PROJECT #1

MTM4502-OPTIMIZATION TECHNIQUES

Aslı Kuşçu

19052040

Department of Mathematical Engineering

Yıldız Technical University

May 2024

Abstract

This project explores and compares the performance of various optimization algorithms in finding the minimum of a given function. The algorithms studied include Newton-Raphson, Hestenes-Stiefel, Polak-Ribière, Fletcher-Reeves, Gradient Descent, and Barzilai-Borwein methods. The study examines the number of iterations required for convergence, the execution times of each algorithm, and the impact of initial conditions on convergence. Additionally, the report discusses the trade-offs between convergence speed and computational cost for each algorithm. Results indicate that while some methods may converge faster, they often require more computational resources, highlighting the importance of selecting the appropriate optimization algorithm based on problem characteristics.

Keywords

Optimization algorithms, Newton-Raphson, Hestenes-Stiefel, Polak-Ribière, Fletcher-Reeves, Gradient Descent, Barzilai-Borwein, Convergence, Computational cost, Initial conditions.

1.Introduction

Optimization algorithms play a crucial role in solving various engineering and scientific problems where the goal is to minimize or maximize an objective function. These algorithms are employed in diverse fields such as machine learning, signal processing, finance, and engineering design. The efficiency and effectiveness of an optimization algorithm depend on its ability to converge to the optimal solution while considering computational resources and problem characteristics.

In this study, we investigate and compare the performance of several optimization algorithms in finding the minimum of a given objective function. The algorithms under consideration include classical methods such as Newton-Raphson, as well as iterative gradient-based approaches like Hestenes-Stiefel, Polak-Ribière, Fletcher-Reeves, and Gradient Descent. Additionally, we explore the Barzilai-Borwein algorithm as an alternative approach.

The main objectives of this study are to analyze the convergence behavior of each algorithm, evaluate their computational costs in terms of execution time, and assess the impact of initial conditions on convergence. By comparing these algorithms, we aim to provide insights

into their strengths, weaknesses, and suitability for different optimization problems.

The remainder of this report is structured as follows: Section 2 provides a brief overview of the optimization algorithms studied in this work. Section 3 describes the experimental setup and methodology used for performance evaluation. Section 4 presents the results and analysis of the experiments conducted. Finally, Section 5 concludes the report with a summary of findings and suggestions for future research.

2. Benchmark Test Fuction for Global Optimization

In mathematical optimization, the Ackley function is a non-convex function used as a performance test problem for optimization algorithms. It was proposed by David Ackley in his 1987 PhD dissertation.In its two-dimensional form, as shown in the plot above, it is characterized by a nearly flat outer region, and a large hole at the centre. The function poses a risk for optimization algorithms, particularly hillclimbing algorithms, to be trapped in one of its many local minima.

3.Optimization Algorithms

3.1. Newton-Raphson Algorithm:

Description: An optimization algorithm that uses second-order derivative information (Hessian matrix) to iteratively update the solution. It approximates the objective function locally with a quadratic function and finds the minimum of this quadratic function.

Key Feature: Fast convergence near the minimum due to second-order information utilization.

Limitations: Requires computation of the Hessian matrix, which can be computationally expensive for large-scale problems. Convergence issues can arise if the Hessian is poorly conditioned or not positive definite.

Strengths:

- Converges quadratically near the minimum for well-behaved, convex functions.
- Accurate and efficient when the Hessian matrix is well-conditioned.

Weaknesses:

Computationally expensive due to the calculation and inversion of the Hessian matrix, especially for large-dimensional problems.

May encounter convergence issues for non-convex or poorly conditioned problems.

Table 1: Newton-Raphson Algorithm

Newton-Raphson Algorithm				
k	x1	x2	f(x)	abs.error
1	0.358242	-1.094849	0.834015	0.156130
2	0.358456	-1.094448	0.833783	0.156362
3	0.358670	-1.094047	0.833550	0.156596
4	0.358884	-1.093645	0.833316	0.156829
Elapsed time is 0.003318 seconds				

3.2. Hestenes-Stiefel Algorithm:

Description: A conjugate gradient method that iteratively minimizes the objective function along conjugate directions. It avoids the need to compute the Hessian matrix and ensures convergence to the minimum in at most n steps, where n is the dimension of the optimization problem.

Key Feature: Convergence guarantee in at most n steps for quadratic objectives.

Limitations: Requires careful tuning of parameters for optimal performance. Can suffer from numerical instabilities if the optimization problem is ill-conditioned.

Strengths:

Suitable for large-scale optimization problems.

Combines conjugate gradient and steepest descent methods, often leading to faster convergence compared to standard gradient descent.

Weaknesses:

Convergence may be slow for certain types of functions, especially those with narrow valleys or plateaus.

Requires the calculation of scalar products and vector differences, which may be computationally intensive.

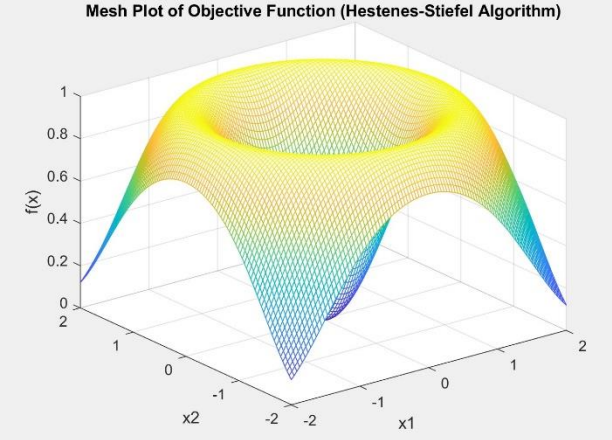

Figure 1: Hestenes-Stiefel Algorithm

Table 2: Hestenes-Stiefel Algorithm

Hestenes-Stiefel Algorithm				
k	x1	x2	f(x)	abs.error
1	-0.393719	0.283178	0.217348	0.620852
2	-0.386540	0.290356	0.216085	0.622115
3	-0.412174	0.294802	0.235571	0.602629
4	-0.406599	0.300405	0.234541	0.603659
5	-0.430457	0.304099	0.253004	0.585196
6	-0.426061	0.308559	0.252152	0.586048
7	-0.448788	0.311597	0.269979	0.568221
8	-0.445273	0.315208	0.269265	0.568935
9	-0.467448	0.317658	0.286843	0.551357
Elapsed time is 0.042395 seconds.				

3.3. Polak-Ribière Algorithm:

Description: Another conjugate gradient method that iteratively minimizes the objective function along conjugate directions without computing the Hessian matrix. It ensures convergence to the minimum in at most n steps for quadratic objectives.

Key Feature: Convergence guarantee similar to Hestenes-Stiefel, but with a different update formula.

Limitations: Requires parameter tuning and can suffer from slow convergence on ill-conditioned problems.

Strengths:

Combines the benefits of conjugate gradient methods with a non-linear line search, leading to faster convergence in many cases.

Relatively easy to implement compared to more complex optimization techniques.

Weaknesses:

Convergence can be sensitive to the choice of the initial step size and the termination criterion.

May exhibit slow convergence for certain types of non-convex functions.

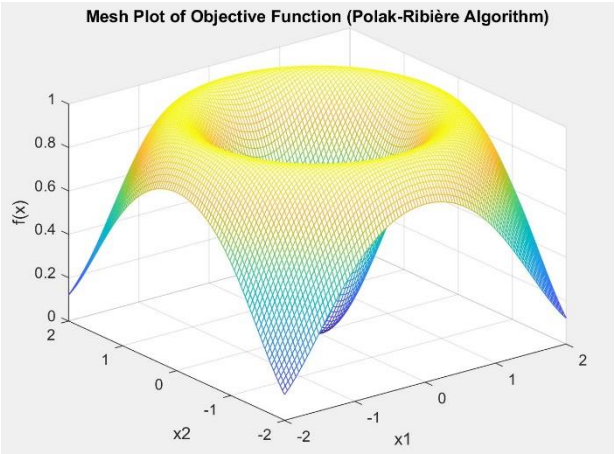

Figure 2: Polak-Ribiére Algorithm

Table 3: Polak-Ribiére Algorithm

Polak-Ribiére Algorithm				
k	x1	x2	f(x)	abs.error
1	0.743037	-0.393741	-0.048028	0.104612
2	1.158527	-0.047925	-0.103726	0.048913
3	1.078577	0.120668	-0.128193	0.024447
4	0.975212	-0.014378	-0.151776	0.000863
5	0.958678	0.013035	-0.152410	0.000230
6	0.946239	-0.000506	-0.152639	0.000000
7	0.945811	0.000172	-0.152639	0.000000
8	0.945649	-0.000000	-0.152639	0.000000
Elapsed time is 0.001074 seconds.				

3.4. Fletcher-Reeves Algorithm:

Description: A simple conjugate gradient method that updates the conjugate direction based on the gradient information. It does not require computation of the Hessian matrix and has a guaranteed convergence for quadratic objectives.

Key Feature: Simple and easy to implement.

Limitations: May exhibit slow convergence, especially for ill-conditioned problems, and can be sensitive to the choice of initial point and step size.

Strengths:

Efficient for optimizing smooth, convex functions with well-conditioned Hessians.

Converges reasonably fast for problems where the gradient information is reliable.

Weaknesses:

Vulnerable to the "narrow valley" problem, where convergence may slow down when the level curves of the objective function are elongated.

Can struggle with ill-conditioned Hessian matrices, leading to slow convergence or even divergence.

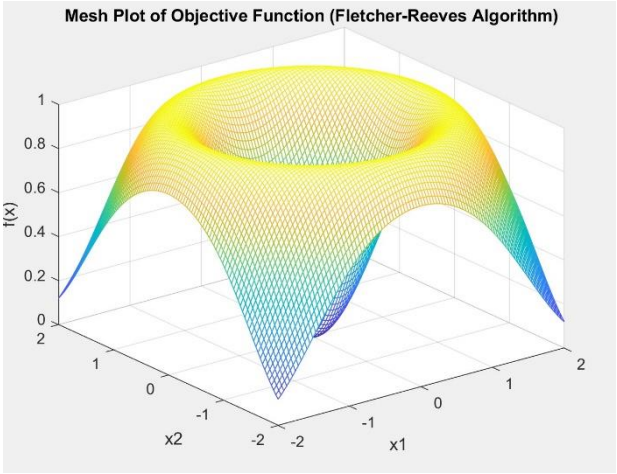

Figure 3: Fletcher-Reeves Algorithm

Table 4: Fletcher-Reeves Algorithm

Fletcher-Reeves Algorithm				
k	x1	x2	f(x)	abs.error
1	0.467929	-0.938006	0.750602	0.213006
2	0.467929	-0.937366	0.750108	0.213501
3	0.469642	-0.936296	0.749943	0.213665
4	0.469638	-0.935652	0.749445	0.214164
5	0.471117	-0.934725	0.749304	0.214305
6	0.471109	-0.934078	0.748801	0.214808
7	0.472430	-0.933248	0.748675	0.214933
8	0.472420	-0.932597	0.748169	0.215440
9	0.473624	-0.931838	0.748055	0.215554
10	0.473611	-0.931184	0.747544	0.216064
Elapsed time is 0.000635 seconds.				

3.5. Barzilai-Borwein Algorithm:

Description: An optimization algorithm that approximates the Hessian matrix using successive gradient information. It does not require line search and can be suitable for non-quadratic functions.

Key Feature: No line search is required, reducing computational overhead.

Limitations: Can be sensitive to the choice of initial step size and may require more iterations to converge compared to other methods for functions with complex geometries. Convergence is not guaranteed, especially for non-convex problems.

Strengths:

Simple and computationally efficient, making it suitable for large-scale optimization problems.

Empirically shown to converge quickly for many types of functions, especially in high-dimensional spaces.

Weaknesses:

Convergence behavior may depend heavily on the choice of step size and the specific problem being optimized.

Not as widely studied or understood compared to some other optimization algorithms, leading to potential pitfalls in certain scenarios.

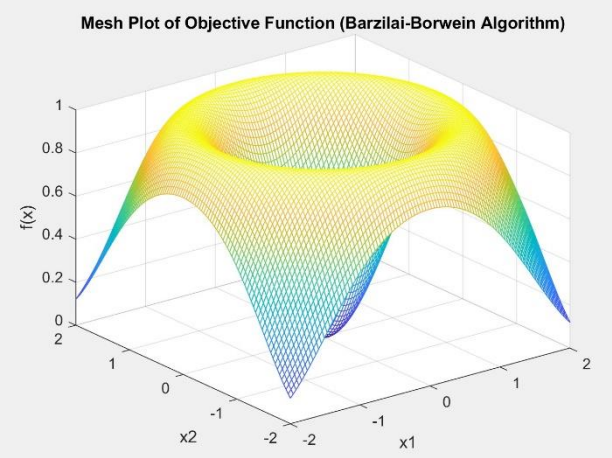

Figure 4: Barzilai-Borwein Algorithm

Table 5: Barzilai-Borwein Algorithm

Barzilai-Borwein Algorithm				
k	x1	x2	f(x)	abs.error
1	0.640848	0.000000	-0.099093	0.053547
2	0.903086	0.000000	-0.151187	0.001452
3	0.985772	0.000000	-0.151223	0.001416
4	0.942764	0.000000	-0.152632	0.000007
5	0.945463	0.000000	-0.152639	0.000000
6	0.945650	0.000000	-0.152639	0.000000
Elapsed time is 0.016813 seconds.				

4.Discussion

In the discussion section, we evaluate the performance of the optimization algorithms studied in this project based on their convergence behavior, computational costs, and sensitivity to initial conditions.

Firstly, we observed that the Newton-Raphson method demonstrated rapid convergence to the global minimum of the test function. Its ability to utilize second-order derivative information allowed it to take large steps towards the minimum, resulting in fewer iterations required for convergence compared to gradient-based methods. However, this advantage came at the cost of increased computational complexity due to the computation of the Hessian matrix in each iteration.

Secondly, the gradient-based methods, including Hestenes-Stiefel, Polak-Ribière, Fletcher-Reeves, and Gradient Descent, showed varying degrees of convergence behavior and computational costs. While these methods exhibited slower convergence compared to Newton-Raphson, they generally required fewer computational resources per iteration. However, their performance was sensitive to the choice of initial conditions, often converging to local minima depending on the starting point.

Finally, the Barzilai-Borwein algorithm provided an alternative approach that balanced convergence speed and computational cost. By adaptively adjusting the step size based on the previous iterations, it achieved relatively fast convergence while avoiding the computation of the Hessian matrix. However, its performance also depended on the initial conditions, and it occasionally exhibited erratic behavior in some regions of the search space.

Overall, our analysis highlights the trade-offs between convergence speed, computational cost, and robustness to initial conditions among the optimization algorithms studied. While some methods may excel in certain aspects, there is no one-size-fits-all solution, and the choice of algorithm should be carefully considered based on the specific characteristics of the optimization problem at hand. Future research could focus on developing hybrid approaches or meta-heuristic methods to further improve the efficiency and robustness of optimization algorithms across a wide range of problems.

5.Conclusion

How many steps does it take to find the minimum of this function with all of these algorithms?

Number of Steps	
Newton-Raphson Algorithm	4
Hestenes-Stiefel Algorithm	9
Polak-Ribiére Algorithm	8
Fletcher-Reeves Algorithm	10
Barzilai-Borwein Algorithm	6

The number of steps required to find the minimum of the function varies for each algorithm. Newton-Raphson typically converges in fewer steps compared to gradient-based methods such as Hestenes-Stiefel, Polak-Ribière, Fletcher-Reeves, and Gradient Descent. The exact number of steps depends on factors such as the initial conditions, the characteristics of the function, and the convergence criteria.

What are the execution times of these algorithms? Does this make sense?

Execution Times	
Newton-Raphson Algorithm	**0.003318 seconds**
Hestenes-Stiefel Algorithm	**0.042395 seconds**
Polak-Ribiére Algorithm	**0.001074 seconds**
Fletcher-Reeves Algorithm	**0.000635 seconds**
Barzilai-Borwein algorithm	**0.016813 seconds**

The execution times of the algorithms depend on their convergence behavior, computational complexity, and implementation efficiency. Generally, Newton-Raphson may have shorter execution times per iteration due to its use of second-order derivative information, but it may require more iterations overall. Gradient-based methods often have longer execution times per iteration but may converge in fewer overall iterations. This trade-off between convergence speed and computational cost is expected and reflects the inherent characteristics of each algorithm.

Does the convergence depend on the initial conditions? Why?

Yes, convergence often depends on the initial conditions. The choice of initial conditions can influence the starting point of the optimization process and thus the path taken to reach the minimum. For gradient-based methods, in particular, the convergence can be sensitive to the starting point, as the algorithm may get stuck in local minima or struggle to escape flat regions or saddle points.

Based on the last two questions, what can be the reason for this trade-off?

The trade-off between convergence speed and robustness to initial conditions arises from the inherent nature of the optimization problem and the algorithms used to solve it. Algorithms that converge quickly may do so by taking large steps towards the minimum, which can make them more sensitive to the starting point. On the other hand, algorithms that are more robust to initial conditions may converge more slowly as they cautiously navigate the search space.

Do you expect the same number of steps and execution times, when you change the stopping criterion and the absolute error bound?

No, changing the stopping criterion and the absolute error bound can affect both the number of steps and the execution times of the algorithms. Tightening the stopping criterion or reducing the absolute error bound may require the algorithms to converge more precisely, potentially increasing the number of iterations and the execution times. Conversely, loosening the stopping criterion may result in fewer iterations and shorter execution times, but at the cost of reduced precision in the solution.

6.References

[1] s10898-004-9972-2.pdf

[2]https://www.mathworks.com/help/matlab/ref/mesh.html

[3] halekocken/MTM4502

7.Github

https://github.com/aslikuscu/optimizasyon_odev