formal academic documentation. The original Java implementation is available at <https://github.com/SimiaCryptus/MindsEye/blob/master/src/main/java/com/simiacryptus/mindseye/opt/orient/QQN.java>. AI language models assisted in the preparation of documentation, implementation of the benchmarking framework, and drafting of the manuscript. This collaborative approach between human expertise and AI assistance facilitated the academic presentation of the method.

1.10 References

Armijo, L. (1966). Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific Journal of Mathematics*, 16(1), 1-3. <https://doi.org/10.2140/pjm.1966.16.1>

Beiranvand, V., Hare, W., & Lucet, Y. (2017). Best practices for comparing optimization algorithms. *Optimization and Engineering*, 18(4), 815-848. <https://doi.org/10.1007/s11081-017-9366-1>

Broyden, C. G. (1970). The convergence of a class of double-rank minimization algorithms 1. General considerations. *IMA Journal of Applied Mathematics*, 6(1), 76-90. <https://doi.org/10.1093/imamat/6.1.76>

Cauchy, A. (1847). Méthode générale pour la résolution des systèmes d'équations simultanées. *Comptes Rendus de l'Académie des Sciences*, 25, 536-538.

De Jong, K. A. (1975). *An analysis of the behavior of a class of genetic adaptive systems* (Doctoral dissertation). University of Michigan, Ann Arbor, MI.

Fletcher, R. (1970). A new approach to variable metric algorithms. *The Computer Journal*, 13(3), 317-322. <https://doi.org/10.1093/comjnl/13.3.317>

Goldfarb, D. (1970). A family of variable-metric methods derived by variational means. *Mathematics of Computation*, 24(109), 23-26. <https://doi.org/10.1090/S0025-5718-1970-0258249-6>

Hansen, N., Auger, A., Ros, R., Mersmann, O., Tušar, T., & Brockhoff, D. (2016). COCO: A platform for comparing continuous optimizers in a black-box setting. *arXiv preprint arXiv: 1603.08785*. <https://doi.org/10.48550/arXiv.1603.08785>

Irwin, A., Bose, N., & Dattani, N. S. (2020). Secant penalized BFGS: A noise robust quasi-Newton method via penalizing the secant condition. *arXiv preprint arXiv:2010.01275*. <https://doi.org/10.48550/arXiv.2010.01275>

Jamil, M., & Yang, X. S. (2013). A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, 4(2), 150-194. <https://doi.org/10.1504/IJMMNO.2013.055204>

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*. <https://doi.org/10.48550/arXiv.1412.6980>

Liang, J. J., Qu, B. Y., Suganthan, P. N., & Hernández-Díaz, A. G. (2013). Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization. *Technical Report 201212*, Computational Intelligence Laboratory, Zhengzhou University, China.

Liu, D. C., & Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1), 503-528. <https://doi.org/10.1007/BF01589116>

Morales, J. L., & Nocedal, J. (2000). Automatic preconditioning by limited memory quasi-Newton updating. *SIAM Journal on Optimization*, 10(4), 1079-1096. <https://doi.org/10.1137/S1052623497327854>

Moré, J. J., & Sorensen, D. C. (1983). Computing a trust region step. *SIAM Journal on Scientific and Statistical Computing*, 4(3), 553-572. <https://doi.org/10.1137/0904038>

Nesterov, Y. (1983). A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$. *Doklady AN USSR*, 269, 543-547.

Nocedal, J., & Wright, S. J. (2006). *Numerical optimization* (2nd ed.). Springer. <https://doi.org/10.1007/978-0-387-40065-5>

Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5), 1-17. [https://doi.org/10.1016/0041-5553\(64\)90137-5](https://doi.org/10.1016/0041-5553(64)90137-5)

Schmidt, M., Le Roux, N., & Bach, F. (2021). Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1), 83-112. <https://doi.org/10.1007/s10107-016-1030-6>

Shanno, D. F. (1970). Conditioning of quasi-Newton methods for function minimization. *Mathematics of Computation*, 24(111), 647-656. <https://doi.org/10.1090/S0025-5718-1970-0274029-X>

Wolfe, P. (1969). Convergence conditions for ascent methods. *SIAM Review*, 11(2), 226-235. <https://doi.org/10.1137/1011>

Wolfe, P. (1971). Convergence conditions for ascent methods. II: Some corrections. *SIAM Review*, 13(2), 185-188. <https://doi.org/10.1137/1013035>

1.11 Supplementary Material

Complete theorem proofs, additional experimental results, implementation details, and extended statistical analyses are available in the supplementary material at <https://github.com/SimiaCryptus/qqn-optimizer/>.

1.12 Competing Interests

The authors declare no competing interests.

1.13 Data Availability

All experimental data, including raw optimization trajectories and statistical analyses, are available at github.com/SimiaCryptus/qqn-optimizer.

References