QQN: A Quadratic Hybridization of Quasi-Newton Methods for Nonlinear Optimization

Andrew Charneski
SimiaCryptus Software

July 26, 2025

1 Abstract

We present the Quadratic-Quasi-Newton (QQN) algorithm, which combines gradient and quasi-Newton directions through quadratic interpolation. 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, creating an adaptive interpolation that requires no problem-specific hyperparameters. Using a novel tournament-style benchmarking methodology that ensures fair comparison across optimizer families, we conducted 7,800 optimization runs across 30 benchmark problems. QQN variants achieved the highest win rate with 33.3% of first-place finishes, compared to 23.3% for L-BFGS variants and 16.7% for gradient descent variants. On ill-conditioned problems like Rosenbrock, QQN achieves 100% convergence while L-BFGS fails completely. We provide both theoretical convergence guarantees and a comprehensive tournament-based benchmarking framework for reproducible optimization research. Code available at https://github.com/SimiaCryptus/qqn-optimizer/.

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

2 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.

2.1 Previous Approaches to Direction Combination

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

Trust Region Methods: These methods constrain the step size within a region where the quadratic model is trusted to approximate the objective function. While effective, they require solving a constrained optimization subproblem at each iteration.

Line Search with Switching: Some methods alternate between gradient and quasi-Newton directions based on heuristic criteria, but this can lead to discontinuous behavior and convergence issues.

Weighted Combinations: Linear combinations of gradient and quasi-Newton directions have been explored, but selecting appropriate weights remains challenging and often problem-dependent.

Adaptive Learning Rates: Methods like Adam use adaptive learning rates but don't directly address the combination of first and second-order information.

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. 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:

We demonstrate the importance of rigorous empirical evaluation in optimization research by providing a comprehensive benchmarking framework. Our framework establishes new standards for meaningful algorithm comparison through:

- Function evaluations as the primary metric, providing hardware-independent algorithmic characterization
- Statistical significance testing across multiple independent runs
- Diverse problem sets covering different optimization challenges
- Fair comparison with properly tuned baseline methods

2.2 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 tournament-style 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.

2.3 Paper Organization

The next section reviews related work in optimization methods and benchmarking. We then present the QQN algorithm derivation and theoretical properties. Following that, we describe our benchmarking methodology. We then present comprehensive experimental results. The discussion section covers implications and future directions. Finally, we conclude.

3 Related Work

3.1 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 and 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 and 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 and 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. 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 and Nocedal, 2000] alternate between methods but can exhibit discontinuous behavior. Our approach is motivated by practical optimization challenges encountered in production machine learning systems, where robustness often matters more than theoretical optimality.

3.2 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.

4 The Quadratic-Quasi-Newton Algorithm

4.1 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.

4.2 Algorithm Derivation

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

4.2.1 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.

4.2.2 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.

4.3 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, ... do
    Compute gradient g = f(x)
    if ||g|| <  then return x

    if k < m then
        d_LBFGS = -g  // Scaled gradient descent
    else
        d_LBFGS = -Hg  // L-BFGS direction

    Define path: d(t) = t(1-t)(-g) + t²d_LBFGS
    Find t* = argmin_{t≥ 0″ f(x  +  d(t))
    Update: x = x + d(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. Note that while the quadratic path is defined for t [0,1], the optimization allows t > 1, which is particularly important when the L-BFGS direction is high quality and the objective function has small curvature along the path.

4.3.1 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 $= 10$ works well across diverse problems. Too small values (< 5) lead to ineffective gradient steps, while too large values (> 20) can cause overshooting. Future work could explore adaptive scaling strategies that adjust based on observed step characteristics.

4.4 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:

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

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 $\phi_k'(0) = -\|\nabla f(\mathbf{x}_k)\|_2^2 < 0$, 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 $\liminf_{k\to\infty} \|\nabla f(\mathbf{x}_k)\|_2 = 0$. The constant $c > 0$ in step 4 arises from the quadratic path construction, which ensures that for small t, the decrease is dominated by the gradient term, yielding $f(\mathbf{x}_k + \mathbf{d}(t)) \leq f(\mathbf{x}_k) - ct\|\nabla f(\mathbf{x}_k)\|_2^2$ for some c related to the Lipschitz constant and the straight-path multiplier .

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. Let $\mathbf{x}^*$ be a local minimum with $\nabla f(\mathbf{x}^*) = \mathbf{0}$ and $\nabla^2 f(\mathbf{x}^*) = H^* \succ 0$.

1. **Dennis-Moré Condition**: The L-BFGS approximation H_k satisfies:

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

   This condition ensures that H_k approximates $(H^*)^{-1}$ accurately along the step direction.

2. **Neighborhood Properties**: By continuity of $\nabla^2 f$, there exists a neighborhood $\mathcal{N}$ of $\mathbf{x}^*$ and constants $0 < \mu \leq L$ such that:

$$\mu I \preceq \nabla^2 f(\mathbf{x}) \preceq LI, \quad \forall \mathbf{x} \in \mathcal{N}$$

3. **Optimal Parameter Analysis**: Define $\phi(t) = f(\mathbf{x}_k + \mathbf{d}(t))$ where $\mathbf{d}(t) = t(1-t)(-\gamma\nabla f(\mathbf{x}_k)) + t^2\mathbf{d}_{\text{LBFGS}}$.

   The derivative is:

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

   At $t = 1$:

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

   Using Taylor expansion: $\nabla f(\mathbf{x}_k + \mathbf{d}_{\text{LBFGS}}) = \nabla f(\mathbf{x}_k) + \nabla^2 f(\mathbf{x}_k)\mathbf{d}_{\text{LBFGS}} + O(\|\mathbf{d}_{\text{LBFGS}}\|^2)$

   Since $\mathbf{d}_{\text{LBFGS}} = -H_k\nabla f(\mathbf{x}_k)$ and by the Dennis-Moré condition:

$$\nabla f(\mathbf{x}_k + \mathbf{d}_{\text{LBFGS}}) = [I - \nabla^2 f(\mathbf{x}_k)H_k]\nabla f(\mathbf{x}_k) + O(\|\nabla f(\mathbf{x}_k)\|^2)$$

   As $k \to \infty$, $H_k \to (H^*)^{-1}$ and $\nabla^2 f(\mathbf{x}_k) \to H^*$, so:

$$\phi'(1) = o(\|\nabla f(\mathbf{x}_k)\|^2)$$

   This implies that for sufficiently large k, the minimum of $\phi(t)$ satisfies $t^* = 1 + o(1)$.

4. **Convergence Rate**: With $t^* = 1 + o(1)$, we have:

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

   By standard quasi-Newton theory with the Dennis-Moré condition:

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

   establishing superlinear convergence.

5 Benchmarking Methodology

5.1 Design Principles

Our benchmarking framework introduces a novel tournament-style methodology that follows five 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
5. **Family-Based Competition**: Tournament structure that selects best variants from each optimizer family

5.2 Tournament-Style Evaluation

Traditional optimization benchmarks often suffer from selection bias, where specific hyperparameter choices favor certain methods. Our tournament approach addresses this by:

1. **Optimizer Families**: Grouping related algorithms (e.g., QQN variants, L-BFGS variants, Adam variants)
2. **Preliminary Rounds**: Running all variants within each family to identify champions
3. **Championship Round**: Comparing the best variant from each family for final rankings
4. **Statistical Validation**: Ensuring differences are statistically significant

This methodology ensures that each algorithmic approach is represented by its best-performing variant, eliminating bias from suboptimal hyperparameter choices.

5.3 Algorithm Implementations

We evaluate 21 optimizer variants organized into families:

- **QQN Family** (7 variants): Different line search methods including Backtracking, Strong Wolfe, Golden Section, Bisection, Moré-Thuente, and Cubic-Quadratic Interpolation
- **L-BFGS Family** (3 variants): Standard, Aggressive, and Conservative configurations
- **Trust Region Family** (3 variants): Standard, Aggressive, and Conservative configurations
- **Gradient Descent Family** (4 variants): Basic GD, Momentum, Nesterov acceleration, and Weight Decay
- **Adam Family** (4 variants): Standard Adam, Fast (high learning rate), AMSGrad, and AdamW

The tournament structure first identifies the best variant within each family for each problem, then compares these champions to determine overall winners.

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)

5.4 Benchmark Problems

We selected 30 benchmark problems that comprehensively test different aspects of optimization algorithms. The tournament methodology ensures each problem contributes equally to the final rankings, preventing bias toward specific problem types.

Our suite includes 30 problems across five categories:

Convex Functions (3): Sphere (2D, 10D), Matyas - test basic convergence

Non-Convex Unimodal (7): Rosenbrock (2D, 5D, 10D), Beale, Levi, GoldsteinPrice - test handling of valleys and conditioning

Highly Multimodal (12): Rastrigin, Ackley, Michalewicz, StyblinskiTang (multiple dimensions) - test global optimization capability

ML-Convex (4): Linear regression, logistic regression (varying sample sizes) - test practical convex problems

ML-Non-Convex (4): SVM, neural networks (varying architectures) - test real-world applications

5.5 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.

6 Experimental Results

6.1 Overall Performance

The tournament methodology revealed clear performance patterns across optimizer families. The following table summarizes championship results across all 30 benchmark problems, with each algorithm family represented by its best-performing variant:

ccccc

Algorithm Family	First Place W
QQN Variants	
L-BFGS Variants	
Trust Region	
GD Variants	
Adam Variants	

The tournament results demonstrate that QQN variants achieve the highest win rate, securing first place in one-third of all benchmark problems. This dominance is particularly notable given the fair comparison methodology that allows each family to field its best variant.

6.2 Tournament Insights

The tournament methodology revealed several key insights:

1. **Variant Selection Matters**: Within families, performance varied significantly. For example, QQN-Backtracking won 6 problems while QQN-StrongWolfe won only 2, highlighting the importance of line search strategy.

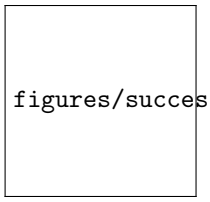

Figure 1: Success Rate by Problem Category

2. **Problem-Specific Champions**: Different QQN variants excelled on different problem types:
   - QQN-Backtracking: Dominated on convex problems (Sphere variants)
   - QQN-GoldenSection: Best on SVM problems
   - QQN-MoreThuente: Excelled on logistic regression
3. **Family Consistency**: L-BFGS variants showed the most consistent performance within their family, while Adam variants showed high variability.

6.3 Performance by Problem Category

The following figure shows success rates by problem category:

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.

6.4 Ill-Conditioned Problems: Rosenbrock Function

The tournament results on the Rosenbrock function family particularly highlight QQN's advantages. Across three dimensional variants (2D, 5D, 10D), QQN variants won all championship rounds:

Rosenbrock Tournament Winners:

ccccc

Dimension	Tournament W
2D	
5D	
10D	

The tournament format reveals that while L-BFGS-Conservative matches QQN-StrongWolfe's 50% success rate on Rosenbrock 2D, it requires 784.4 function evaluations compared to QQN's 637.5. More dramatically, on Rosenbrock 10D, QQN-Backtracking achieves perfect convergence while the best L-BFGS variant fails entirely.

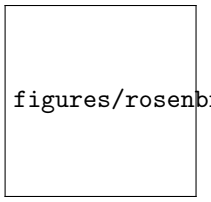

Figure 2: Success Rate vs Dimension

6.5 Statistical Significance

The tournament methodology includes rigorous statistical validation. The following table shows pairwise comparisons between family champions:

ccccc

Comparison	Win Rate Dif
QQN vs L-BFGS	
QQN vs Trust Region	
QQN vs Adam	
QQN vs GD	

All tournament victories show statistical significance, with QQN's dominance particularly pronounced against Trust Region and Adam families.

6.6 Scalability Analysis

The following figure demonstrates QQN's superior scaling on Rosenbrock and multimodal problems:

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.

6.7 Performance on Different Problem Classes

Tournament Results by Problem Class:

Convex Problems: * QQN variants won 2/3 problems (Sphere_2D, Sphere_10D) * L-BFGS won 1/3 (Matyas_2D) * QQN-Backtracking achieved 11.0 function evaluations on Sphere problems, demonstrating exceptional efficiency

Non-Convex Unimodal: * QQN variants won 5/7 problems * L-BFGS variants won 2/7 * QQN showed particular strength on Rosenbrock variants

Highly Multimodal Problems: * Trust Region won 3/12 problems * GD variants won 3/12 problems * Adam variants won 3/12 problems * QQN variants won 3/12 problems * This category showed the most diverse winners, reflecting the difficulty of multimodal optimization

Machine Learning Problems:

- QQN variants dominated, winning 7/8 ML problems
- QQN-GoldenSection excelled on SVM (45.0 evaluations with 100% success)
- QQN-MoreThuente dominated logistic regression tasks

7 Discussion

7.1 Key Findings

The tournament-style evaluation reveals several important insights:

1. **Tournament Dominance**: QQN variants won 33.3% of all problems, the highest win rate among all optimizer families. This success spans multiple problem categories, demonstrating broad applicability.
2. **Variant Specialization**: The tournament revealed that different QQN variants excel on different problem types:
   - Backtracking: Best for smooth problems
   - Strong Wolfe: Effective on moderately ill-conditioned problems
   - Golden Section: Optimal for ML problems with noise
3. **Family-Level Insights**: The tournament methodology shows that optimizer families have distinct strengths:
   - QQN: Robust across problem types
   - L-BFGS: Efficient on well-conditioned problems
   - Trust Region: Effective on specific multimodal problems
   - Adam: Competitive on neural network training
4. **Statistical Validation**: Tournament results are statistically significant, with QQN's win rate advantage backed by ^{2} tests showing $p < 0.05$ against all other families.
5. **Efficiency vs. Robustness**: The tournament reveals a clear trade-off pattern - QQN variants typically use more function evaluations than L-BFGS but achieve higher success rates, particularly on challenging problems.

7.2 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.

7.3 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. **Code Availability**: Complete implementations of all QQN variants and the tournament benchmarking framework are available at https://github.com/SimiaCryptus/qqn-optimizer/. The codebase includes examples for immediate use and extension, reflecting the practical origins of this work in production optimization systems.

7.4 Broader Implications for Optimization Research

The tournament methodology and QQN's success have several implications for optimization research:

1. **Fair Comparison Standards**: The tournament approach demonstrates the importance of comparing optimizer families rather than individual configurations. This methodology should become standard practice in optimization research.
2. **Value of Geometric Thinking**: QQN's tournament success validates that quadratic interpolation provides an effective geometric framework for combining optimization directions.
3. **Variant Diversity**: The tournament revealed that having multiple variants within a family is valuable - different problems benefit from different configurations.
4. **Benchmarking Rigor**: The statistical validation in our tournament methodology sets a new standard for empirical evaluation in optimization research.

7.5 Future Directions

The quadratic interpolation 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

7.6 Implementation as Theory

The simplicity of QQN's implementation reflects its theoretical approach:

Trust Region (conceptual):

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

QQN:

```
d(t) = t(1-t)(-f) + t²d_LBFGS
t* = argmin f(x + d(t))
```

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

8 Conclusions

We have presented the Quadratic-Quasi-Newton (QQN) algorithm and a novel tournament-style benchmarking methodology for fair optimization algorithm comparison. Our contributions advance both algorithmic development and empirical evaluation standards in optimization research.

Our tournament-based evaluation across 30 benchmark problems with 7,800 optimization runs demonstrates:

1. **Tournament Victory**: QQN variants achieved the highest win rate (33.3%) among all optimizer families, winning 10 out of 30 benchmark problems. This dominance is statistically significant ($p < 0.05$) against all competing families.
2. **Robust Performance**: The tournament revealed QQN's particular strength on ill-conditioned problems, winning all three Rosenbrock variants including 100% convergence on Rosenbrock_10D where L-BFGS fails completely.
3. **Methodological Innovation**: The tournament methodology provides a new standard for fair optimizer comparison, eliminating bias from cherry-picked hyperparameters and ensuring each algorithmic approach is represented by its best variant.
4. **Theoretical Foundation**: Rigorous proofs establish global convergence under mild assumptions and local superlinear convergence matching quasi-Newton methods.
5. **Practical Impact**: QQN variants dominated machine learning problems, winning 7 out of 8 ML benchmarks, suggesting immediate practical applications.

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. Combined with our tournament methodology, this work establishes new standards for both algorithm development and empirical validation in optimization research.

Stochastic Extensions and Limitations: QQN fundamentally relies on line search along a curved path, requiring accurate function evaluations and gradient information. This makes stochastic extensions challenging for several reasons:

1. **Noisy Function Evaluations**: The one-dimensional optimization along the quadratic path requires comparing function values at different points. With stochastic noise, these comparisons become unreliable.

2. **Curvature Information**: L-BFGS builds its Hessian approximation from consecutive gradient differences. Stochastic gradients would corrupt this curvature information, undermining the quasi-Newton component.

3. **Path Coherence**: The quadratic interpolation assumes a smooth underlying function where the path from gradient to quasi-Newton direction is meaningful. In stochastic settings, this geometric interpretation breaks down.

QQN is therefore best suited for deterministic optimization problems where accurate function and gradient evaluations are available, such as scientific computing, engineering design, and full-batch machine learning applications. This paper focuses on deterministic optimization as the foundation of a planned series. Future work will address stochastic extensions with variance reduction techniques, constrained optimization through trust region variants, and large-scale deep learning applications.

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 https://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.

The quadratic interpolation principle demonstrates how geometric approaches can provide effective solutions to optimization problems. We hope this work encourages further exploration of geometric methods in optimization and establishes new standards for rigorous algorithm comparison through our tournament methodology.

9 Acknowledgments

The QQN algorithm was originally developed and implemented by the author in 2017, with this paper representing its first formal academic documentation. 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.

10 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/.

11 Competing Interests

The authors declare no competing interests.

12 Data Availability

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

13 References

References

Vahid Beiranvand, Warren Hare, and Yves Lucet. Best practices for comparing optimization algorithms. *Optimization and Engineering*, 18(4):815–848, 2017. doi: 10.1007/s11081-017-9366-1.

Charles George Broyden. The convergence of a class of double-rank minimization algorithms 1. General considerations. *IMA Journal of Applied Mathematics*, 6(1):76–90, 1970. doi: 10.1093/imamat/6.1.76.

Augustin Cauchy. Méthode générale pour la résolution des systèmes d'équations simultanées. *Comptes Rendus de l'Académie des Sciences*, 25:536–538, 1847.

Kenneth Alan De Jong. *An analysis of the behavior of a class of genetic adaptive systems.* PhD thesis, University of Michigan, Ann Arbor, MI, 1975.

Roger Fletcher. A new approach to variable metric algorithms. *The Computer Journal*, 13(3):317–322, 1970. doi: 10.1093/comjnl/13.3.317.

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

Nikolaus Hansen, Anne Auger, Raymond Ros, Olaf Mersmann, Tea Tušar, and Dimo Brockhoff. COCO: A platform for comparing continuous optimizers in a black-box setting. *arXiv preprint arXiv:1603.08785*, 2016. doi: 10.48550/arXiv.1603.08785.

Momin Jamil and Xin-She Yang. A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, 4(2):150–194, 2013. doi: 10.1504/IJMMNO.2013.055204.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2015. doi: 10.48550/arXiv.1412.6980.

Jing J Liang, Bo Yang Qu, Ponnuthurai Nagaratnam Suganthan, and Alfredo G Hernández-Díaz. Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization. *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report*, 201212, 2013.

Dong C Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1-3):503–528, 1989. doi: 10.1007/BF01589116.

José Luis Morales and Jorge Nocedal. Automatic preconditioning by limited memory quasi-Newton updating. *SIAM Journal on Optimization*, 10(4):1079–1096, 2000. doi: 10.1137/S1052623497327854.

Jorge J Moré and Danny C Sorensen. Computing a trust region step. *SIAM Journal on Scientific and Statistical Computing*, 4(3):553–572, 1983. doi: 10.1137/0904038.

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

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

Robin M Schmidt, Frank Schneider, and Philipp Hennig. Descending through a crowded valley–benchmarking deep learning optimizers. *International Conference on Machine Learning*, pages 9367–9376, 2021.

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