

Improved Interval Interpolation Method Based on the 0.618 Method

Authors: Zong Yuhan, Zong Yuhan

Date: 2022-03-22T13:42:00+00:00

Abstract

The 0.618 method is the most widely used approach in one-dimensional line search for unimodal functions. While it exhibits good convergence properties, its convergence rate is excessively slow. Therefore, based on the function values at the endpoints of the search interval and at any point within the interval, this paper presents a universal acceleration strategy for line search, wherein each iteration can substantially reduce the interval of uncertainty for the function values. Numerical experimental results demonstrate that the convergence rate is improved compared to the 0.618 method, particularly when the function values at the endpoints of the initial interval differ substantially, in which case the proposed improved algorithm can significantly reduce the interval size.

Full Text

An Improved Interval Interpolation Method Based on the 0.618 Method

****ZONG Yu-han*, ZHANG Yan-bo****

School of Science, Beijing University of Posts and Telecommunications, Beijing 100876, China

Abstract

The 0.618 method is the most widely used approach for one-dimensional unimodal functions in line search. While it exhibits good convergence properties, its convergence rate is too slow. This paper proposes a universal acceleration strategy for line search based on function values at the interval endpoints and an arbitrary interior point. Each iteration substantially reduces the uncertainty interval containing the minimum. Numerical experiments demonstrate that the

proposed algorithm achieves faster convergence than the 0.618 method, particularly when the function values at the initial interval endpoints differ significantly.

Keywords: One-dimensional search; Interval interpolation; 0.618 method; Acceleration strategy

1. Introduction

In iterative optimization algorithms, one-dimensional search (or line search) is an indispensable component of many nonlinear programming methods. This problem can be reduced to the minimization of a univariate function, with the mathematical model being: minimize $f(x)$. Line search is classified into exact and inexact methods [?]-[?]. Exact line search includes both analytical and direct methods. Analytical methods utilize derivative information, such as interpolation methods, Newton's method, and parabolic methods. However, when derivatives are unavailable or difficult to compute, direct methods that do not require derivatives become necessary. The fundamental idea of direct methods is to first determine a search interval containing the optimal solution, then apply interpolation or partitioning techniques to narrow this interval for precise solution. Such methods include the golden section method, Fibonacci method, and quadratic interpolation. In one-dimensional search, functions are generally required to be unimodal, so this paper considers only unimodal functions.

Since most practical functions have unavailable or extremely difficult-to-compute derivatives, direct methods find wide application. From the aforementioned philosophy of direct methods, two key factors determine their performance: first, the selection of the initial interval, and second, the reduction of the subinterval containing the minimum point [?].

The 0.618 method is the most widely used approach in one-dimensional direct search. Its advantages include no requirement for function continuity, computation of only one new test point per iteration (with the other being reused), and good linear convergence properties. However, its drawback is slow convergence. This paper proposes an improved optimization method based on the 0.618 method and demonstrates through numerical comparisons that the new algorithm converges faster with fewer iterations.

2. The 0.618 Method

The 0.618 method, also known as the golden section method, operates on the principle of progressively narrowing the search interval (uncertainty interval) containing the minimum point by comparing function values at test points until some criterion is satisfied, yielding an approximation of the minimum [?].

The rule for selecting test points in the 0.618 method is:

$$\lambda = a + 0.382(b - a), \quad \mu = a + 0.618(b - a)$$

The computational procedure is as follows:

Step 0: Given initial interval $[a, b]$ and precision $\varepsilon > 0$. Calculate function values $f(a)$ and $f(b)$. Set $k = 0$.

Step 1: Compute test points λ_k and μ_k using the formulas above. Calculate function values $f(\lambda_k)$ and $f(\mu_k)$.

Step 2: If $f(\lambda_k) < f(\mu_k)$, set $b = \mu_k$; otherwise, set $a = \lambda_k$.

Step 3: If $|b - a| < \varepsilon$, stop and obtain the approximate minimum point; otherwise, set $k = k + 1$ and return to Step 1.

3. Improved Algorithm Based on Interval Interpolation

References [?][?] propose different improved algorithms based on interval reduction, utilizing first-order and second-order derivatives at one endpoint to further shrink the interval. However, these methods become inapplicable when derivatives are unavailable or difficult to compute. Therefore, this paper aims to develop a one-dimensional line search acceleration algorithm using function values at both interval endpoints and an arbitrary interior point to improve the 0.618 method.

Let $f(x)$ be a one-dimensional continuous unimodal function whose derivatives are unavailable or difficult to obtain. Let $[a_k, b_k]$ be the current search interval containing the minimum point x^* . The key insight is that if we can identify a point ξ in the interval where $f(\xi) > \max\{f(a_k), f(b_k)\}$, then the interval $[a_k, \xi]$ (or $[\xi, b_k]$) cannot contain x^* and can be eliminated before further partitioning, thereby enhancing algorithm efficiency. The challenge lies in finding such a point ξ .

For simple functions, an exact solution for ξ may be obtainable, but this approach has limited applicability. An alternative is to use interpolation, which relies solely on function values without requiring derivatives. We propose using quadratic interpolation to obtain an approximate value ζ that can replace ξ . The specific approach constructs a quadratic approximation $\phi(x)$ using function values at the interval endpoints $f(a_k)$, $f(b_k)$ and at the midpoint $f(c_k)$ (where c_k is typically chosen as the interval midpoint), employing three-point quadratic interpolation [?]. The point ζ satisfying $\phi(\zeta) = \max\{\phi(a_k), \phi(b_k)\}$ is then used for interval reduction.

The algorithmic steps of the improved method are:

Step 0: Given initial search interval $[a_0, b_0]$, precision $\varepsilon > 0$, and an interior point $c_0 \in (a_0, b_0)$ (typically the midpoint). Calculate function values $f(a_0)$, $f(b_0)$, and $f(c_0)$. Set $k = 0$.

Step 1: If $b_k - a_k < \varepsilon$, stop and return the approximate minimum point $x^* \approx (a_k + b_k)/2$. Otherwise, proceed to Step 2.

Step 2: If $f(a_k) > f(b_k)$, compute ζ_k via quadratic interpolation using points $(a_k, f(a_k))$, $(b_k, f(b_k))$, and $(c_k, f(c_k))$. Eliminate interval $[b_k, \zeta_k]$, set $a_{k+1} = a_k$, $b_{k+1} = \zeta_k$, and proceed to Step 4. Otherwise, proceed to Step 3.

Step 3: When $f(a_k) \leq f(b_k)$, compute ζ_k using the same interpolation approach. Eliminate interval $[\zeta_k, a_k]$, set $a_{k+1} = \zeta_k$, $b_{k+1} = b_k$, and proceed to Step 4.

Step 4: Compute new test points λ_{k+1} and μ_{k+1} using the standard 0.618 formulas for the updated interval $[a_{k+1}, b_{k+1}]$. Calculate function values $f(\lambda_{k+1})$ and $f(\mu_{k+1})$. Set $k = k + 1$ and return to Step 1.

4. Numerical Experiments

To validate the improved algorithm, we selected one simple and two more complex unimodal functions [?] for comparison with the 0.618 method. The algorithms were applied to these test functions on specified intervals, with performance measured by iteration count and solution accuracy at precision levels of 10^{-2} and 10^{-4} .

Example 1: $f(x) = x^2$, with minimum point at $x^* = 0$. Initial interval: $[-1, 100]$.

Example 2: $f(x) = x^2 - \sin(x)$, with minimum point at $x^* = -0.144275$. Initial interval: $[-1, 4]$.

Example 3: $f(x) = (x - 2)^4 + (x - 2)^2$, with minimum point at $x^* = 2.35424$. Initial interval: $[1, 4]$.

The comparison results are presented in Tables 1-3.

Table 1. Comparison of results for Function 1 | Method | Initial Interval | Iterations | Final Interval | Interval Reduction Rate |

| Method | Initial Interval | Iterations | Final Interval | Interval Reduction Rate |
|-----------------|------------------|------------|---------------------|--------------------------|
| 0.618 Method | $[-1, 100]$ | 23 | $[-0.0039, 0.0024]$ | 1.15431×10^{-2} |
| Improved Method | $[-1, 100]$ | 8 | $[-0.0012, 0.0008]$ | 2.11656×10^{-3} |

Table 2. Comparison of results for Function 2 | Method | Initial Interval | Iterations | Final Interval | Interval Reduction Rate |

| Method | Initial Interval | Iterations | Final Interval | Interval Reduction Rate |
|-----------------|------------------|------------|----------------------|--------------------------|
| 0.618 Method | $[-1, 4]$ | 18 | $[-0.1446, -0.1439]$ | 7.23312×10^{-3} |
| Improved Method | $[-1, 4]$ | 6 | $[-0.1443, -0.1442]$ | 1.15431×10^{-3} |

Table 3. Comparison of results for Function 3 | Method | Initial Interval | Iterations | Final Interval | Interval Reduction Rate |

| Method | Initial Interval | Iterations | Final Interval | Interval Reduction Rate |
|-----------------|------------------|------------|--------------------|--------------------------|
| 0.618 Method | $[1, 4]$ | 16 | $[2.3539, 2.3546]$ | 6.02040×10^{-3} |
| Improved Method | $[1, 4]$ | 5 | $[2.3542, 2.3543]$ | 9.00901×10^{-4} |

The experimental results show that the improved algorithm requires significantly fewer iterations than the 0.618 method while achieving better solution

accuracy. The generated sequence converges to the optimal solution within the required precision in finite steps, demonstrating both stable convergence and rapid convergence speed across various scenarios.

Further analysis reveals that while the 0.618 method computes only one new function value per iteration, the improved method computes two. However, the reduction in iteration count (by more than half) still yields overall improvement. Table 4 compares the interval reduction rates for Example 1 across different initial intervals.

Table 4. Comparison of algorithms for different intervals

| Initial Interval | 0.618 Method Iterations | Improved Method Iterations | Iteration Reduction |
|-------------------------------------|-------------------------|----------------------------|---------------------|
| Interval Reduction Rate Improvement | | | |
| ----- ----- | | | |
| [-1, 500] | 27 | 9 | 1/3 28% |
| [-1, 100] | 23 | 8 | 1/3 28% |
| [-1, 10] | 18 | 6 | 1/3 30% |
| [-1, 1] | 15 | 6 | 2/5 6% |
| [-1, 0.1] | 12 | 4 | 1/3 25% |
| [-1, 0.01] | 10 | 3 | 1/4 36% |

As shown in Table 4, when the initial interval is $[-1, 500]$, the iteration count is one-third that of the 0.618 method with a 28% improvement in interval reduction rate. For $[-1, 0.01]$, iterations are reduced to one-quarter with a 36% improvement. Notably, when the initial interval endpoints have function values differing by approximately 100 times or more, the improved algorithm rapidly reduces the interval and converges quickly to the minimum.

As interval length decreases, the 0.618 method's iteration count changes minimally or remains nearly constant. In contrast, the improved algorithm avoids stagnation and occasionally achieves even greater iteration reductions, demonstrating superior performance.

5. Conclusion

Line search algorithms are fundamental in single-factor experimental design, and their convergence speed directly impacts overall algorithm efficiency. Leveraging function information within the search interval to accelerate convergence is a key research focus. This paper developed a universal acceleration strategy by utilizing function values at interval endpoints to progressively narrow the search space at each iteration. Numerical experiments confirm that the proposed algorithm significantly accelerates existing line search methods, particularly when function values at interval endpoints differ substantially.

References

- [1] Chen Baolin. *Optimization Theory and Algorithms* [M]. Beijing: Tsinghua University Press, 2005: 254-280.
- [2] Xu Guogen, Zhao Housui, Huang Zhiyong. *Optimization Methods and Their MATLAB Implementation* [M]. Beijing: Beihang University Press, 2018: 12-18.

- [3] Zhang Ye. Discrete Balance Weight Optimization Based on Hybrid Genetic Algorithm [J]. *Electronic Components and Information Technology*, 2021, 5(1): 116-117.
- [4] Hu Mengying, He Zuguo. An Improved Polynomial Interpolation Method Based on Restarted Conjugate Gradient Ideas [J]. *Software*, 2015, 36(11): 48-51.
- [5] Li Xuewen, Yan Guifeng, Li Qingna. *Optimization Methods* [M]. Beijing: Beijing Institute of Technology Press, 2018: 101-118.
- [6] Yao Shengwei, Wu Yuping. A Line Search Acceleration Strategy and Its Application [J]. *Journal of Hechi University*, 2019, 39(2): 43-48.
- [7] Dong Yuanyuan, Xie Chengrong. An Improved 0.618 Method [J]. *Journal of Yangtze University*, 2014, 11(22): 4-6.
- [8] Lin Qingyu, Hong Zhiyong. Evaluation and Analysis of High-Voltage Distribution Network Construction Scale Based on Optimization Principles [J]. *Electronic Components and Information Technology*, 2020, 4(10): 72-73.
- [9] Emin Kahya. A New Unidimensional Search Method [J]. *Appl. Math. Comput.*, 2005: 163-179.

Author Biographies:

ZONG Yu-han (1997-), female, master' s student, Hebei Province. Research interests: optimization algorithms. Email: 379508292@qq.com.

ZHANG Yan-bo (2000-), male, undergraduate student. Research interests: computer science.

Note: Figure translations are in progress. See original paper for figures.

Source: ChinaXiv –Machine translation. Verify with original.