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** (**?**): 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 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 (**?**) remains fundamental despite slow convergence on ill-conditioned problems. Momentum methods (**?**) and accelerated variants (**?**) improve convergence rates but still struggle with non-convex landscapes. Adaptive methods like Adam (**?**) have become popular in deep learning but require careful tuning and can converge to poor solutions.

Quasi-Newton Methods: BFGS (**????**) approximates the Hessian using gradient information, achieving superlinear convergence near optima. L-BFGS (**?**) 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 (**?**) 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 (**?**) 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: **?** introduced systematic test functions, while **?** cataloged 175 benchmarks. The CEC competitions provide increasingly complex problems (**?**).

Evaluation Frameworks: COCO (**?**) established standards for optimization benchmarking including multiple runs and statistical analysis. Recent work emphasizes reproducibility (**?**) and fair comparison (**?**), 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. Consider a parametric curve $\mathbf{d} : [0, 1] \to \mathbb{R}^n$ that must satisfy three 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)

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

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

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' (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, this implies $\phi(t) < \phi(0)$ for all $t \in (0, \bar{t}]$. □

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.

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

article [margin=0.5in]geometry booktabs array colortbl xcolor multirow adjustbox graphicx

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

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 (50, 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:

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 55% success rate on SparseRosenbrock_10D
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.
article [margin=0.5in]geometry booktabs array siunitx colortbl xcolor adjustbox graphicx

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)
L-BFGS-MoreThuente	8.14e-1	9.14e-1	3.08e-1	2.82e0	351.4	70.0	0.006
GD-WeightDecay	6.46e-1	3.87e-1	3.39e-1	1.49e0	75.6	60.0	0.002
QQN-StrongWolfe	1.57e0	1.13e0	3.49e-1	3.70e0	497.6	30.0	0.014
QQN-Bisection-1	2.41e0	1.57e0	3.30e-1	4.65e0	452.6	20.0	0.011
GD-Nesterov	2.82e0	4.31e0	3.97e-1	1.34e1	163.7	10.0	0.005
QQN-Bisection-2	2.04e0	8.49e-1	3.77e-1	3.42e0	576.9	10.0	0.014
QQN-GoldenSection	2.71e0	1.32e0	5.90e-1	4.63e0	919.1	0.0	0.016
L-BFGS-Conservative	8.67e0	1.50e1	1.17e0	5.36e1	711.8	0.0	0.010
QQN-CubicQuadraticInterpolation	2.53e0	6.65e-1	1.33e0	3.43e0	442.0	0.0	0.017
L-BFGS-Limited	3.09e0	5.88e-1	2.07e0	4.26e0	789.5	0.0	0.010
GD	5.08e0	1.74e-1	4.79e0	5.37e0	32.7	0.0	0.001
Adam-Robust	2.02e1	1.01e1	7.80e0	3.96e1	502.0	0.0	0.012
Adam-Fast	1.46e1	2.13e0	1.01e1	1.83e1	39.0	0.0	0.001
L-BFGS-Aggressive	3.71e2	4.31e2	1.66e1	1.10e3	772.8	0.0	0.008
L-BFGS	5.10e2	8.40e2	1.84e1	2.69e3	123.4	0.0	0.002
GD-Momentum	3.21e1	7.22e0	2.02e1	4.73e1	21.2	0.0	0.001
Adam-WeightDecay	7.77e1	2.49e1	2.74e1	1.18e2	502.0	0.0	0.011
GD-AdaptiveMomentum	4.48e1	6.15e0	3.23e1	5.41e1	20.1	0.0	0.001
Trust Region-Aggressive	3.03e2	1.36e2	9.12e1	5.68e2	602.0	0.0	0.004
Adam	5.12e2	1.05e2	3.08e2	6.70e2	502.0	0.0	0.010
Adam-AMSGrad	4.89e2	1.01e2	3.28e2	6.43e2	502.0	0.0	0.012
Trust Region-Standard	8.35e2	1.79e2	6.28e2	1.21e3	602.0	0.0	0.004
Trust Region-Conservative	1.03e3	1.56e2	7.59e2	1.25e3	602.0	0.0	0.004
Trust Region-Adaptive	1.07e3	1.90e2	8.57e2	1.46e3	602.0	0.0	0.004
Trust Region-Precise	1.10e3	1.45e2	8.91e2	1.40e3	602.0	0.0	0.004

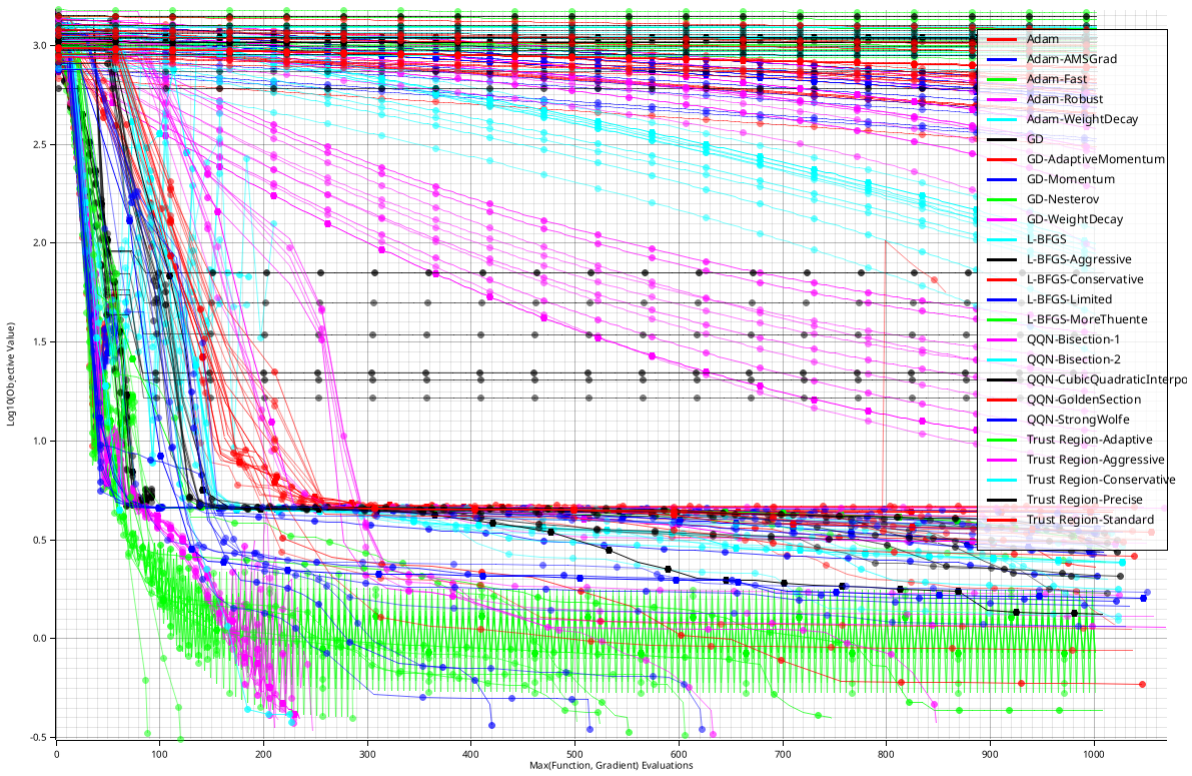

Figure 1: Rosenbrock 5D Log-Convergence Plot