Proceedings of the 4th International Conference on Computing Innovation and Applied Physics
DOI: 10.54254/2753-8818/2025.22638

Gradient Descent Optimization for A Quartic Polynomial

Shiyun Xia[1], Yuming Cai[2], Xuanting Wang[3*], Shengjie Lin[4], Libiao Ling[5]

[1]*Colby College, Maine, USA*
[2]*University of California Santa Barbara, California, USA*
[3]*Shandong University, Jinan, China*
[4]*Rosedale Global High School, Fuzhou, China*
[5]*Guangdong Experimental High School, Guangzhou, China*
**Corresponding Author. Email: 27263892@qq.com*

Abstract: In this paper, we aim to minimize the function $g(w) = \frac{1}{50}(w^4 + w^2 + 10w)$ via gradient descent. However, the study involves many variables that play an important role in how the algorithm performs, such as step size, number of iterations, tolerance for the stopping criterion, and the starting value. To improve these drawbacks, this research proposes adding momentum to increase the convergence rate and gain more stability for this algorithm. This can be fixed by properly tuning the momentum parameters. This paper provides a thorough analysis of the choice of parameters and suggests guidelines for choosing an optimal configuration that can provide valuable insights into the optimization problem as a whole and practical advice for other application domains.

Keywords: Gradient Descent Optimization, Quartic Polynomial, Momentum, Convergence, Variable Tuning

1. Introduction

When trying to minimize $g(w) = \frac{1}{50}(w^4 + w^2 + 10w)$, crucial aspects consist of the starting point, step magnitudes, number of iterations, and stopping criterion. High starting points like 100 or -100 usually make the algorithm diverge, underscoring the necessity of picking proper initial values that are much closer to the actual minimum of g(w).

The assessment of how well the gradient descent algorithm minimizes the function, in other words, pinpoints the w value at which g(w) gets minimized, relied on looking at w at which g(w) attained its global minimum. The information available is that by tweaking the number of iterations, the perfect g(w) value at this point is realized at around 1827, while the same result can equally be approached with 1500 iterations. Just to put this into perspective, after 100 iterations, w in the best case approaches -0.194, which is far away from the real minimum.

In the case of tolerance less than 1e - 6, there is a gradual increase in the execution time, but the optimum value is within a reasonable frame. For example, with a tolerance of e -12, convergence is achieved at w-1.2347 after 2960 iterations.

Furthermore, the momentum technique is studied for application regarding augmentation and improvement. Keeping iterations, step sizes, tolerance levels, and the value of momentum at 0.9 for

DOI: 10.54254/2753-8818/2025.22638

the first three points reduces the execution time to 1423, which is even less than that in the case of a conventional gradient descent approach.

In addition, the momentum factor allows bigger step sizes to be used while keeping the process of optimization stable. For instance, with α 10 and momentum on, the algorithm converged at the 135th iteration with <-0.169969, and the w value that is desired is -1.2349.

Conversely, in the absence of momentum, convergence occurs at w near -1.2345 after 22 iterations using a comparable g(w).

A multitude of algorithms in machine learning utilize techniques based on gradients to optimize parameters. Nonetheless, the inherent characteristics of this function make its direct optimization through gradient descent a complex task. This document focuses on employing gradient descent optimization to reduce the specified function. We carefully evaluate elements like the size of the step, iteration count, tolerance, and initial point in our method.[1]

Moreover, our goal is to reduce the function's smooth approximation through gradient descent, akin to current techniques. The unique aspects of our method set it apart:

1. Our in-depth examination focuses on how varying step sizes, iterations, tolerances, and initial points influence the algorithm's convergence, as demonstrated by the data we have.

2. It is crucial to meticulously choose these parameters to attain the best outcomes. As an illustration, a step size that is excessively large or too small may result in either non?convergence or less than ideal convergence.

3. It's also observed that the solution's precision is greatly affected by the iteration count and the level of tolerance.

In the following parts of this document, we aim to showcase detailed experimental outcomes derived from in-depth data analysis and explore the consequences of our discoveries.

Additionally, we intend to juxtapose our method with other current techniques to underscore its strengths and weaknesses.

2. Methodology

2.1. Objective function

The specific function we aim to minimize is given by:

$$g(w) = \frac{1}{50}(w^4 + w^2 + 10w)$$

This function combines both polynomial terms (w^4,w^2) and a linear term (10w). The function g(w) is convex for certain regions and non-convex for others, making it an interesting target for testing the performance of gradient descent. The goal of the gradient descent algorithm is to find the value of w that minimizes this function.

2.2. Batch gradient descent optimization

In this project, Batch gradient descent optimization and momentum are used as the main numerical model. This optimization is illustrated through the equation below:

$$w = w_0 - \alpha \times f'(w_0)$$

In this equation, w represents the output, which is the optimizing value of the objective function. w_0 stands for the initial point, the point that the calculation starts. α is the step size, which determines the speed of descent in the direction of the gradient contribution. Finally, $f'(w_0)$ is the gradient contribution, and it ensures the function converges at the minimum value.[2]

DOI: 10.54254/2753-8818/2025.22638

2.3. Numerical gradient approximation

In this implementation, the gradient $\nabla g(w)$ is approximated using the central difference method for numerical differentiation. The gradient of a function g(w) is computed using a small perturbation h, as follows:

$$\nabla g(x) = \frac{g(w+h) - g(w-h)}{2h}$$

Where h is a small number (in this case, h=e^{-5}). This approach is useful when the exact analytical gradient is difficult to derive or compute. The central difference method provides a more accurate approximation compared to the forward or backward difference methods, as it averages the function's slope from both sides of the point.[3]

2.4. Convergence criteria

The gradient descent algorithm in this study employs a stopping condition based on the change in the function value. If the absolute difference between $g(w_{new})$and $g(w_{old})$ falls below a given tolerance ϵ, the algorithm is considered to have converged:

$$|g(w_{new}) - g(w_{old})| < \epsilon$$

Where ϵ is a small positive value (e.g., e^{-8}) that defines how precise the solution needs to be before the algorithm halts. This ensures that the solution is sufficiently close to the true minimum.

2.5. Momentum

In addition, momentum is used as an optimizing model for the gradient descent optimization. Momentum is a technique that helps improve both the convergence speed and stability of gradient-based optimization algorithms.

The most significant advantage of using momentum is its ability to accelerate convergence. In traditional gradient descent, the update at each step is proportional to the gradient of the function at that point, which can lead to slow progress in certain regions of the function's landscape.

Momentum builds upon past gradients to create more significant updates in directions where the gradient consistently points in the same direction. This reduces the number of iterations required to reach the optimal solution. As a result, gradient descent with momentum tends to converge more quickly compared to the standard approach.[4]

Mathematically, the update rule for gradient descent with momentum is given by:

$$v_{t+1} = \beta v_t + (1-\beta)\nabla g(w_t)$$

$$w_{t+1} = w_t - \alpha v_{t+1}$$

In this function, v_t represents the velocity or the exponentially weighted average of past gradients, α is the step size, and β is the momentum factor, typically set between 0.8 and 0.9. This formulation allows the updates to accumulate speed in the directions where the gradient continues to point in the same direction, thus accelerating convergence.

3. Result

We systematically varied the following variables:

Step Size (α): Values of 100,10,1,0.1,0.01,0.001,0.0001.

Number of Iterations: Between 100 and 1,000,000.

Tolerance: Ranging from 1×10^{-1} to 1×10^{-15}

DOI: 10.54254/2753-8818/2025.22638

Starting Point: Values of w=100,10,0,−10,−100

The convergence criterion was defined as the change in g(w) being less than the tolerance. The results were recorded for each configuration.

3.1. Experimental output

3.1.1. Effect of step size

Large Step Size (100, 10): The algorithm failed to converge when using large step sizes. For example, with α=100, the optimization failed after 2 iterations, indicating that the step size was too large to navigate the function's curvature. And with α=100, converged at iteration 2 with g(w) = 0.0 and the optimal value of w is 0.0, It is imprecise.

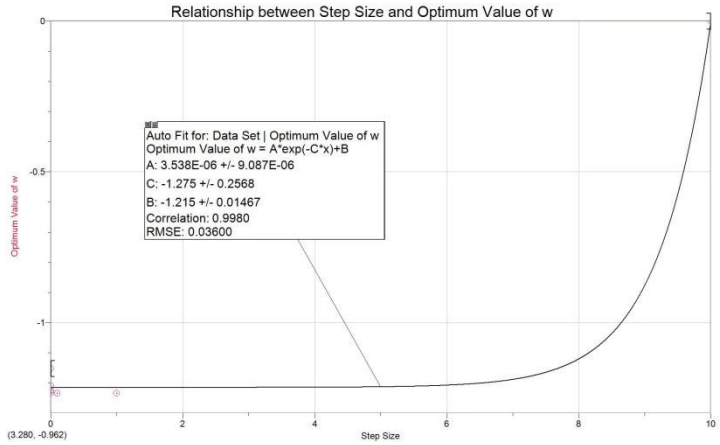

Figure 1: Relationship between step size from 0 to 10 and optimum value

Medium Step Size (1, 0.1, 0.01): A step size of 1 produced reliable results, converging at w=−1.2345 after 22 iterations, with g(w)≈−0.169969. Smaller step sizes, such as 0.01, also showed stable convergence, with convergence taking longer but achieving similarly accurate results.

Small Step Size (0.0001): While accurate, smaller step sizes resulted in significantly slower convergence, requiring up to 125,230 iterations to reach an optimal w.

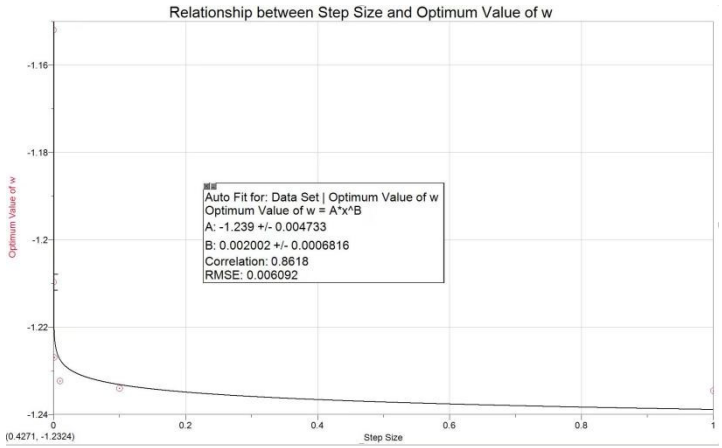

Figure 2: Relationship between step size from 0 to 1 and optimum value

The two figures show the relationship between step size and the optimum value. Distinctively, in Figure 1, the step size between 0 and 1 varies too slowly, so we use Figure 2 as a supplement.

DOI: 10.54254/2753-8818/2025.22638

3.1.2. Effect of iterations

Few Iterations (100, 500): When limiting the number of iterations, the algorithm failed to reach the optimal solution. With 100 iterations, the optimal w was approximately −0.194, far from the true minimum.

Many Iterations (up to 1,000,000): Increasing the iteration count allowed the algorithm to refine its approximation of w, especially when paired with smaller step sizes and tight tolerances. However, diminishing returns were observed beyond 10,000 iterations for many parameter settings.

3.1.3. Effect of tolerance

Large Tolerance (e^{-1},e^{-2},e^{-3}): The algorithm quickly converged to suboptimal values. For example, with tolerance=e^{-2}, convergence occurred after 2 iterations with w=0.0, which is not the true minimum.

Small Tolerance (e^{-8},e^{-10},e^{-12}): Smaller tolerances yielded more accurate results, but at the cost of increased computation. For tolerance=e^{-12}, convergence occurred at w≈−1.2347 after 2960 iterations.

The following Figure 3 intuitively shows the varies of optimum value.

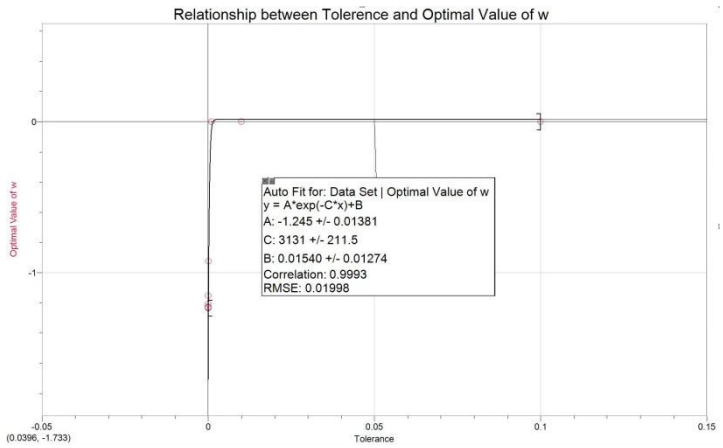

Figure 3: Relationship between tolerance from 0 to 0.15 and optimum value

3.1.4. Effect of starting point

Large Positive and Negative Starting Points: Starting at large positive values of w=100 and w=−100, the algorithm failed to converge, signaling that the gradient was misdirected at these points. However, starting at smaller values such as w=10 or w=−10, the algorithm was able to find the optimal solution.

Zero Starting Point: With w=0, the algorithm converged to w≈−1.232 after 1827 iterations.

Our results demonstrate the significant impact of variables tuning on the performance of gradient descent. The step size and tolerance, in particular, are critical in balancing the trade-off between convergence speed and accuracy.

Step Size: The experiments confirm that step sizes that are too large or too small can impede convergence. A moderate step size (0.01 to 1) provides the best balance between speed and accuracy.

Iterations and Tolerance: A higher iteration limit and smaller tolerance generally improve accuracy but may lead to diminishing returns in practical applications. Interestingly, the choice of the starting point influences the number of iterations required for convergence, particularly for larger initial values of w.

DOI: 10.54254/2753-8818/2025.22638

Furthermore, we use momentum as an optimization model for the gradient descent optimization. After applying this model, convergence occurs at w≈−1.232 after 1792 iterations.

When α = 10, converged at iteration 135 with g(w)≈−0.169969 and the optimal value of w is −1.2349. Momentum allows for the use of larger step sizes without compromising the stability of the optimization process. In standard gradient descent, large step sizes can cause the algorithm to overshoot the minimum, leading to divergent or imprecise behavior. However, by incorporating momentum, the updates become smoother, and the algorithm can handle larger step sizes without overshooting.

4. Discussion

In this project, we evaluate how the optimum values of a gradient descent optimization change when the starting point, step size, number of iterations, and tolerance change. Firstly, we find that different starting points result in different outputs. If the values of starting points are too large or too small, then the function optimization function won't converge. Secondly, the number of iterations affects the results. If the function is repeated for less than 1827 times, it is not going to converge. Thirdly, different step sizes lead to different results. If the step size is too large, the function will not converge. Fourthly, we investigate the effect of tolerance on the outputs. When the tolerance is too small, the results are not going to be effective.

In addition, we also explore the effect of applying the momentum algorithm on gradient descent optimization. Our computational results show that importing momentum can increase the efficiency of optimization, making the updates smoother.

In short, our paper explores how the optimum value of our objective function $g(w) = \frac{1}{50}(w^4 + w^2 + 10w)$ can change based on different starting points, step size, number of iterations, and tolerance. We also investigate in what ways does applying momentum algorithm makes the computation more efficient.

5. Conclusion

5.1. Outcome

In machine learning, the gradient descent algorithm is a common optimization algorithm used to solve the minimum value of the objective function.

5.2. Significance

It gradually approximates the optimal solution by constantly updating the parameters along the negative gradient direction of the objective function.

5.3. Broader implication

Stochastic gradient descent is widely used in deep learning, and the well-known deep neural networks use stochastic gradient descent as an optimization algorithm to iterate to approximate the optimal point. In addition, stochastic gradient descent is widely used in kernel methods and linear regression, two types of algorithms that play an important role in machine learning.

5.4. Future prospects

Gradient descent and its variants are widely used in machine learning and deep learning. The convergence analysis of gradient descent algorithms for convex optimization problems has been well discussed, but the convergence analysis of non-convex problems, such as deep neural networks, is

DOI: 10.54254/2753-8818/2025.22638

currently the focus of attention in the academic community, and we hope to be able to have a more in-depth study and discussion of such problems in the future.[5]

References

[1] *Abdallah, C., Lauvaux, T., Lian, J., Breon, F., Ramonet, M., Laurent, O., Ciais, P., Van Der Gon, H. D., Dellaert, S., Perrussel, O., Baudic, A., Utard, H., & Gros, V. (2023). A Gradient-Descent Optimization of CO2–CO–NOx Emissions over the Paris Megacity─The Case of the First SARS-CoV-2 Lockdown. Environmental Science Technology.https://doi.org/10.1021/acs.est.3c00566*

[2] *Andrychowicz, M., Denil, M., Gómez Colmenarejo, S., Hoffman, M. W., Pfau, D., Schaul, T., Google DeepMind, University of Oxford, & Canadian Institute for Advanced Research. (2016). Learning to learn by gradient descent[Journal-article].*

[3] *Ruder, S., Insight Centre for Data Analytics, NUI Galway, & Aylien Ltd. (2017). [An overview of gradient descent optimization algorithms] [Article].arXiv, 2–15.http://arxiv.org/abs/1609.04747v2*

[4] *Peng Xianlun. (2020).Convergence of the momentum gradient descent method*

[5] *Yuan Wenbo, Chen Qingyu, Gao Hongbing, & Wang Qinruo. (2018). An operation of gradient descent method in binocular vision navigation. In ELECTRONICS WORLD, ELECTRONICS WORLD (pp. 127–128) [Journal-article]. https://doi.org/10.19353/j.cnki.dzsj. (2018)*