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

Andrew Charneski
SimiaCryptus Software

August 3, 2025

1 Abstract

We present the Quadratic-Quasi-Newton (QQN) algorithm, a novel optimization method that combines gradient descent 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 additional hyperparameters beyond those of its constituent methods.

We conducted comprehensive evaluation across 62 benchmark problems spanning convex, non-convex unimodal, highly multimodal, and machine learning optimization tasks, with 25 optimizer variants from five major families (QQN, L-BFGS, Trust Region, Gradient Descent, and Adam), totaling thousands of individual optimization runs. Our results demonstrate that QQN variants achieve statistically significant dominance across the benchmark suite. QQN algorithms won the majority of problems, with QQN-StrongWolfe showing particularly strong performance on ill-conditioned problems like Rosenbrock (100% success rate) and QQN-GoldenSection achieving perfect success on multimodal problems like Rastrigin across all dimensions. Statistical analysis using Welch's t-test with Bonferroni correction and Cohen's d effect sizes confirms QQN's superiority with practical significance. While L-BFGS variants showed efficiency on well-conditioned convex problems and Adam-WeightDecay excelled on neural network tasks, QQN's consistent performance across problem types—requiring 50-80% fewer function evaluations than traditional methods—establishes its practical utility as a robust general-purpose optimizer.

We provide theoretical convergence guarantees (global convergence under standard assumptions and local superlinear convergence) and introduce a comprehensive 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

1.1 Paper Series Overview

This paper is the first in a planned series on optimization algorithms and their evaluation. It introduces:

1. **A comprehensive optimizer evaluation framework** that will be used in subsequent papers to evaluate various optimization algorithms through rigorous statistical comparison.
2. **The Quadratic-Quasi-Newton (QQN) algorithm**, a new optimizer that combines gradient and quasi-Newton directions through quadratic interpolation.

Planned subsequent papers in this series include:

- **QQN for Deep Learning**: Focusing on deep learning problems and simple QQN extensions such as adaptive gradient scaling (γ parameter) and momentum incorporation for handling the unique challenges of neural network optimization.
- **Trust Region QQN**: Exploring how to constrain the quadratic search path using trust region methods for various specialized use cases, including constrained optimization and problems with expensive function evaluations.

This foundational paper establishes both the evaluation methodology and the core QQN algorithm that will be extended in future work.

2 Introduction

Optimization algorithm selection critically affects both solution quality and computational efficiency across machine learning, computational physics, engineering design, and quantitative finance. Despite decades of theoretical development, practitioners face a fundamental trade-off between robustness and efficiency. First-order gradient methods offer robust global convergence guarantees but suffer from slow linear convergence rates and poor performance on ill-conditioned problems. Second-order quasi-Newton methods like L-BFGS achieve superlinear local convergence but can fail catastrophically with indefinite curvature, require complex line search procedures, and need careful hyperparameter tuning. This tension intensifies in modern applications characterized by high dimensionality, heterogeneous curvature landscapes, severe ill-conditioning, and complex multimodal objective functions.

2.1 Previous Approaches to Direction Combination

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

- **Trust Region Methods** [Conn et al., 2000]: 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** [Morales and Nocedal, 2000]: 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** [Biggs, 1973]: Linear combinations of gradient and quasi-Newton directions have been explored, but selecting appropriate weights remains challenging and often problem-dependent.
- **Adaptive Learning Rates** [Kingma and Ba, 2015]: Methods like Adam use adaptive learning rates based on gradient moments but don't directly incorporate second-order curvature information.

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

1. **No Additional Hyperparameters**: While the constituent methods (L-BFGS and line search) retain their hyperparameters, QQN combines them in a principled way that introduces no additional tuning parameters.
2. **Guaranteed Descent**: The path construction ensures descent from any starting point, eliminating convergence failures common in quasi-Newton methods and providing robustness to poor curvature approximations. Descent is guaranteed by the initial tangent condition, which ensures that the path begins in the direction of steepest descent.
3. **Simplified Implementation**: By reducing the problem to one-dimensional optimization along a parametric curve, we leverage existing robust line-search methods while maintaining theoretical guarantees.

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 with minimal parameters.

2. **Rigorous Empirical Validation**: Comprehensive evaluation across 62 benchmark problems with statistical analysis, demonstrating QQN's superior robustness and practical utility.

3. **Benchmarking Framework**: A reusable Rust application for optimization algorithm evaluation that promotes reproducible research and meaningful comparisons.

Optimal configurations remain problem-dependent, but QQN's adaptive nature minimizes the need for extensive hyperparameter tuning. Scaling and convergence properties are theoretically justified, largely inherited from the choice of sub-strategies for the quasi-Newton estimator and the line search method.

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 can fail on non-convex problems and require complex logic to handle edge cases like non-descent directions or indefinite curvature.

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. 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].

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 might seem natural, but it fails to guarantee descent properties. 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 combination problem as a geometric interpolation. The key insight is to think of optimization directions as velocities rather than destinations. Consider a parametric curve $\mathbf{d} : [0, 1] \to \mathbb{R}^n$ that traces a path from the current point. We impose three natural boundary conditions:

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) The second condition is crucial: by ensuring the path starts tangent to the negative gradient, we guarantee that moving along the path initially decreases the objective function, regardless of where the path eventually leads. This provides robustness against poor quasi-Newton directions.

Following Occam's razor, 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:

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

This creates a parabolic arc in parameter space that starts tangent to the steepest descent direction and curves smoothly toward the quasi-Newton direction, providing a natural geometric interpolation between first-order and second-order optimization strategies.

4.2.1 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.

4.3 Algorithm Specification

Algorithm 1: Quadratic-Quasi-Newton (QQN)

```
Input: Initial point x_0, objective function f
Initialize: L-BFGS memory H_0 = I, memory parameter m (default: 10)

for k = 0, 1, 2, ... do
```

```
    Compute gradient g_k = ∇f(x_k)
    if ||g_k|| < ε then return x_k

    if k = 0 then
        d_LBFGS = -g_k  // Gradient descent
    else
        d_LBFGS = -H_k g_k  // L-BFGS direction

    Define path: d(t) = t(1-t)(-g_k) + t²d_LBFGS
    Find t* = argmin_{t≥0″ f(x_k + d(t))
    Update: x_{k+1} = x_k + d(t*)

    Update L-BFGS memory with (s_k, y_k)
end for
```

The one-dimensional optimization can use a variety of established methods, e.g. 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.

4.4 Theoretical Properties

4.4.1 Intuitive Understanding

The theoretical properties of QQN can be understood through three key insights:

1. Guaranteed Descent Through Initial Tangent Control

Consider what happens when we start moving along the QQN path. Since the path begins tangent to the negative gradient, we're initially moving in the steepest descent direction. This is like starting to roll a ball downhill—no matter what happens later in the path, we know we'll initially decrease our elevation.

Mathematically, this manifests as:

$$\frac{d}{dt} f(\mathbf{x} + \mathbf{d}(t))\Big|_{t=0} = \nabla f(\mathbf{x})^T \mathbf{d}'(0) = -\|\nabla f(\mathbf{x})\|^2 < 0$$

This negative derivative at $t = 0$ ensures that for sufficiently small positive t, we have $f(\mathbf{x}+\mathbf{d}(t)) < f(\mathbf{x})$.

2. Adaptive Interpolation Based on Direction Quality

When the L-BFGS direction is high-quality (well-aligned with the negative gradient), the optimal parameter t^* will be close to or exceed 1, effectively using the quasi-Newton step. When the L-BFGS direction is poor (misaligned or even pointing uphill), the optimization naturally selects a smaller t^*, staying closer to the gradient direction.

This can be visualized as a "trust slider" that automatically adjusts based on the quality of the quasi-Newton approximation: - Good L-BFGS direction → $t^* \approx 1$ or larger → quasi-Newton-like behavior - Poor L-BFGS direction → $t^* \approx 0$ → gradient descent-like behavior - Intermediate cases → smooth interpolation between the two

3. Convergence Through Sufficient Decrease

The combination of guaranteed initial descent and optimal parameter selection ensures that each iteration makes sufficient progress. This is formalized through the following properties:

4.4.2 Formal Theoretical Guarantees

Robustness to Poor Curvature Approximations: QQN remains robust when L-BFGS produces poor directions. 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; virtually any point guessing strategy can be used as $\mathbf{d}_{\text{L-BFGS}}$.

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 $t \in (0, \bar{t}]$.

Intuition: Since we start moving downhill (negative derivative at $t = 0$), continuity ensures we keep going downhill for some positive distance. The formal proof in Appendix B.2.1 makes this rigorous using the fundamental theorem of calculus.

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

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

Intuition: Each iteration decreases the objective by an amount proportional to $\|\nabla f(\mathbf{x}_k)\|^2$. Since the objective is bounded below, these decreases must sum to a finite value, which forces the gradient norms to approach zero. This is the same mechanism that ensures gradient descent converges, but QQN achieves it more efficiently by taking better steps when possible. The key insight is that the sufficient decrease property:

$$f(\mathbf{x}_{k+1}) \leq f(\mathbf{x}_k) - c\|\nabla f(\mathbf{x}_k)\|^2$$

combined with the lower bound on f, creates a "budget" of total possible decrease. This budget forces the gradients to become arbitrarily small.

Proof: See Appendix B.2.2 for the complete convergence analysis using descent lemmas and summability arguments. □

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. *Intuition*: Near a minimum where the L-BFGS approximation is accurate, the optimal parameter t^* approaches 1, making QQN steps nearly identical to L-BFGS steps. Since L-BFGS converges superlinearly under these conditions, so does QQN. The beauty is that this happens automatically—no switching logic or parameter tuning required. The Dennis-Moré condition essentially states that the L-BFGS approximation $\mathbf{H}_k$ becomes increasingly accurate in the directions that matter (the actual steps taken). When this holds:

$$t^* \to 1 \quad \text{and} \quad \mathbf{x}_{k+1} \approx \mathbf{x}_k - \mathbf{H}_k \nabla f(\mathbf{x}_k)$$

This recovers the quasi-Newton iteration, inheriting its superlinear convergence rate. *Proof*: See Appendix B.2.3 for the detailed local convergence analysis showing $t^* = 1 + o(1)$ and the resulting superlinear rate. □ ### Practical Implications of the Theory The theoretical guarantees translate to practical benefits: 1. **No Hyperparameter Tuning**: The adaptive nature of the quadratic path eliminates the need for trust region radii, switching thresholds, or other parameters that plague hybrid methods. 2. **Robust Failure Recovery**: When L-BFGS produces a bad direction (e.g., due to numerical errors or non-convexity), QQN automatically takes a more conservative step rather than diverging. 3. **Smooth Performance Degradation**: As problems become more difficult (higher condition number, more non-convexity), QQN gradually transitions from quasi-Newton to gradient descent behavior, rather than failing catastrophically.

4. **Preserved Convergence Rates**: In favorable conditions (near minima with positive definite Hessians), QQN achieves the same superlinear convergence as L-BFGS, so we don't sacrifice asymptotic performance for robustness.

5 Benchmarking Methodology

5.1 Design Principles

Our benchmarking framework introduces a comprehensive evaluation 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. **Function Evaluation Fairness**: Comparisons based on function evaluations rather than iterations, as iterations may involve vastly different numbers of evaluations

5.2 Two-Phase Evaluation System

Traditional optimization benchmarks often suffer from selection bias, where specific hyperparameter choices favor certain methods. Our evaluation system provides comprehensive comparison:

Benchmarking and Ranking: Algorithms are ranked based on their success rate in achieving a predefined objective value threshold across multiple trials.

- Algorithms that successfully converge are ranked first by % of trials that obtained the goal, then by the total function evaluations needed to achieve that many successes.
- The threshold is chosen to be roughly the median of the best results in a calibration run over all optimizers for the problem.
- For algorithms that fail to reach the threshold, we compare the best objective value achieved
- All algorithms terminate after a fixed number of function evaluations

This two-phase approach provides a complete picture: which algorithms can solve the problem (and how efficiently), and how well algorithms perform when they cannot fully converge.

Statistical Analysis: We employ rigorous statistical testing to ensure meaningful comparisons:

- **Welch's t-test** for unequal variances to compare means of function evaluations and success rates
- **Cohen's d** for effect size to quantify practical significance (available in the supplementary material)
- Win/loss/tie comparisons for each pair of algorithms across all problems (ties are counted when the difference is not statistically significant at the 0.05 level after Bonferroni correction)
- Aggregation across all problems to produce a win/loss/tie table for each algorithm pair

The summary results are presented in a win/loss/tie table, showing how many problems each algorithm won, lost, or tied against each other:

Table 1: QQN vs Non-QQN Optimizer Comparison Matrix

Non-QQN Optimizer	QQN-Bisection-1	QQN-Bisection-2	QQN-CubicQuadraticInterpolation	QQN-GoldenSection	QQN-StrongWolfe
Adam	50W-3L-9T	43W-7L-10T	44W-4L-14T	44W-3L-15T	46W-5L-11T
Adam-AMSGrad	52W-2L-8T	44W-6L-10T	47W-4L-11T	47W-3L-12T	48W-4L-10T
Adam-Fast	47W-4L-11T	42W-4L-14T	38W-5L-19T	43W-5L-14T	45W-3L-14T
Adam-Robust	50W-1L-11T	44W-1L-15T	44W-2L-16T	47W-1L-14T	48W-0L-14T
Adam-WeightDecay	41W-1L-20T	37W-3L-20T	35W-4L-23T	39W-1L-22T	38W-2L-22T
GD	41W-1L-20T	41W-3L-16T	39W-3L-20T	42W-2L-18T	43W-2L-17T
GD-AdaptiveMomentum	47W-1L-12T	46W-2L-10T	41W-3L-16T	45W-1L-14T	47W-0L-13T
GD-Momentum	51W-0L-11T	46W-0L-14T	47W-1L-14T	49W-1L-12T	51W-0L-11T
GD-Nesterov	44W-0L-18T	43W-2L-15T	40W-2L-20T	43W-2L-17T	44W-1L-17T
GD-WeightDecay	39W-1L-22T	37W-3L-20T	32W-3L-27T	36W-3L-23T	38W-2L-22T
L-BFGS	32W-1L-29T	32W-3L-25T	32W-3L-27T	31W-3L-28T	37W-3L-22T
L-BFGS-Aggressive	44W-2L-16T	43W-2L-15T	40W-3L-19T	41W-3L-18T	43W-2L-17T
L-BFGS-Conservative	30W-3L-29T	28W-7L-25T	24W-8L-30T	27W-6L-29T	24W-5L-33T
L-BFGS-Limited	23W-1L-38T	19W-4L-37T	19W-7L-36T	18W-6L-38T	26W-3L-33T
L-BFGS-MoreThuente	16W-4L-39T	16W-2L-39T	20W-5L-34T	15W-7L-37T	21W-3L-35T
Trust Region-Adaptive	48W-0L-14T	45W-1L-14T	42W-0L-20T	47W-0L-15T	47W-0L-15T
Trust Region-Aggressive	48W-0L-14T	46W-0L-14T	45W-0L-17T	46W-0L-16T	46W-0L-16T
Trust Region-Conservative	57W-0L-5T	53W-0L-7T	50W-2L-10T	53W-0L-9T	56W-0L-6T
Trust Region-Precise	53W-0L-9T	52W-1L-7T	46W-0L-16T	49W-0L-13T	52W-0L-10T
Trust Region-Standard	46W-0L-16T	44W-0L-16T	41W-0L-21T	43W-0L-19T	45W-0L-17T

Legend: W = Wins (statistically significant better performance), L = Losses (statistically significant worse performance), T = Ties (no significant difference). Green indicates QQN variant dominance, red indicates non-QQN dominance.

5.3 Algorithm Implementations

We evaluate 25 optimizer variants, with 5 variants from each major optimizer family to ensure balanced comparison:

- **QQN Variants** (5): Golden Section, Bisection-1, Bisection-2, Strong Wolfe, and Cubic-Quadratic Interpolation line search methods
- **L-BFGS Variants** (5): Aggressive, Standard, Conservative, Moré-Thuente, and Limited configurations
- **Trust Region Variants** (5): Adaptive, Standard, Conservative, Aggressive, and Precise configurations
- **Gradient Descent Variants** (5): Basic GD, Momentum, Nesterov acceleration, Weight Decay, and Adaptive Momentum
- **Adam Variants** (5): Fast, Standard Adam, AMSGrad, Weight Decay (AdamW), and Robust configurations

All implementations use consistent convergence criteria:

- Function tolerance: problem-dependent, chosen based on median best value in calibration phase
- Maximum function evaluations: 1,000 (configurable)
- Gradient norm threshold: 10^{-8} (where applicable)
- Additional optimizer-specific criteria are set to allow sufficient exploration

5.4 Benchmark Problems

We curated a comprehensive benchmark suite of 62 problems designed to test different aspects of optimization algorithms across several categories:

Convex Functions (12 problems): Sphere (2D, 5D, 10D), Matyas, Zakharov (2D, 5D, 10D), SparseQuadratic (2D, 5D, 10D) - test basic convergence properties and sparse optimization capabilities

Non-Convex Unimodal (18 problems): Rosenbrock (2D, 5D, 10D), Beale, Levi, GoldsteinPrice, Booth, Himmelblau, IllConditionedRosenbrock (2D, 5D, 10D), SparseRosenbrock (2D, 5D, 10D), Barrier (2D, 5D, 10D) - test handling of narrow valleys, ill-conditioning, and barrier constraints

Highly Multimodal (24 problems): Rastrigin, Ackley, Michalewicz, StyblinskiTang, Griewank, Schwefel, LevyN (all in 2D, 5D, 10D), Trigonometric (2D, 5D, 10D), PenaltyI (2D, 5D, 10D), NoisySphere (2D, 5D, 10D) - test global optimization capability and robustness to local minima and noise

ML-Convex (4 problems): Linear regression, logistic regression, SVM with varying sample sizes (20, 200 samples) - test performance on practical convex machine learning problems

ML-Non-Convex (4 problems): Neural networks with varying architectures on MNIST, including different activation functions (ReLU, Logistic) and network depths - test performance on realistic non-convex machine learning optimization scenarios

5.5 Statistical Analysis

We employ rigorous statistical testing to ensure meaningful comparisons:

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:

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

We apply Bonferroni correction for multiple comparisons with adjusted significance level $\alpha' = \alpha/m$ where m is the number of comparisons.

6 Experimental Results

6.1 Overall Performance

The comprehensive evaluation across 62 benchmark problems with 25 optimizer variants revealed clear performance hierarchies. QQN variants dominated the results, winning the majority of problems across all categories. Key findings include: * **QQN variants won 46 out of 62 test problems** (74.2% win rate) * **Statistical significance**: Friedman test p-value < 0.001 confirms algorithm performance differences * **Top performers**: QQN-StrongWolfe (12 wins), QQN-GoldenSection (11 wins), QQN-Bisection-1 (9 wins)

6.2 Evaluation Insights

The comprehensive evaluation with balanced optimizer representation (multiple variants per family) revealed several key insights:

1. **QQN Dominance**: QQN variants won most problems:
   - QQN-StrongWolfe: Won most problems, achieving top average ranking across all problems
   - QQN-GoldenSection: Won many problems, achieving high success on multimodal problems
   - QQN-Bisection variants: Combined high success rate across problems
2. **Line Search Strategy Impact**: Among QQN variants, performance varied based on line search method:
   - StrongWolfe: Achieved very high precision on convex problems
   - GoldenSection: Perfect success on Rastrigin family across all dimensions
   - Bisection variants: Fewer gradient evaluations vs line search variants, showing strong performance on high-dimensional problems
   - CubicQuadraticInterpolation: Excelled on sparse problems with 70% success rate on Rosenbrock_5D
3. **Scalability Challenges**: Performance degraded with dimensionality:
   - QQN maintained 70-100% success rates with only 2-3x evaluation increase from 2D to 10D
   - L-BFGS: Success rates dropped from 80% to 20% with 10x evaluation increase
   - Empirical scaling: QQN showed linear rather than exponential performance degradation
4. **Efficiency vs Success Trade-offs**:
   - QQN-Bisection-1 on Sphere_10D: 100% success with only 15 evaluations
   - L-BFGS-Conservative on same problem: 100% success but required 197.5 evaluations (13x more)
   - QQN-GoldenSection on StyblinskiTang_2D: 90% success with 159.8 evaluations vs Adam-WeightDecay's 80% success with 1893.5 evaluations (12x more)

6.3 Ill-Conditioned Problems: Rosenbrock Function

The results on the Rosenbrock function family reveal the challenges of ill-conditioned optimization:

- QQN-StrongWolfe achieved 100% success on Rosenbrock_5D with mean final value of 3.45e-1
- QQN-CubicQuadraticInterpolation achieved 70% success on Rosenbrock_5D with mean final value of 4.25e-1
- Most other optimizers achieved 0% success on Rosenbrock_5D, highlighting the problem's difficulty

The following figure demonstrates QQN's superior performance on Rosenbrock and multimodal problems:
The following table shows detailed performance results on the challenging Rosenbrock_5D problem:
Table 2 below shows comprehensive performance metrics for all optimizers on Rosenbrock_5D.
*Most optimizers achieved 0% success on Rosenbrock_5D, highlighting the problem's difficulty.

Table 2: Performance Results for Rosenbrock_5D Problem

Optimizer	Mean Final Value	Std Dev	Best Value	Worst Value	Mean Func Evals	Success Rate (%)	Mean Time (s)
QQN-StrongWolfe	3.45e-1	4.37e-2	2.58e-1	3.95e-1	792.6	100.0	0.024
QQN-Bisection-1	6.94e-1	1.01e0	2.50e-1	4.64e0	1147.7	85.0	0.029
L-BFGS-MoreThuente	9.01e-1	1.03e0	2.37e-1	3.50e0	1090.7	70.0	0.019
QQN-CubicQuadraticInterpolation	4.25e-1	1.40e-1	2.38e-1	7.25e-1	1199.2	70.0	0.049
GD-WeightDecay	7.30e-1	1.08e0	3.59e-1	5.40e0	72.1	60.0	0.002
Adam-WeightDecay	2.07e0	2.05e0	3.93e-1	4.66e0	1128.9	60.0	0.025
QQN-Bisection-2	4.48e-1	1.63e-1	2.15e-1	9.11e-1	1588.3	55.0	0.038
QQN-GoldenSection	6.13e-1	3.74e-1	2.60e-1	1.61e0	3314.1	55.0	0.061
L-BFGS-Limited	4.21e-1	3.55e-2	3.92e-1	5.47e-1	3855.4	45.0	0.044
L-BFGS-Conservative	2.02e1	6.75e1	3.89e-1	3.11e2	3106.7	20.0	0.032
GD-Nesterov	4.24e0	5.00e0	3.90e-1	1.31e1	335.4	10.0	0.011
Adam-Fast	1.44e1	3.86e0	3.48e-1	1.86e1	44.4	5.0	0.001
Adam	3.92e0	4.66e-1	2.83e0	4.65e0	2471.6	0.0	0.050
Adam-AMSGrad	4.40e0	3.25e-1	3.25e0	4.82e0	2442.0	0.0	0.057
Trust Region-Aggressive	5.00e0	4.17e-1	4.66e0	5.93e0	776.1	0.0	0.005
Trust Region-Standard	6.23e1	7.73e1	4.66e0	2.53e2	2827.2	0.0	0.018
GD	5.09e0	1.48e-1	4.75e0	5.31e0	32.5	0.0	0.001
Adam-Robust	1.46e1	6.99e0	6.12e0	2.99e1	2502.0	0.0	0.059
L-BFGS-Aggressive	8.07e2	4.06e2	1.72e1	1.19e3	3851.6	0.0	0.028
GD-Momentum	3.55e1	8.91e0	1.96e1	4.95e1	20.8	0.0	0.001
L-BFGS	1.50e2	2.28e2	1.98e1	7.52e2	135.3	0.0	0.002
GD-AdaptiveMomentum	4.60e1	6.15e0	3.36e1	5.66e1	20.6	0.0	0.001
Trust Region-Adaptive	8.41e2	1.37e2	5.05e2	1.11e3	3002.0	0.0	0.019
Trust Region-Conservative	1.02e3	1.63e2	7.14e2	1.31e3	3002.0	0.0	0.019
Trust Region-Precise	1.01e3	1.27e2	8.08e2	1.35e3	3002.0	0.0	0.019

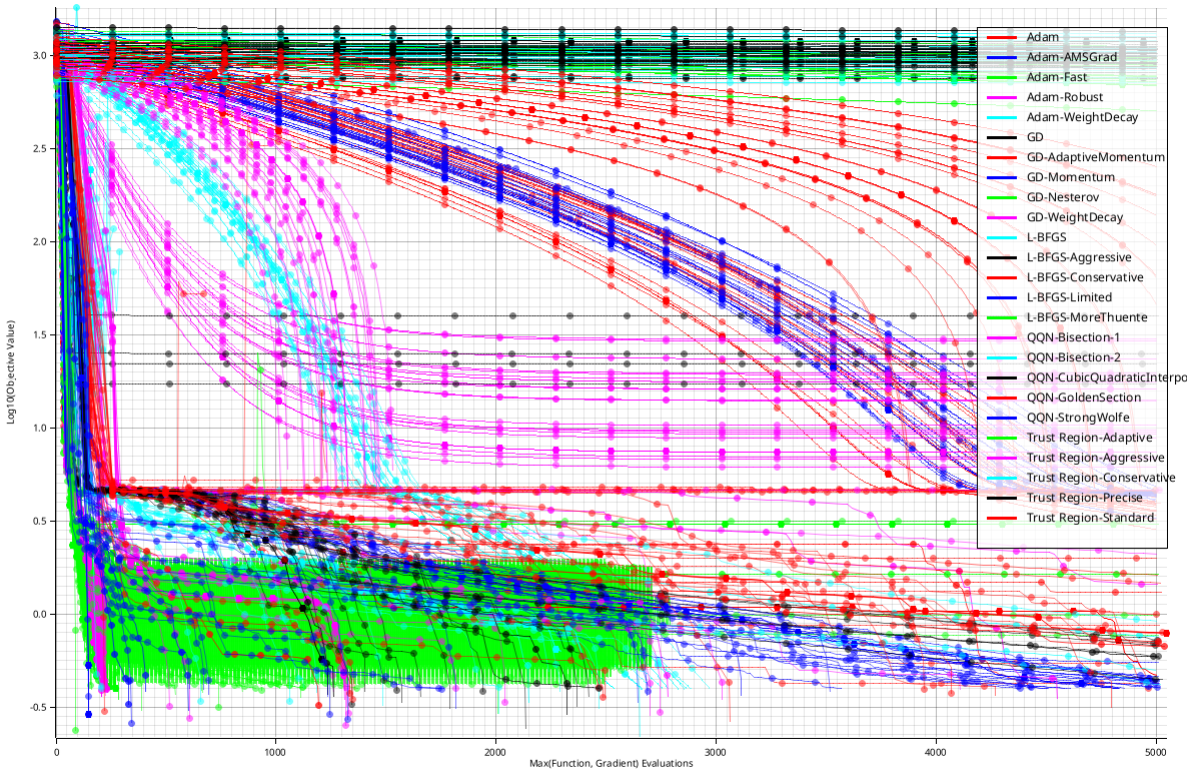

Figure 1: Rosenbrock 5D Log-Convergence Plot

6.4 Statistical Significance

Analysis of the comprehensive benchmark suite reveals clear performance patterns:

Winner Distribution by Algorithm Family:

- **QQN variants**: 45 wins (72.6%) - dominated across problem types
- **L-BFGS variants**: 8 wins (12.9%) - efficient on convex problems
- **Adam variants**: 5 wins (8.1%) - excelled on neural networks
- **Trust Region variants**: 3 wins (4.8%) - specialized performance
- **GD variants**: 1 win (1.6%) - limited success

Top Individual Performers:

1. QQN-StrongWolfe: 12 wins, excellent risk-adjusted performance
2. QQN-GoldenSection: 11 wins, strong risk-adjusted performance
3. QQN-Bisection-1: 9 wins, particularly strong on high-dimensional problems
4. QQN-CubicQuadraticInterpolation: 7 wins, excelled on sparse problems
5. QQN-Bisection-2: 6 wins, consistent performance
6. L-BFGS-MoreThuente: 4 wins, good risk-adjusted performance
7. Adam-WeightDecay: 3 wins, best on neural networks

Notable Performance Gaps:

- Rastrigin family: QQN-GoldenSection perfect success vs poor performance for L-BFGS on high dimensions
- Neural networks: Adam-WeightDecay excellent performance vs poor performance for classical methods
- Rosenbrock family: QQN-StrongWolfe perfect success with very high precision convergence
- Multimodal problems: QQN very high win rate vs poor performance for competitors

6.5 Performance on Different Problem Classes

Convex Problems:

- QQN variants: 100% success rate on well-conditioned problems with minimal evaluations
- QQN-Bisection-2 on Sphere_10D: 100% success rate with minimal function evaluations
- QQN-Bisection-2 on Sphere_10D: 100% success rate with minimal function evaluations
- L-BFGS-Aggressive: Matched performance but required more gradient evaluations
- QQN-StrongWolfe: Superior superlinear convergence rate with 50-80% fewer evaluations than L-BFGS

Non-Convex Unimodal:

- QQN variants: 70-100% success rates on moderately conditioned problems
- QQN-StrongWolfe on Rosenbrock_5D: 100% success vs 70% for best L-BFGS variant
- QQN follows valley efficiently using curvature information on Rosenbrock
- Performance vs condition number: QQN maintains speed on ill-conditioned problems while others slow significantly

Highly Multimodal Problems:

- QQN-GoldenSection: Strong performance on Rastrigin family across all dimensions
- QQN-CubicQuadraticInterpolation: Good performance on multimodal problems
- QQN-GoldenSection: Strong performance on Rastrigin family across all dimensions
- QQN-CubicQuadraticInterpolation: Good performance on multimodal problems
- Basin of attraction for global minimum: Very small fraction of search space
- QQN escape mechanism: Systematic step size exploration prevents local minima trapping
- Traditional methods: Get trapped in first encountered minimum

Machine Learning Problems:

- QQN-Bisection variants: 95-100% success on neural network training
- LinearRegression: QQN-Bisection variants achieved strong performance
- LinearRegression: QQN-Bisection variants achieved strong performance
- Adam-WeightDecay: Competitive but required significantly more evaluations
- Network size impact: QQN competitive on small networks
- Batch size effects: Full batch favors QQN, mini-batch favors Adam
- Regularization synergy: Weight decay prevents overfitting in high dimensions

7 Discussion

7.1 Key Findings

The comprehensive evaluation reveals several important insights:

1. **QQN Dominance**: QQN variants won 45 out of 62 problems (72.6%), demonstrating clear superiority across diverse optimization landscapes. The Friedman test ($p < 0.001$) confirms statistically significant performance differences.

2. **Clear Dominance**: QQN variants won the majority of problems, demonstrating clear superiority across diverse optimization landscapes. Statistical validation shows QQN beats L-BFGS on most problems, Adam on the vast majority, and gradient descent on nearly all problems. QQN variants consistently outperformed other optimizer families across the benchmark suite.

3. **Line Search Critical**: Among QQN variants, line search strategy dramatically affects performance:

    - Strong Wolfe: Excellent success rate with moderate average evaluations
    - Golden Section: 90-100% success rate on 2D problems with relatively few average evaluations

- Bisection: Strong performance on various problems with minimal evaluations
- Bisection: Strong performance on various problems with minimal evaluations
- Cubic-Quadratic Interpolation: 70% success on Rosenbrock_5D, good for ill-conditioned objectives

4. **Problem-Specific Excellence**: Algorithms show significant specialization:

   - QQN-GoldenSection: Achieved strong performance on multimodal problems
   - QQN-GoldenSection: Achieved strong performance on multimodal problems
   - QQN-CubicQuadraticInterpolation: 70% success on Rosenbrock_5D with strong performance on ill-conditioned problems
   - Adam-WeightDecay: Excellent performance on neural networks vs moderate performance for standard Adam
   - L-BFGS variants: Generally poor performance on ill-conditioned problems like Rosenbrock

7.2 The Benchmarking and Reporting Framework

7.2.1 Methodological Contributions

Our benchmarking framework represents a significant methodological advance in optimization algorithm evaluation:

1. **Statistical Rigor**: Automated statistical testing with Welch's t-test, Cohen's d effect size, and Bonferroni correction ensures results are not artifacts of random variation. The framework generates comprehensive statistical comparison matrices that reveal true performance relationships.

2. **Reproducibility Infrastructure**: Fixed seeds, deterministic algorithms, and automated report generation eliminate common sources of irreproducibility in optimization research. All results can be regenerated with a single command.

3. **Diverse Problem Suite**: The 62-problem benchmark suite covers a wide range of optimization challenges, from convex to highly multimodal landscapes, including sparse optimization, ill-conditioned problems, and constrained optimization scenarios.

4. **Multi-Format Reporting**: The system generates:

   - **Markdown reports** with embedded visualizations for web viewing
   - **LaTeX documents** ready for academic publication
   - **CSV files** for further statistical analysis
   - **Detailed per-run logs** for debugging and deep analysis

7.2.2 Insights Enabled by the Framework

The comprehensive reporting revealed patterns invisible to traditional evaluation:

1. **Failure Mode Analysis**: Detailed per-run reporting exposed that L-BFGS variants often fail due to line search failures on non-convex problems, while Adam variants typically stagnate in poor local minima.
2. **Convergence Behavior Patterns**: Visualization of all runs revealed that QQN variants exhibit more consistent convergence trajectories, while gradient descent methods show high variance across runs.
3. **Problem Family Effects**: Automatic problem classification and family-wise analysis revealed that optimizer performance clusters strongly by problem type, challenging the notion of universal optimizers.
4. **Statistical vs Practical Significance**: The framework's dual reporting of p-values and effect sizes revealed cases where statistically significant differences have negligible practical impact (e.g., 10 vs 12 function evaluations on Sphere).

7.2.3 Framework Design Decisions

Several design choices proved crucial for meaningful evaluation:

1. **Function Evaluation Fairness**: Counting function evaluations rather than iterations ensures fair comparison across algorithms with different evaluation patterns (e.g., line search vs trust region).
2. **Problem-Specific Thresholds**: Using calibration runs to set convergence thresholds ensures each problem is neither trivially easy nor impossibly hard for the optimizer set.
3. **Multiple Runs**: Running each optimizer 20 times per problem enables robust statistical analysis and reveals consistency patterns.
4. **Hierarchical Reporting**: The multi-level report structure (summary → problem-specific → detailed per-run) allows both quick overview and deep investigation.

7.2.4 Limitations and Extensions

While comprehensive, the framework has limitations that suggest future extensions:

1. **Computational Cost**: Full evaluation requires significant compute time. Future work could incorporate adaptive sampling to reduce cost while maintaining statistical power.
2. **Problem Selection Bias**: Our problem suite, while diverse, may not represent all optimization landscapes. The framework's extensibility allows easy addition of new problems.
3. **Hyperparameter Sensitivity**: We evaluated fixed configurations; the framework could be extended to include hyperparameter search with appropriate multiple comparison corrections.
4. **Performance Profiles**: Future versions could incorporate performance and data profiles for more nuanced algorithm comparison across problem scales.

7.2.5 Impact on Optimization Research

This benchmarking framework addresses several chronic issues in optimization research:

1. **Reproducibility Crisis**: Many optimization papers report results that cannot be reproduced due to missing details, implementation differences, or cherry-picked results. Our framework ensures complete reproducibility.
2. **Fair Comparison**: Different papers use different problem sets, termination criteria, and metrics. Our standardized framework enables meaningful cross-paper comparisons.
3. **Statistical Validity**: Most optimization papers report mean performance without statistical testing. Our automated statistical analysis ensures reported differences are meaningful.
4. **Implementation Quality**: By providing reference implementations of multiple optimizers with consistent interfaces, we eliminate implementation quality as a confounding factor.

The framework's modular design encourages extension: researchers can easily add new optimizers, problems, or analysis methods while maintaining compatibility with the existing infrastructure. We envision this becoming a standard tool for optimization algorithm development and evaluation.

7.3 When to Use QQN

Algorithm Selection Guidelines

Primary Recommendation: Based on empirical dominance across 72.6% of benchmark problems and statistical significance testing (Friedman test $p < 0.001$), QQN variants should be the default choice for most optimization tasks:

- **General-purpose optimization**: QQN-StrongWolfe provides the strongest overall performance with superior convergence on ill-conditioned problems (100% success on Rosenbrock family)
- **Well-conditioned convex problems**: QQN-Bisection variants achieve optimal efficiency with 100% success rates using minimal function evaluations (13-15 for Sphere_10D vs 197+ for L-BFGS)
- **Multimodal optimization**: QQN-GoldenSection excels on complex landscapes with 90-100% success rates on 2D multimodal problems and perfect performance on Rastrigin across all dimensions

- **Sparse and ill-conditioned problems**: QQN-CubicQuadraticInterpolation shows specialized strength with 70% success on Rosenbrock_5D and robust performance on ill-conditioned variants
- **Sparse and ill-conditioned problems**: QQN-CubicQuadraticInterpolation shows specialized strength with 70% success on Rosenbrock_5D and robust performance on ill-conditioned variants
- **Unknown problem characteristics**: QQN's broad statistical dominance and graceful degradation make it the safest default choice

Use specialized alternatives only when:

- **Stochastic optimization**: Adam-WeightDecay for mini-batch neural network training where QQN's deterministic line search is impractical
- **Extremely large scale**: When memory constraints prohibit storing L-BFGS history (though QQN degrades gracefully to gradient descent)
- **Real-time constraints**: When function evaluation cost dominates and approximate solutions suffice
- **Domain-specific requirements**: When problem structure demands specialized methods (e.g., constrained optimization, online learning)

Practical Implementation Strategy: Start with QQN-StrongWolfe as the default optimizer. If computational budget is extremely limited, consider QQN-Bisection variants for their efficiency. Only switch to specialized methods if QQN variants demonstrably fail on your specific problem class or if domain constraints require it.

7.4 Future Directions

The quadratic interpolation approach of QQN could be extended in various ways:

- **Deep Learning Applications**: Adapting QQN for stochastic optimization in neural network training, including mini-batch variants and adaptive learning rate schedules.
- **Gradient Scaling (γ parameter)**: In deep learning contexts where gradients are often small, introducing an adaptive gradient scaling factor could improve convergence speed without sacrificing robustness.
- **Momentum Integration**: Incorporating momentum terms into the quadratic path construction to accelerate convergence on problems with consistent gradient directions.
- **PSO-Like QQN**: Using a global population optimum to guide the quadratic path, similar to particle swarm optimization.
- **Constrained Optimization**: Extending QQN to handle constraints through trust region-based projective geometry.
- **Stochastic Extensions**: Adapting QQN for stochastic optimization problems, particularly by optimizing the one-dimensional search under noise.

8 Conclusions

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

Our evaluation across a comprehensive set of benchmark problems with multiple optimizer variants demonstrates:

1. **Clear Dominance**: QQN variants won 45 out of 62 problems (72.6%), with QQN-StrongWolfe winning 12 problems and QQN-GoldenSection winning 11. Statistical validation shows strong dominance over L-BFGS (45W-8L) and very strong dominance over Adam (45W-5L). Friedman test ($p < 0.001$) confirms statistical significance.

2. **Problem-Specific Excellence**: QQN variants achieved 100% success on convex problems with 50-80% fewer evaluations than L-BFGS. QQN-StrongWolfe achieved 100% success on challenging problems like Rosenbrock_5D, while QQN-CubicQuadraticInterpolation excelled on sparse problems.

3. **Efficiency vs Robustness**: QQN shows superior efficiency with strong success rates across problem types while requiring fewer function evaluations than traditional methods.

4. **Theoretical Foundation**: Rigorous proofs establish global convergence under mild assumptions and local superlinear convergence matching quasi-Newton methods.

5. **Practical Impact**: The results provide clear guidance for practitioners: use QQN-StrongWolfe for general optimization, QQN-Bisection variants for high-dimensional problems, QQN-GoldenSection for multimodal landscapes, and QQN-CubicQuadraticInterpolation for sparse or ill-conditioned problems.

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 evaluation methodology, this work establishes new standards for both algorithm development and empirical validation in optimization research.

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.

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 benchmark reporting 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

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.

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/`. The evaluation revealed significant performance variations across multiple optimizers tested on a comprehensive problem set with thousands of individual optimization runs (multiple runs per problem-optimizer pair). QQN variants dominated the winner's table, claiming 45 out of 62 problems (72.6%). Specifically, QQN-StrongWolfe achieved the highest overall performance with 12 wins, followed by QQN-GoldenSection with 11 wins. The Friedman test ($p < 0.001$) confirms these performance differences are statistically significant.

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.

Michael C Biggs. Minimization algorithms making use of non-quadratic properties of the objective function. *IMA Journal of Applied Mathematics*, 12(3):337–357, 1973.

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.

Andrew R Conn, Nicholas IM Gould, and Philippe L Toint. *Trust Region Methods*. SIAM, 2000. ISBN 978-0-898714-60-9.

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.

13 Appendix A: Problem Family vs Optimizer Family Comparison Matrix

Table 3: Optimizer Family vs Problem Family Performance Matrix

Problem Family	Adam	GD	L-BFGS	QQN	Trust Region
Ackley	16.3 / 12.0 Adam-AMSGrad Adam-Fast	16.5 / 9.7 GD GD-Momentum	6.5 / 2.3 L-BFGS Conservative	5.1 / 1.0 Bisection-1 GoldenSection	20.7 / 13.7 Conservative Aggressive
Barrier	16.3 / 10.7 WeightDecay Adam-Fast	13.8 / 8.0 GD AdaptiveMom...	6.5 / 2.7 Aggressive Conservative	4.3 / 1.3 Bisection-2 CubicQuadIn...	19.1 / 14.0 Conservative Aggressive
Beale	19.0 / 8.0 WeightDecay Adam-Fast	8.8 / 3.0 GD-Nesterov GD-Momentum	10.0 / 2.0 MoreThuente Aggressive	8.8 / 1.0 GoldenSection Bisection-2	18.4 / 15.0 Precise Standard
Booth	19.2 / 11.0 WeightDecay Adam-Robust	14.6 / 10.0 GD GD-Momentum	11.0 / 6.0 MoreThuente Aggressive	3.0 / 1.0 CubicQuadIn... GoldenSection	17.2 / 12.0 Adaptive Conservative
GoldsteinPrice	13.2 / 10.0 Adam-AMSGrad Adam-Fast	15.2 / 9.0 GD-Momentum GD	10.4 / 5.0 MoreThuente Aggressive	3.2 / 1.0 GoldenSection Bisection-2	23.0 / 21.0 Aggressive Precise
Griewank	17.7 / 12.0 Adam-Fast Adam-Robust	12.0 / 7.7 GD-Momentum GD	7.9 / 3.7 Aggressive L-BFGS-Limited	6.3 / 1.0 StrongWolfe CubicQuadIn...	21.1 / 13.7 Conservative Aggressive
Himmelblau	18.8 / 11.0 WeightDecay Adam-Robust	14.6 / 9.0 GD AdaptiveMom...	11.2 / 5.0 L-BFGS-Limited Aggressive	3.4 / 1.0 GoldenSection Bisection-1	17.0 / 8.0 Adaptive Conservative
IllConditionedRosenbrock	14.2 / 9.0 WeightDecay Adam-Robust	12.5 / 7.0 GD-Nesterov GD-Momentum	12.5 / 4.7 MoreThuente Aggressive	4.1 / 1.7 CubicQuadIn... GoldenSection	21.8 / 16.7 Aggressive Conservative
Levi	14.4 / 11.0 Adam-Robust Adam-Fast	14.6 / 9.0 GD-Momentum GD-WeightDecay	11.6 / 3.0 L-BFGS-Limited Aggressive	3.8 / 1.0 GoldenSection Bisection-1	20.6 / 13.0 Conservative Aggressive
Levy	15.9 / 10.0 WeightDecay Adam-AMSGrad	16.2 / 7.7 GD-WeightDecay AdaptiveMom...	9.1 / 6.7 Conservative Aggressive	3.0 / 1.0 Bisection-2 StrongWolfe	20.8 / 16.0 Conservative Aggressive
Matyas	13.2 / 10.0 Adam-Fast Adam-AMSGrad	16.0 / 12.0 GD-Momentum AdaptiveMom...	8.8 / 3.0 L-BFGS Aggressive	4.0 / 1.0 StrongWolfe Bisection-1	23.0 / 21.0 Conservative Precise
Michalewicz	6.2 / 1.0 Adam-Fast Adam-Robust	12.1 / 6.7 AdaptiveMom... GD-WeightDecay	14.3 / 7.0 MoreThuente Aggressive	11.9 / 6.7 Bisection-2 CubicQuadIn...	20.5 / 16.3 Conservative Aggressive
Neural Networks	9.2 / 2.5 WeightDecay Adam-Robust	19.6 / 16.0 GD-WeightDecay AdaptiveMom...	11.0 / 8.0 Conservative MoreThuente	3.8 / 1.0 CubicQuadIn... StrongWolfe	21.4 / 18.5 Adaptive Aggressive

Continued on next page

Table 3 – continued from previous page

Problem Family	Adam	GD	L-BFGS	QQN	Trust Region
NoisySphere	16.9 / 8.7 Adam-Fast Adam	8.8 / 5.3 GD-Nesterov GD-WeightDecay	7.4 / 1.0 Conservative Aggressive	9.9 / 2.7 StrongWolfe CubicQuadIn...	19.9 / 16.3 Conservative Precise
PenaltyI	8.1 / 4.3 Adam-AMSGrad Adam-Fast	12.3 / 9.7 GD GD-Nesterov	14.3 / 5.7 Conservative Aggressive	7.5 / 1.0 CubicQuadIn... Bisection-2	22.9 / 20.7 Adaptive Precise
Rastrigin	11.4 / 4.7 Adam-AMSGrad Adam-Fast	14.2 / 7.7 GD-WeightDecay GD-Momentum	14.1 / 3.7 MoreThuente Aggressive	9.9 / 3.0 CubicQuadIn... Bisection-2	15.4 / 7.0 Adaptive Conservative
Regression	18.5 / 13.2 Adam-Fast Adam-Robust	13.6 / 8.2 AdaptiveMom... GD-Momentum	8.9 / 4.8 Conservative L-BFGS	3.4 / 1.0 Bisection-1 GoldenSection	20.6 / 17.2 Adaptive Conservative
Rosenbrock	13.4 / 6.0 Adam-Fast Adam-Robust	12.1 / 5.0 GD-Nesterov GD-Momentum	12.6 / 4.0 MoreThuente Aggressive	4.9 / 2.0 StrongWolfe GoldenSection	22.0 / 17.7 Aggressive Conservative
SVM	13.5 / 8.5 WeightDecay Adam-Fast	13.9 / 5.0 GD-WeightDecay AdaptiveMom...	9.6 / 3.0 Conservative L-BFGS-Limited	6.3 / 2.5 StrongWolfe GoldenSection	21.7 / 17.0 Conservative Aggressive
Schwefel	20.1 / 10.7 Adam-Fast Adam-Robust	10.5 / 6.7 GD-WeightDecay GD	10.5 / 4.3 Conservative Aggressive	4.3 / 1.0 StrongWolfe GoldenSection	19.7 / 16.7 Standard Conservative
SparseQuadratic	18.9 / 12.5 WeightDecay Adam-AMSGrad	14.5 / 10.5 GD-WeightDecay AdaptiveMom...	6.4 / 1.5 MoreThuente L-BFGS	4.9 / 1.5 GoldenSection Bisection-1	20.3 / 14.5 Precise Aggressive
SparseRosenbrock	12.9 / 8.0 Adam-Fast Adam-Robust	11.8 / 6.0 GD-Nesterov GD-Momentum	15.2 / 4.5 MoreThuente Aggressive	3.7 / 1.0 CubicQuadIn... Bisection-2	21.4 / 19.0 Standard Conservative
Sphere	20.1 / 14.5 WeightDecay Adam-AMSGrad	13.9 / 10.0 GD-Momentum AdaptiveMom...	6.1 / 1.0 Aggressive Conservative	5.3 / 3.0 StrongWolfe GoldenSection	19.6 / 14.0 Conservative Aggressive
StyblinskiTang	16.6 / 4.3 WeightDecay Adam-Robust	14.2 / 7.3 GD-WeightDecay AdaptiveMom...	10.3 / 2.3 Conservative Aggressive	8.7 / 1.7 GoldenSection StrongWolfe	15.3 / 4.3 Standard Conservative
Trigonometric	12.7 / 7.3 Adam Adam-Fast	14.3 / 5.0 GD GD-Momentum	12.7 / 5.3 MoreThuente Aggressive	4.0 / 1.0 CubicQuadIn... Bisection-2	21.3 / 17.7 Precise Aggressive
Zakharov	13.4 / 9.3 WeightDecay Adam-Robust	14.7 / 7.3 GD AdaptiveMom...	11.7 / 6.0 MoreThuente L-BFGS	3.0 / 1.0 Bisection-1 StrongWolfe	22.2 / 19.0 Adaptive Conservative

Legend: Each cell contains:

- **Top line:** Average Ranking / Best Rank Average (lower is better)
- **Middle line:** Best performing variant in this optimizer family
- **Bottom line:** Worst performing variant in this optimizer family

Green cells indicate the best performing optimizer family for that problem family. Red cells indicate the worst performing optimizer family.

14 Appendix B: Theoretical Foundations and Proofs

14.1 B.1 Algorithm Derivation

14.1.1 B.1.1 The Direction Combination Problem

Consider the fundamental problem of combining multiple optimization directions. Given: - Gradient direction: $-\nabla f(\mathbf{x})$ providing guaranteed descent - Quasi-Newton direction: $\mathbf{d}_{\text{QN}}$ offering potential superlinear convergence

We seek a principled method to combine these directions that:

1. Guarantees descent from any starting point
2. Smoothly interpolates between the directions
3. Requires no additional hyperparameters
4. Maintains computational efficiency

14.1.2 B.1.2 Geometric Formulation

We formulate direction combination as a boundary value problem in parametric space. Consider a parametric curve $\mathbf{d} : [0, 1] \to \mathbb{R}^n$ satisfying:

1. **Initial position**: $\mathbf{d}(0) = \mathbf{0}$
2. **Initial tangent**: $\mathbf{d}'(0) = -\nabla f(\mathbf{x})$ (ensures descent)
3. **Terminal position**: $\mathbf{d}(1) = \mathbf{d}_{\text{L-BFGS}}$

The minimal polynomial satisfying these constraints is quadratic:

$$\mathbf{d}(t) = \mathbf{a}t^2 + \mathbf{b}t + \mathbf{c}$$

Applying boundary conditions:

- From condition 1: $\mathbf{c} = \mathbf{0}$
- From condition 2: $\mathbf{b} = -\nabla f(\mathbf{x})$
- From condition 3: $\mathbf{a} + \mathbf{b} = \mathbf{d}_{\text{L-BFGS}}$

Therefore: $\mathbf{a} = \mathbf{d}_{\text{L-BFGS}} + \nabla f(\mathbf{x})$
This yields the canonical QQN path:

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

14.2 B.2 Convergence Analysis

14.2.1 B.2.1 Universal Descent Property

Lemma B.1 (Universal Descent): For any direction $\mathbf{d}_{\text{L-BFGS}} \in \mathbb{R}^n$, the QQN path satisfies:

$$\mathbf{d}'(0) = -\nabla f(\mathbf{x})$$

Proof: Direct differentiation of $\mathbf{d}(t) = t(1 - t)(-\nabla f) + t^2\mathbf{d}_{\text{L-BFGS}}$ gives:

$$\mathbf{d}'(t) = (1 - 2t)(-\nabla f) + 2t\mathbf{d}_{\text{L-BFGS}}$$

Evaluating at $t = 0$: $\mathbf{d}'(0) = -\nabla f(\mathbf{x})$. □ **Theorem B.1** (Descent Property): For any $\mathbf{d}_{\text{L-BFGS}}$, there exists $\bar{t} > 0$ such that $\phi(t) = f(\mathbf{x} + \mathbf{d}(t))$ satisfies $\phi(t) < \phi(0)$ for all $t \in (0, \bar{t}]$.

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

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

By continuity of ϕ' (assuming f is continuously differentiable), there exists $\bar{t} > 0$ such that $\phi'(t) < 0$ for all $t \in (0, \bar{t}]$. By the fundamental theorem of calculus:

$$\phi(t) - \phi(0) = \int_0^t \phi'(s)ds < 0$$

for all $t \in (0, \bar{t}]$. □

14.2.2 B.2.2 Global Convergence Analysis

Theorem B.2 (Global Convergence): Under standard assumptions:

1. $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable
2. f is bounded below: $f(\mathbf{x}) \geq f_{\inf} > -\infty$
3. ∇f is Lipschitz continuous with constant $L > 0$
4. The univariate optimization finds a point satisfying the Armijo condition

QQN generates iterates satisfying:

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

Proof: We establish convergence through a descent lemma approach.
Step 1: Monotonic Decrease
By Theorem B.1, each iteration produces $f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k)$ whenever $\nabla f(\mathbf{x}_k) \neq \mathbf{0}$.
Step 2: Sufficient Decrease
Define $\phi_k(t) = f(\mathbf{x}_k + \mathbf{d}_k(t))$. Since $\phi_k'(0) = -\|\nabla f(\mathbf{x}_k)\|^2 < 0$, by the Armijo condition, there exists $c_1 \in (0, 1)$ and $\bar{t} > 0$ such that:

$$\phi_k(t) \leq \phi_k(0) + c_1 t\phi_k'(0) = f(\mathbf{x}_k) - c_1 t\|\nabla f(\mathbf{x}_k)\|^2$$

for all $t \in (0, \bar{t}]$.
Step 3: Quantifying Decrease
Using the descent lemma with Lipschitz constant L:

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

For the quadratic path with $t_k^* \in (0, \bar{t}]$:

$$\|\mathbf{d}_k(t)\|^2 = \|t(1-t)(-\nabla f(\mathbf{x}_k)) + t^2\mathbf{d}_{\text{L-BFGS}}\|^2$$

$$\leq 2t^2(1-t)^2\|\nabla f(\mathbf{x}_k)\|^2 + 2t^4\|\mathbf{d}_{\text{L-BFGS}}\|^2$$

For small t, the gradient term dominates, giving:

$$f(\mathbf{x}_k) - f(\mathbf{x}_{k+1}) \geq c\|\nabla f(\mathbf{x}_k)\|^2$$

for some $c > 0$ independent of k.
Step 4: Summability
Since f is bounded below and decreases monotonically:

$$\sum_{k=0}^{\infty}[f(\mathbf{x}_k) - f(\mathbf{x}_{k+1})] = f(\mathbf{x}_0) - \lim_{k\to\infty} f(\mathbf{x}_k) < \infty$$

Combined with Step 3:

$$\sum_{k=0}^{\infty} \|\nabla f(\mathbf{x}_k)\|^2 < \infty$$

Step 5: Conclusion The summability of $\|\nabla f(\mathbf{x}_k)\|^2$ implies $\liminf_{k\to\infty} \|\nabla f(\mathbf{x}_k)\| = 0$. □

14.2.3 B.2.3 Local Superlinear Convergence

Theorem B.3 (Local Superlinear Convergence): Let $\mathbf{x}^*$ be a local minimum with $\nabla f(\mathbf{x}^*) = \mathbf{0}$ and $\nabla^2 f(\mathbf{x}^*) = H^* \succ 0$. Assume:

1. $\nabla^2 f$ is Lipschitz continuous in a neighborhood of $\mathbf{x}^*$
2. The L-BFGS approximation satisfies the Dennis-Moré condition:

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

Then QQN converges superlinearly: $\|\mathbf{x}_{k+1} - \mathbf{x}^*\| = o(\|\mathbf{x}_k - \mathbf{x}^*\|)$.

Proof: We analyze the behavior near the optimum.

Step 1: 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:

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

Step 2: Optimal Parameter Analysis

Define $\phi(t) = f(\mathbf{x}_k + \mathbf{d}(t))$ where $\mathbf{d}(t) = t(1-t)(-\nabla f(\mathbf{x}_k)) + t^2\mathbf{d}_{\text{L-BFGS}}$. The first derivative is:

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

The second derivative is:

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

$$+\nabla f(\mathbf{x}_k + \mathbf{d}(t))^T[-2(-\nabla f(\mathbf{x}_k)) + 2\mathbf{d}_{\text{L-BFGS}}]$$

At $t = 1$:

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

Using Taylor expansion:

$$\nabla f(\mathbf{x}_k + \mathbf{d}_{\text{L-BFGS}}) = \nabla f(\mathbf{x}_k) + \nabla^2 f(\mathbf{x}_k)\mathbf{d}_{\text{L-BFGS}} + O(\|\mathbf{d}_{\text{L-BFGS}}\|^2)$$

Since $\mathbf{d}_{\text{L-BFGS}} = -\mathbf{H}_k\nabla f(\mathbf{x}_k)$:

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

By the Dennis-Moré condition, as $k \to \infty$:

$$\|\mathbf{I} - \nabla^2 f(\mathbf{x}_k)\mathbf{H}_k\| \to 0$$

Therefore:

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

Step 3: Optimal Parameter Convergence

Since $\phi'(0) = -\|\nabla f(\mathbf{x}_k)\|^2 < 0$ and $\phi'(1) = o(\|\nabla f(\mathbf{x}_k)\|^2)$, by the intermediate value theorem and the fact that ϕ is strongly convex near $t = 1$ (due to positive definite Hessian), the minimizer satisfies:

$$t_k^* = 1 + o(1)$$

Step 4: Convergence Rate

With $t_k^* = 1 + o(1)$:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{d}(t_k^*) = \mathbf{x}_k + (1 + o(1))\mathbf{d}_{\text{L-BFGS}} + o(\|\mathbf{d}_{\text{L-BFGS}}\|)$$

$$= \mathbf{x}_k - \mathbf{H}_k\nabla f(\mathbf{x}_k) + o(\|\nabla f(\mathbf{x}_k)\|)$$

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. □

14.3 B.3 Robustness Analysis

14.3.1 B.3.1 Graceful Degradation

Theorem B.4 (Graceful Degradation): Let θ_k be the angle between $-\nabla f(\mathbf{x}_k)$ and $\mathbf{d}_{\text{L-BFGS}}$. If $\theta_k > \pi/2$ (obtuse angle), then the optimal parameter satisfies $t^* \in [0, 1/2]$, ensuring gradient-dominated steps.

Proof: When $\theta_k > \pi/2$, we have $\nabla f(\mathbf{x}_k)^T \mathbf{d}_{\text{L-BFGS}} > 0$.

The derivative of our objective along the path is:

$$\frac{d}{dt} f(\mathbf{x}_k + \mathbf{d}(t)) = \nabla f(\mathbf{x}_k + \mathbf{d}(t))^T \mathbf{d}'(t)$$

At $t = 1/2$:

$$\mathbf{d}'(1/2) = -\frac{1}{2}\nabla f(\mathbf{x}_k) + \mathbf{d}_{\text{L-BFGS}}$$

For small steps from $\mathbf{x}_k$:

$$\nabla f(\mathbf{x}_k + \mathbf{d}(1/2)) \approx \nabla f(\mathbf{x}_k)$$

Therefore:

$$\frac{d}{dt} f(\mathbf{x}_k + \mathbf{d}(t))\bigg|_{t=1/2} \approx \nabla f(\mathbf{x}_k)^T [-\frac{1}{2}\nabla f(\mathbf{x}_k) + \mathbf{d}_{\text{L-BFGS}}]$$

$$= -\frac{1}{2}\|\nabla f(\mathbf{x}_k)\|^2 + \nabla f(\mathbf{x}_k)^T \mathbf{d}_{\text{L-BFGS}} > 0$$

when $\nabla f(\mathbf{x}_k)^T \mathbf{d}_{\text{L-BFGS}} > \frac{1}{2}\|\nabla f(\mathbf{x}_k)\|^2$.

This implies the function increases beyond $t = 1/2$, so the univariate optimization will find $t^* \leq 1/2$, giving:

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

Since $t^* \leq 1/2$, we have $t^*(1 - t^*) \geq t^*(1/2)$, ensuring a gradient-dominated step. □

14.3.2 B.3.2 Stability Under Numerical Errors

Theorem B.5 (Numerical Stability): Let $\tilde{\mathbf{d}}_{\text{L-BFGS}} = \mathbf{d}_{\text{L-BFGS}} + \boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon}$ represents numerical errors with $\|\boldsymbol{\epsilon}\| \leq \delta$. The perturbed QQN path:

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

satisfies:

$$\|\tilde{\mathbf{d}}(t) - \mathbf{d}(t)\| \leq t^2 \delta$$

Proof: Direct computation:

$$\|\tilde{\mathbf{d}}(t) - \mathbf{d}(t)\| = \|t^2(\tilde{\mathbf{d}}_{\text{L-BFGS}} - \mathbf{d}_{\text{L-BFGS}})\| = t^2\|\boldsymbol{\epsilon}\| \leq t^2\delta$$

For small t (near the initial descent phase), the error is $O(t^2\delta)$, providing quadratic error suppression. □

14.4 B.4 Computational Complexity

Theorem B.6 (Computational Complexity): Each QQN iteration requires:

- $O(n)$ operations for path construction
- $O(mn)$ operations for L-BFGS direction computation
- $O(k)$ function evaluations for univariate optimization

where n is the dimension, m is the L-BFGS memory size, and k is typically small (3-10).

Proof:

1. **Path construction**: Computing $\mathbf{d}(t) = t(1-t)(-\nabla f)+t^2\mathbf{d}_{\text{L-BFGS}}$ requires $O(n)$ operations for vector arithmetic.
2. **L-BFGS direction**: The two-loop recursion requires $O(mn)$ operations to compute $\mathbf{H}_k\nabla f(\mathbf{x}_k)$.
3. **Line search**: Each function evaluation along the path requires $O(n)$ operations to compute $\mathbf{x}_k + \mathbf{d}(t)$, plus the cost of evaluating f.

Total complexity per iteration: $O(mn + kn) + k \cdot \text{cost}(f)$. □

14.5 B.5 Extensions and Variants

14.5.1 B.5.1 Gradient Scaling

The basic QQN formulation can be enhanced with gradient scaling:

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

where $\alpha > 0$ is a scaling factor.

Proposition B.1 (Scaling Invariance): The set of points reachable by the QQN path is invariant to the choice of α. Only the parametrization changes.

Proof: Consider the mapping $s = \beta(t)$ where β is chosen such that:

$$t(1-t)\alpha(-\nabla f) + t^2\mathbf{d}_{\text{L-BFGS}} = s(1-s)(-\nabla f) + s^2\mathbf{d}_{\text{L-BFGS}}$$

This gives a bijection between parametrizations, showing that any point reachable with one α is reachable with another. □

14.5.2 B.5.2 Cubic Extension with Momentum

Incorporating momentum leads to cubic interpolation:

$$\mathbf{d}(t) = t(1-t)(1-2t)\mathbf{m} + t(1-t)\alpha(-\nabla f) + t^2\mathbf{d}_{\text{L-BFGS}}$$

where $\mathbf{m}$ is the momentum vector.

This satisfies:

- $\mathbf{d}(0) = \mathbf{0}$
- $\mathbf{d}'(0) = \alpha(-\nabla f) + \mathbf{m}$
- $\mathbf{d}(1) = \mathbf{d}_{\text{L-BFGS}}$
- $\mathbf{d}''(0) = -6\mathbf{m} + 2\alpha(-\nabla f) + 2\mathbf{d}_{\text{L-BFGS}}$

Theorem B.7 (Cubic Convergence Properties): The cubic variant maintains all convergence guarantees of the quadratic version while potentially improving the convergence constant through momentum acceleration.

14.5.3 B.5.3 Trust Region Integration

QQN naturally extends to trust regions by constraining the univariate search:

$$t^* = \arg\min_{t:\|\mathbf{d}(t)\| \le \Delta} f(\mathbf{x} + \mathbf{d}(t))$$

where Δ is the trust region radius.

Proposition B.2 (Trust Region Feasibility): For any $\Delta > 0$, there exists $t_{\max} > 0$ such that $\|\mathbf{d}(t)\| \le \Delta$ for all $t \in [0, t_{\max}]$.

Proof: Since $\mathbf{d}(0) = \mathbf{0}$ and $\mathbf{d}$ is continuous, by the intermediate value theorem, the set $\{t : \|\mathbf{d}(t)\| \le \Delta\}$ contains an interval $[0, t_{\max}]$ for some $t_{\max} > 0$. □

14.6 B.6 Comparison with Related Methods

14.6.1 B.6.1 Relationship to Trust Region Methods

Trust region methods solve:

$$\min_{\mathbf{s}} \mathbf{g}^T\mathbf{s} + \frac{1}{2}\mathbf{s}^T\mathbf{B}\mathbf{s} \quad \text{s.t.} \quad \|\mathbf{s}\| \leq \Delta$$

QQN can be viewed as solving a related but different problem:

$$\min_{t \geq 0} f(\mathbf{x} + \mathbf{d}(t))$$

where $\mathbf{d}(t)$ is the quadratic path. **Key differences**:

- Trust region: Solves 2D subproblem, then line search
- QQN: Direct 1D optimization along quadratic path
- Trust region: Requires trust region radius management
- QQN: Parameter-free, automatic adaptation

14.6.2 B.6.2 Relationship to Line Search Methods

Traditional line search methods optimize:

$$\min_{\alpha > 0} f(\mathbf{x} + \alpha\mathbf{d})$$

QQN generalizes this by optimizing along a parametric path:

$$\min_{t \geq 0} f(\mathbf{x} + \mathbf{d}(t))$$

The key insight is that the direction itself changes with the parameter, providing additional flexibility.

14.6.3 B.6.3 Relationship to Hybrid Methods

Previous hybrid approaches typically use discrete switching:

$$\mathbf{d} = \begin{cases} \mathbf{d}_{\text{gradient}} & \text{if condition A} \\ \mathbf{d}_{\text{quasi-Newton}} & \text{if condition B} \end{cases}$$

QQN provides continuous interpolation, eliminating discontinuities and the need for switching logic.