Theoretical Analysis of the Quadratic-Quasi-Newton Algorithm: Convergence Properties and Geometric Insights

Andrew Charneski
SimiaCryptus Software

August 2, 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. Our key contributions are: (1) a parameter-free framework for combining optimization directions that guarantees descent, (2) global convergence under standard assumptions with explicit convergence rates, (3) local superlinear convergence matching quasi-Newton methods, and (4) automatic graceful degradation to gradient descent when quasi-Newton approximations fail. The algorithm requires no hyperparameters beyond those of its constituent methods and matches the computational complexity of L-BFGS while providing superior robustness.

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

2 Introduction

Optimization lies at the heart of modern computational science, from training neural networks to solving inverse problems in physics. Despite decades of research, practitioners still face a fundamental dilemma: gradient descent methods are robust but slow, while quasi-Newton methods are fast but fragile. This paper resolves this dilemma through a novel geometric framework.

Consider the following motivating example: when training a deep neural network, gradient descent with momentum might take thousands of iterations to converge, while L-BFGS could converge in hundreds—but might also diverge catastrophically if the Hessian approximation becomes poor. Current solutions involve complex heuristics, careful hyperparameter tuning, and frequent manual intervention.

We present the Quadratic-Quasi-Newton (QQN) algorithm, which automatically combines the robustness of gradient descent with the efficiency of quasi-Newton methods through a principled geometric framework. Our approach requires no hyperparameters and provides theoretical guarantees that match or exceed both constituent methods.

2.1 The Direction Combination Problem

Consider the fundamental question in optimization: given multiple directional advisors, how should we combine their recommendations? This problem arises naturally when we have:

- **Gradient direction**: $-\nabla f(\mathbf{x})$ providing guaranteed descent
- **Quasi-Newton direction**: $\mathbf{d}_{\text{QN}}$ offering potential superlinear convergence
- **Trust and uncertainty**: The quasi-Newton direction may be unreliable

Traditional approaches include:

- **Trust region methods** [Conn et al., 2000]: Constrain steps within regions where quadratic models are trusted
- **Line search switching** [Morales and Nocedal, 2000]: Alternate between methods based on heuristics
- **Linear combinations** [Biggs, 1973]: Weighted averages of directions

We propose a geometric solution: construct a smooth parametric path that naturally interpolates between directions while guaranteeing descent properties.

3 The QQN Algorithm

3.1 Geometric Motivation

The key insight is to formulate direction combination as a boundary value problem in parametric space. We seek a 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: $\mathbf{a} = \mathbf{d}_{\text{L-BFGS}} + \nabla f(\mathbf{x})$

This yields the canonical QQN path:

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

3.2 Algorithm Specification

Algorithm 1: Quadratic-Quasi-Newton (QQN)

```
Input: Initial point x, objective function f, tolerance  > 0
Initialize: L-BFGS memory H = I, k = 0
while ||f(x)|| >  do
    Compute g = f(x)
    if k = 0 then
        d_LBFGS = -g
    else
        d_LBFGS = -Hg  // Two-loop recursion
    end if
    Define path: d(t) = t(1-t)(-g) + t²d_LBFGS
    Find t* = argmin_{t[0,2]} f(x + d(t))  // Allow t > 1
    x = x + d(t*)
    s = x - x
    y = f(x) - f(x)
    if sy > 0 then  // Curvature condition
        Update L-BFGS memory with (s, y)
    end if
    k = k + 1
end while
return x
```

3.3 Theoretical Properties

3.3.1 Universal Descent Property

Lemma 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 $t = 0$: $\mathbf{d}'(0) = -\nabla f(\mathbf{x})$. □ This property ensures descent regardless of the quality of $\mathbf{d}_{\text{L-BFGS}}$.

Theorem 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', there exists $\bar{t} > 0$ such that $\phi'(t) < 0$ for $t \in (0, \bar{t}]$. □

3.3.2 Global Convergence Analysis

Theorem 2 (Global Convergence): Under standard assumptions:

1. $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable
2. f is bounded below: $f(\mathbf{x}) \geq f_{\text{inf}} > -\infty$
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 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 $\bar{t} > 0$ 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 $t \in (0, \bar{t}]$ and some $c_1 \in (0, 1)$. The univariate optimization ensures $t_k^* \geq \min\{\bar{t}, 1\}$, giving:

$$f(\mathbf{x}_{k+1}) \leq f(\mathbf{x}_k) - c_1 \min\{\bar{t}, 1\}\|\nabla f(\mathbf{x}_k)\|^2$$

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, we can show there exists $c > 0$ such that:

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

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

3.3.3 Local Superlinear Convergence

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

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

$$\lim_{k\to\infty} \frac{\|(\mathbf{H}_k - (\nabla^2 f(\mathbf{x}^*))^{-1})(\mathbf{x}_{k+1} - \mathbf{x}_k)\|}{\|\mathbf{x}_{k+1} - \mathbf{x}_k\|} = 0$$

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

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

Using Taylor expansion and the Dennis-Moré condition, we can show:

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

This implies $t^* = 1 + o(1)$ for sufficiently large k.
Step 3: Convergence Rate
With $t^* = 1 + o(1)$:

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

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. □

3.4 Robustness Analysis

3.4.1 Graceful Degradation

Theorem 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 $t = 1/2$:

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

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

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

This is equivalent to gradient descent with step size t^*. □

3.4.2 Computational Complexity

Theorem 5 (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). The total complexity per iteration is $O(mn + kn)$, matching L-BFGS when function evaluation dominates.

4 Extensions and Variants

4.1 Gradient Scaling

The basic QQN formulation can be enhanced with gradient scaling to balance the relative magnitudes of the two directions:

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

where $\alpha > 0$ is a scaling factor. Three natural choices emerge:

1. **Unit scaling**: $\alpha = 1$ (default)
2. **Magnitude equalization**: $\alpha = \|\mathbf{d}_{\text{L-BFGS}}\|/\|\nabla f\|$
3. **Adaptive scaling**: α based on problem characteristics

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

Proof: The path $\{\mathbf{x} + \mathbf{d}(t) : t \in [0, 2]\}$ traces the same curve in $\mathbb{R}^n$ regardless of α, as any point on one parametrization can be reached by adjusting t in another. □

4.2 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 preserves all boundary conditions while adding curvature control through the second derivative at $t = 0$.

Theorem 5 (Cubic Convergence Properties): The cubic variant maintains all convergence guarantees of the quadratic version while potentially improving the convergence constant through momentum acceleration.

4.3 Trust Region Integration

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

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

where Δ is the trust region radius.

5 Comparison with Related Methods

5.1 Relationship to Trust Region Methods

Trust region methods solve:

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

This requires solving a constrained quadratic program at each iteration. QQN instead parameterizes a specific path and optimizes along it, avoiding the subproblem complexity while maintaining similar robustness properties.

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

5.2 Relationship to Line Search Methods

Traditional line search methods optimize along a fixed direction:

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

QQN generalizes this by optimizing along a parametric path that adapts its direction based on the parameter value.

5.3 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.

6 Practical Considerations

6.1 Line Search Implementation

The univariate optimization can use various methods:

- **Golden section search**: Robust, no derivatives needed
- **Brent's method**: Faster convergence with parabolic interpolation
- **Bisection on derivative**: When gradient information is available

Implementation Note: We recommend Brent's method with fallback to golden section search. The search interval $[0, 2]$ allows for extrapolation beyond the L-BFGS direction when beneficial.

6.2 Convergence Criteria

We recommend a combined stopping criterion:

$$\|\nabla f(\mathbf{x}_k)\| < \epsilon_{\text{abs}} \quad \text{or} \quad \|\nabla f(\mathbf{x}_k)\| < \epsilon_{\text{rel}} \|\nabla f(\mathbf{x}_0)\|$$

with typical values $\epsilon_{\text{abs}} = 10^{-8}$ and $\epsilon_{\text{rel}} = 10^{-6}$.

6.3 Memory Management

QQN inherits L-BFGS memory requirements:

- Store m vector pairs $(\mathbf{s}_i, \mathbf{y}_i)$
- Typical choice: $m = 5 - 10$
- Memory usage: $O(mn)$

6.4 Numerical Stability

Key stability considerations:

1. **Gradient scaling**: Prevents numerical issues when $\|\nabla f\| \ll \|\mathbf{d}_{\text{L-BFGS}}\|$
2. **Path parameterization**: The quadratic form is numerically stable
3. **Fallback behavior**: Automatic degradation to gradient descent

7 Conclusion

The Quadratic-Quasi-Newton algorithm resolves the robustness-efficiency trade-off in optimization through a novel geometric framework. Our theoretical analysis establishes:

1. **Universal descent property**: Guaranteed descent regardless of quasi-Newton quality
2. **Global convergence**: Under standard assumptions with explicit convergence rates
3. **Local superlinear convergence**: Matching quasi-Newton methods near optima
4. **Graceful degradation**: Automatic fallback to gradient descent when needed
5. **Computational efficiency**: Complexity matching L-BFGS with improved robustness

The geometric insight of quadratic interpolation provides a natural framework for direction combination that maintains theoretical guarantees while offering practical advantages. The method's parameter-free nature and robust behavior make it particularly suitable for practitioners who need reliable optimization without extensive tuning.

Future work includes:

- Extension to stochastic settings with mini-batch gradients
- Application to constrained optimization problems
- Integration with adaptive learning rate methods
- Theoretical analysis of the cubic variant with momentum

The quadratic interpolation principle opens new avenues for geometric approaches to optimization algorithm design, potentially leading to a new class of hybrid methods that combine the best properties of different optimization paradigms.

References

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.

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

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.

8 Appendix

8.1 A. Detailed Proofs

8.1.1 A.1 Proof of Sufficient Decrease Constant

Lemma A.1 (Sufficient Decrease): Under the assumptions of Theorem 2, there exists a constant $c > 0$ independent of k such that:

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

Proof: Consider the quadratic path $\mathbf{d}(t) = t(1-t)(-\nabla f) + t^2\mathbf{d}_{\text{L-BFGS}}$.
For small t, Taylor expansion gives:

$$\mathbf{d}(t) = -t\nabla f + O(t^2)$$

Using the descent lemma:

$$f(\mathbf{x}_k + \mathbf{d}(t)) \leq f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^T\mathbf{d}(t) + \frac{L}{2}\|\mathbf{d}(t)\|^2$$

Substituting the path:

$$f(\mathbf{x}_k + \mathbf{d}(t)) \leq f(\mathbf{x}_k) - t\|\nabla f(\mathbf{x}_k)\|^2 + \frac{Lt^2}{2}\|\nabla f(\mathbf{x}_k)\|^2 + O(t^3)$$

For $t = \min\{1, 1/L\}$, we get:

$$f(\mathbf{x}_k + \mathbf{d}(t)) \leq f(\mathbf{x}_k) - \frac{t}{2}\|\nabla f(\mathbf{x}_k)\|^2$$

Since the univariate optimization finds t^* at least as good as this choice:

$$f(\mathbf{x}_k) - f(\mathbf{x}_{k+1}) \geq \frac{\min\{1, 1/L\}}{2}\|\nabla f(\mathbf{x}_k)\|^2$$

Taking $c = \frac{\min\{1,1/L\}}{2}$ completes the proof. □

8.1.2 A.2 Dennis-Moré Condition Analysis

Lemma A.2 (Dennis-Moré Implies Unit Steps): If the L-BFGS approximation satisfies the Dennis-Moré condition, then $t^* \to 1$ as $k \to \infty$.

Proof: The Dennis-Moré condition states:

$$\lim_{k\to\infty} \frac{\|(\mathbf{H}_k - (\nabla^2 f(\mathbf{x}^*))^{-1})(\mathbf{x}_{k+1} - \mathbf{x}_k)\|}{\|\mathbf{x}_{k+1} - \mathbf{x}_k\|} = 0$$

Near the optimum, the L-BFGS direction becomes:

$$\mathbf{d}_{\text{L-BFGS}} = -\mathbf{H}_k\nabla f(\mathbf{x}_k) \approx -(\nabla^2 f(\mathbf{x}^*))^{-1}\nabla f(\mathbf{x}_k)$$

The optimal step in Newton's method would be:

$$\mathbf{s}_{\text{Newton}} = -(\nabla^2 f(\mathbf{x}_k))^{-1}\nabla f(\mathbf{x}_k) \approx -(\nabla^2 f(\mathbf{x}^*))^{-1}\nabla f(\mathbf{x}_k)$$

At $t = 1$, the QQN path gives:

$$\mathbf{d}(1) = \mathbf{d}_{\text{L-BFGS}} \approx \mathbf{s}_{\text{Newton}}$$

By the optimality of Newton's method near the minimum and continuity of the objective function, the univariate optimization will find $t^* \to 1$. □

8.2 B. Implementation Details

8.2.1 B.1 L-BFGS Direction Computation

The L-BFGS direction is computed using the two-loop recursion:

```
function compute_lbfgs_direction(gradient, memory):
    q = gradient
    alphas = []
    // First loop (backward)
    for i = memory.size-1 down to 0:
        rho_i = 1 / (memory.y[i]^T * memory.s[i])
        alpha_i = rho_i * memory.s[i]^T * q
        q = q - alpha_i * memory.y[i]
        alphas.append(alpha_i)
    // Apply initial Hessian approximation
    if memory.size > 0:
        gamma = (memory.s[-1]^T * memory.y[-1]) / (memory.y[-1]^T * memory.y[-1])
        r = gamma * q
    else:
        r = q
    // Second loop (forward)
    for i = 0 to memory.size-1:
        rho_i = 1 / (memory.y[i]^T * memory.s[i])
        beta = rho_i * memory.y[i]^T * r
        r = r + (alphas[memory.size-1-i] - beta) * memory.s[i]
    return -r
```

8.2.2 B.2 Univariate Optimization Methods

Golden Section Search

```
function golden_section_search(f, a, b, tol):
    phi = (1 + sqrt(5)) / 2
    resphi = 2 - phi
    x1 = a + resphi * (b - a)
    x2 = b - resphi * (b - a)
    f1 = f(x1)
    f2 = f(x2)
    while abs(b - a) > tol:
        if f1 > f2:
            a = x1
            x1 = x2
            f1 = f2
            x2 = b - resphi * (b - a)
            f2 = f(x2)
        else:
            b = x2
            x2 = x1
            f2 = f1
            x1 = a + resphi * (b - a)
            f1 = f(x1)
    return (a + b) / 2
```

Brent's Method Combines golden section search with parabolic interpolation for faster convergence when the function is smooth.

8.2.3 B.3 Memory Update

After each iteration, update the L-BFGS memory:

```
function update_memory(memory, s_k, y_k):
    if memory.size == memory.max_size:
        // Remove oldest pair
        memory.s.pop_front()
        memory.y.pop_front()
    // Add new pair
    memory.s.push_back(s_k)
    memory.y.push_back(y_k)
    memory.size = min(memory.size + 1, memory.max_size)
    // Check curvature condition
    if s_k^T * y_k <= 0:
        // Skip update or apply damping
        memory.s.pop_back()
        memory.y.pop_back()
        memory.size -= 1
```

8.3 C. Convergence Rate Analysis

8.3.1 C.1 Linear Convergence Rate

For strongly convex functions with condition number κ, QQN achieves at least linear convergence with rate:

$$\|\mathbf{x}_{k+1} - \mathbf{x}^*\| \leq \left(1 - \frac{1}{\kappa}\right) \|\mathbf{x}_k - \mathbf{x}^*\|$$

This follows from the fact that QQN reduces to gradient descent in the worst case, and gradient descent achieves this rate on strongly convex functions.

8.3.2 C.2 Superlinear Convergence Rate

Near the optimum, when L-BFGS provides good approximations, QQN achieves superlinear convergence:

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

The exact rate depends on how quickly the L-BFGS approximation converges to the true inverse Hessian.

8.4 D. Numerical Examples

8.4.1 D.1 Quadratic Function

Consider $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x}$ where $\mathbf{A}$ is positive definite.

The gradient is $\nabla f(\mathbf{x}) = \mathbf{A}\mathbf{x}$ and the optimal step is $\mathbf{x}^* = \mathbf{0}$. For this function, the L-BFGS direction (after sufficient iterations) becomes:

$$\mathbf{d}_{\text{L-BFGS}} = -\mathbf{A}^{-1}\mathbf{A}\mathbf{x} = -\mathbf{x}$$

The QQN path becomes:

$$\mathbf{d}(t) = t(1-t)(-\mathbf{A}\mathbf{x}) + t^2(-\mathbf{x}) = -t\mathbf{x}[(1-t)\mathbf{A} + t\mathbf{I}]$$

For a quadratic function with exact L-BFGS approximation, the univariate optimization yields $t^* = 1$, giving the exact Newton step and convergence in one iteration.

8.4.2 D.2 Rosenbrock Function

The Rosenbrock function $f(x, y) = 100(y - x^2)^2 + (1 - x)^2$ is a classic test case for optimization algorithms.

The gradient is:

$$\nabla f = \begin{pmatrix} -400x(y - x^2) - 2(1 - x) \\ 200(y - x^2) \end{pmatrix}$$

Near the optimum $(1, 1)$, the Hessian is:

$$\nabla^2 f = \begin{pmatrix} 802 & -400 \\ -400 & 200 \end{pmatrix}$$

This is ill-conditioned with condition number $\kappa \approx 2416$, making it challenging for first-order methods but suitable for demonstrating QQN's robustness.

8.5 E. Extensions and Variations

8.5.1 E.1 Constrained QQN

For box constraints $\mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$, we can modify the univariate optimization:

$$t^* = \arg\min_{t \geq 0} f(\text{proj}(\mathbf{x} + \mathbf{d}(t)))$$

where proj is the projection onto the feasible region.

8.5.2 E.2 Stochastic QQN

For stochastic optimization, we can use:

1. **Mini-batch gradients**: Compute ∇f on subsets of data
2. **Variance reduction**: Use techniques like SVRG or SAGA
3. **Adaptive sampling**: Increase batch size as optimization progresses

8.5.3 E.3 Preconditioning

The gradient can be preconditioned:

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

where $\mathbf{P}$ is a preconditioning matrix (e.g., diagonal scaling).

8.6 F. Computational Complexity Analysis

8.6.1 F.1 Per-Iteration Cost

The computational cost per iteration consists of:

1. **Gradient computation**: $O(n)$ to $O(n^2)$ depending on the function
2. **L-BFGS direction**: $O(mn)$ where m is memory size
3. **Path evaluation**: $O(n)$ per function evaluation
4. **Univariate optimization**: $O(k)$ function evaluations, typically $k = 3 - 10$

Total: $O(mn + kn)$ operations plus k function evaluations, where the function evaluation cost typically dominates.

8.6.2 F.2 Memory Requirements

8.6.3 F.3 Comparison with Other Methods

Method	Per-iteration ops	Memory	Function evals	Robustness
Gradient Descent	$O(n)$	$O(n)$	1-5	High
L-BFGS	$O(mn)$	$O(mn)$	3-20	Medium
QQN	$O(mn)$	$O(mn)$	3-10	High
Newton	$O(n^3)$	$O(n^2)$	1	Low
Trust Region	$O(n^3)$	$O(n^2)$	1-10	High

QQN matches L-BFGS complexity while providing gradient descent robustness and often requiring fewer function evaluations due to better step selection.