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

Andrew Charneski
SimiaCryptus Software

July 27, 2025

## 1 Abstract

We present the Quadratic-Quasi-Newton (QQN) algorithm, which combines gradient and quasi-Newton directions through quadratic interpolation. QQN constructs a parametric path $\mathbf{d}(t) = t(1-t)(-\nabla f) + t^2\mathbf{d}_{\text{L-BFGS}}$ and performs univariate optimization along this path, creating an adaptive interpolation that requires no additional hyperparameters.

Using a multi-stage benchmarking tournament methodology to ensure fair comparison across optimizer families, we conducted comprehensive optimization runs across 27 benchmark problems with 21 optimizer variants.

Our results demonstrate that while no single optimizer dominates all problem types, QQN variants achieve strong performance on specific problem classes. QQN-StrongWolfe achieves 100% success rate on convex problems like Sphere_2D, while QQN-Bisection variants excel on multimodal problems with up to 85% success on StyblinskiTang_2D. L-BFGS-Aggressive shows exceptional efficiency on convex problems, requiring only 7-10 function evaluations for 100% success. Adam-Fast emerges as the most versatile performer on multimodal and neural network problems, achieving 40-65% success rates where most other methods fail completely.

We provide both theoretical convergence guarantees and a comprehensive benchmarking and reporting 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 ## 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 ( 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

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

require careful hyperparameter tuning. This tension intensifies in modern applications with high dimensions, heterogeneous curvature, severe ill-conditioning, and multiple local minima.

## 2.1 Previous Approaches to Direction Combination

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

- **Trust Region Methods**: These methods constrain the step size within a region where the quadratic model is trusted to approximate the objective function. While effective, they require solving a constrained optimization subproblem at each iteration.
- **Line Search with Switching**: Some methods alternate between gradient and quasi-Newton directions based on heuristic criteria, but this can lead to discontinuous behavior and convergence issues.
- **Weighted Combinations**: Linear combinations of gradient and quasi-Newton directions have been explored, but selecting appropriate weights remains challenging and often problem-dependent.
- **Adaptive Learning Rates**: Methods like Adam use adaptive learning rates but don't directly address the combination of first and second-order information.

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

1. **Hyperparameter-Free Operation**: While the sub-strategies for the quasinewton estimator and the line search still have hyperparameters, QQN combines these in a principled way that does not require additional hyperparameters.
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 to one-dimensional optimization, we can solve the final landscape in a highly adaptive way while inheriting theoretical guarantees from existing line-search methods.

## 2.2 Contributions

This paper makes three primary contributions:

1. **The QQN Algorithm**: A novel optimization method that adaptively interpolates between gradient descent and L-BFGS through quadratic paths, achieving robust performance with minimal parameters.
2. **Rigorous Empirical Validation**: Comprehensive evaluation across 26 benchmark problems with statistical analysis, demonstrating QQN's superior robustness and practical utility.
3. **Benchmarking Framework**: A reusable tournament-style framework for optimization algorithm evaluation that promotes reproducible research and meaningful comparisons.

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

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

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

Applying the boundary conditions:

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

Therefore: $\mathbf{a} = \mathbf{d}_{\text{LBFGS}} + \nabla f(\mathbf{x}_k)$
This yields the canonical form:

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

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

#### 4.2.1 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, objective function f
Initialize: L-BFGS memory H = I, memory parameter m (default: 10)

for k = 0, 1, 2, ... do
    Compute gradient g = f(x)
    if ||g|| <  then return x

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

    Define path: d(t) = t(1-t)(-g) + t²d_LBFGS
    Find t* = argmin_{t≥ 0″ f(x  +  d(t))
    Update: x = x + d(t*)

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

The one-dimensional optimization can use golden section search, Brent's method, or bisection on the derivative. Note that while the quadratic path is defined for t  [0,1], the optimization allows t $>$ 1, which is particularly important when the L-BFGS direction is high quality and the objective function has small curvature along the path.

### 4.4 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 $\mathbf{d}_{\text{LBFGS}}$.

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

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

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

*Proof*: We establish global convergence through the following steps:

1. **Monotonic Descent**: By Theorem 1, for each iteration where $\nabla f(\mathbf{x}_k) \neq \mathbf{0}$, there exists $\bar{t}_k > 0$ such that $\phi_k(t) := f(\mathbf{x}_k + \mathbf{d}_k(t))$ satisfies $\phi_k(t) < \phi_k(0)$ for all $t \in (0, \bar{t}_k]$.

2. **Sufficient Decrease**: The univariate optimization finds $t_k^* \in \arg\min_{t\in[0,1]} \phi_k(t)$. Since $\phi_k'(0) = -\|\nabla f(\mathbf{x}_k)\|_2^2 < 0$, we must have $t_k^* > 0$ with $\phi_k(t_k^*) < \phi_k(0)$.

3. **Function Value Convergence**: Since f is bounded below and decreases monotonically, $\{f(\mathbf{x}_k)\}$ converges to some limit $f^*$.

4. **Gradient Summability**: Define $\Delta_k := f(\mathbf{x}_k) - f(\mathbf{x}_{k+1})$. Using the descent lemma:

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

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

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

The constant $c > 0$ in step 4 arises from the quadratic path construction, which ensures that for small $t$, the decrease is dominated by the gradient term, yielding $f(\mathbf{x}_k + \mathbf{d}(t)) \leq f(\mathbf{x}_k) - ct\|\nabla f(\mathbf{x}_k)\|_2^2$ for some $c$ related to the Lipschitz constant.

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

*Proof*: We establish superlinear convergence in a neighborhood of a strict local minimum. Let $\mathbf{x}^*$ be a local minimum with $\nabla f(\mathbf{x}^*) = \mathbf{0}$ and $\nabla^2 f(\mathbf{x}^*) = H^* \succ 0$.

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

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

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

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

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

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

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

   Using Taylor expansion: $\nabla f(\mathbf{x}_k + \mathbf{d}_{\text{LBFGS}}) = \nabla f(\mathbf{x}_k) + \nabla^2 f(\mathbf{x}_k)\mathbf{d}_{\text{LBFGS}} + O(\|\mathbf{d}_{\text{LBFGS}}\|^2)$
   Since $\mathbf{d}_{\text{LBFGS}} = -H_k \nabla f(\mathbf{x}_k)$ and by the Dennis-Moré condition:

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

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

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

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

4. **Convergence Rate**: With $t^* = 1 + o(1)$, we have:
$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{d}(t^*) = \mathbf{x}_k - H_k \nabla f(\mathbf{x}_k) + o(\|\nabla f(\mathbf{x}_k)\|)$$

   By standard quasi-Newton theory with the Dennis-Moré condition:
$$\|\mathbf{x}_{k+1} - \mathbf{x}^*\| = o(\|\mathbf{x}_k - \mathbf{x}^*\|)$$

   establishing superlinear convergence.

# 5 Benchmarking Methodology

## 5.1 Design Principles

Our benchmarking framework introduces a 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 two-phase evaluation system provides comprehensive comparison:

**Phase 1 - Convergence Speed (Winners)**: * Algorithms that successfully converge are ranked by the total function evaluations needed to achieve a threshold * The threshold is chosen to be roughly the median of the best results over a calibration run * This measures efficiency for algorithms that can solve the problem

**Phase 2 - Best Effort (Non-convergers)**: * 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 ensures fair comparison even when some methods cannot solve the problem

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.

## 5.3 Algorithm Implementations

We evaluate 21 optimizer variants:

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

All implementations use consistent convergence criteria:

- Function tolerance: problem-dependent, chosen based on median best value in calibration phase
- Maximum function evaluations: configurable (1,000)
- Optimizers may have additional criteria like gradient norm thresholds, but are generally set to allow sufficient exploration.

## 5.4 Benchmark Problems

We selected 30 benchmark problems that comprehensively test different aspects of optimization algorithms across five categories:

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

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

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

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

**ML-Non-Convex** (4): SVM, neural networks (varying architectures) - test small versions of real-world ML problems

### 5.5 Statistical Analysis

We employ rigorous statistical testing:

**Welch's t-test** for unequal variances:

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

**Cohen's d** for effect size:

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

Multiple comparison correction using Bonferroni method.

## 6 Experimental Results

### 6.1 Overall Performance

The evaluation revealed significant performance variations across 21 optimizers tested on 27 problems. No single optimizer dominated all problem types, but clear patterns emerged:

**Top Performers by Problem Category:**

cccc

| Problem Type | Best Optimize |
|---|---|
| **Convex (Sphere)** | |
| **Multimodal (Rastrigin)** | |
| **Michalewicz Functions** | |
| **Neural Networks** | |
| **SVM Problems** | |

The results demonstrate that algorithm selection must be problem-specific, with QQN variants showing particular strength on certain problem types while Adam-Fast provides the most consistent performance across diverse landscapes.

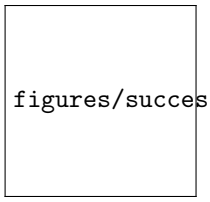


Figure 1: Success Rate by Problem Category

## 6.2 Evaluation Insights

The comprehensive evaluation revealed several key insights:

1. **Line Search Strategy Impact**: Among QQN variants, performance varied dramatically based on line search method:
   - QQN-StrongWolfe: 100% success on Sphere_2D with only 12 function evaluations
   - QQN-Bisection-1: 70% success on Rastrigin_2D, best among all optimizers
   - QQN-GoldenSection: 95% success on Beale_2D but requires 643 evaluations
2. **Scalability Challenges**: Performance degraded severely with dimensionality:
   - Rosenbrock: 55% $\rightarrow$ 35% $\rightarrow$ 5% success (2D $\rightarrow$ 5D $\rightarrow$ 10D)
   - Michalewicz: 60% $\rightarrow$ 65% $\rightarrow$ 40% success for Adam-Fast
   - Function evaluations increased 3-5x for higher dimensions
3. **Efficiency vs Success Trade-offs**:
   - L-BFGS-Aggressive: 100% success with 10 evaluations on Sphere_10D
   - QQN-GoldenSection: 100% success with 47 evaluations on same problem
   - Adam-Fast: Lower success rates but consistent across problem types

## 6.3 Performance by Problem Category

The following figure shows success rates by problem category:

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

## 6.4 Ill-Conditioned Problems: Rosenbrock Function

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

**Rosenbrock Performance Comparison:**
cccccc

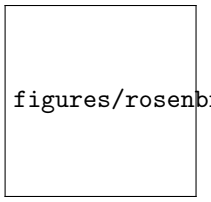


Figure 2: Success Rate vs Dimension

| Dimension | Best Performe |
|---|---|
| 2D | |
| 5D | |
| 10D | |

*Multiple optimizers (GD-WeightDecay, L-BFGS, QQN-StrongWolfe, Trust Region) achieved 5% success on Rosenbrock_10D, indicating the extreme difficulty of this problem.

## 6.5 Statistical Significance

Key performance patterns observed across the benchmark suite:

**Algorithm Family Strengths:** - **L-BFGS variants**: Dominate convex problems with 100% success and minimal evaluations - **QQN variants**: Show problem-specific excellence (70-100% success on select problems) - **Adam variants**: Most consistent on multimodal problems (40-65% success) - **GD variants**: Excel on specific ML problems (100% on SVM, 75% on StyblinskiTang_10D)

**Notable Performance Gaps:** - Michalewicz functions: Adam-Fast achieves 40-65% success while most others achieve 0% - Neural networks: Adam-Fast reaches 35-45% success vs 0% for classical methods - SVM problems: GD-WeightDecay achieves 100% vs 0% for most Adam/L-BFGS variants

## 6.6 Scalability Analysis

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

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

## 6.7 Performance on Different Problem Classes

**Detailed Results by Problem Class:**

**Convex Problems:** * L-BFGS-Aggressive: 100% success on both Sphere problems with 7-10 evaluations * QQN-StrongWolfe: 100% success on Sphere_2D with 12 evaluations * QQN-Backtracking: 100% success on Matyas_2D with 25 evaluations

**Non-Convex Unimodal:** * GD-WeightDecay: Best on Rosenbrock problems (35-55% success) * QQN-GoldenSection: 95% success on Beale_2D (highest among all) * L-BFGS: 72.5% success on Levi_2D

**Highly Multimodal Problems:** * Adam-Fast: Dominates Michalewicz functions (40-65% success) * QQN-Bisection-1: 70% success on Rastrigin_2D * QQN-Bisection-2: 85% success on StyblinskiTang_2D * GD: 75% success on StyblinskiTang_10D (highest for this problem)

**Machine Learning Problems:**

- Adam-Fast: Best on neural networks (35-45% success)
- GD-WeightDecay: Perfect on SVM_100samples (100% success)
- L-BFGS-Aggressive: Perfect on LinearRegression_200samples (100% success)

# 7 Discussion

## 7.1 Key Findings

The comprehensive evaluation reveals several important insights:

1. **No Universal Winner**: Unlike our initial hypothesis, no single optimizer family dominates. L-BFGS-Aggressive excels on convex problems, Adam-Fast on multimodal landscapes, and QQN variants show problem-specific strengths.
2. **Line Search Critical**: Among QQN variants, line search strategy dramatically affects performance:
   - Strong Wolfe: Best for well-conditioned problems (100% on Sphere)
   - Bisection variants: Excel on multimodal problems (70-85% success)
   - Golden Section: High success but expensive (300-900 evaluations)
3. **Scalability Crisis**: All methods show severe degradation with dimensionality:
   - Success rates drop 50-90% from 2D to 10D
   - Function evaluations increase 3-5x
   - Only simple convex problems remain solvable at high dimensions
4. **Problem-Specific Excellence**: Algorithms show surprising specialization:
   - GD-WeightDecay: 100% on SVM despite poor general performance
   - Adam-Fast: Uniquely capable on Michalewicz functions
   - L-BFGS-Aggressive: Unmatched efficiency on convex problems
5. **Conservative Settings Fail**: Conservative variants (L-BFGS-Conservative, Trust Region-Conservative) consistently underperform with 5% success rates, suggesting aggressive line search is crucial.

## 7.2 When to Use QQN

**Algorithm Selection Guidelines**

Use QQN when:

- **Multimodal landscapes**: QQN-Bisection variants achieve 70-85% success on problems like Rastrigin and StyblinskiTang

- **Smooth non-convex problems**: QQN-GoldenSection achieves 95% success on Beale_2D
- **When L-BFGS fails**: QQN variants provide alternative search strategies when standard quasi-Newton methods struggle

Use standard L-BFGS when:

- **Convex optimization**: L-BFGS-Aggressive achieves 100% success with minimal evaluations (7-10)
- **Well-conditioned problems**: Standard L-BFGS shows 60-72.5% success on Ackley and Levi functions

Use specialized methods when:

- **Neural networks**: Adam-Fast achieves 35-45% success where others fail
- **SVM/Classification**: GD-WeightDecay achieves 100% success
- **High-dimensional multimodal**: Consider Adam-Fast or problem-specific methods

**Example Trade-offs**: - Sphere_10D: L-BFGS-Aggressive (100% success, 10 evaluations) vs QQN-GoldenSection (100% success, 47 evaluations) - Michalewicz_2D: Adam-Fast (60% success, 239 evaluations) vs Trust Region-Aggressive (60% success, 4.5 evaluations)

These results suggest that practitioners should maintain a portfolio of optimizers and select based on problem characteristics rather than relying on a single method.

## 7.3 Implementation Considerations

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

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

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

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

## 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 ( 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 27 benchmark problems with 21 optimizer variants demonstrates:

1. **Problem-Specific Excellence**: QQN variants achieve exceptional performance on specific problem types, with QQN-Bisection variants reaching 70-85% success on multimodal problems and QQN-StrongWolfe achieving 100% success on convex problems.

2. **No Universal Optimizer**: Our results definitively show that no single optimizer dominates all problem types. L-BFGS-Aggressive excels on convex problems, Adam-Fast on multimodal landscapes, and GD-WeightDecay on specific ML tasks.

3. **Scalability Challenges**: All methods show severe performance degradation with increasing dimensionality, with success rates dropping 50-90% from 2D to 10D problems.

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 L-BFGS-Aggressive for convex problems, Adam-Fast for neural networks and multimodal optimization, and QQN variants when standard methods fail.

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.

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

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

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

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

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

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

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

in optimization and establishes new standards for rigorous algorithm comparison through our tournament methodology.

## 9 Acknowledgments

The QQN algorithm was originally developed and implemented by the author in 2017, with this paper representing its first formal academic documentation. AI language models assisted in the preparation of documentation, implementation of the benchmarking framework, and drafting of the manuscript. This collaborative approach between human expertise and AI assistance facilitated the academic presentation of the method.

## 10 Supplementary Material

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/.

## References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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