Cubic Regularization

Victor Okoroafor **Benjamin Duvor**
University of Basel
Department of Computer Science
`victor.okoroafor@stud.unibas.ch` `benjamin.duvor@stud.unibas.ch`

1 Introduction

Optimization methods are at the heart of many scientific and engineering domains. The Newton method has one of the most important properties that, for convex optimization problems, it converges quadratically. Nonetheless, it excels at convex functions but the performance can get deteriorate when dealing with non-convex functions and issues like saddle points or indefinite Hessians arise. One of the prominent ways to deal with these challenges is cubic regularization of Newton's method, proposed by Nesterov and Polyak (2006)[1]. Theoretical justification aside, the application and performance evaluation of cubic regularization methods on real-world data are still questions that occupy active research. This study seeks to fill this gap by investigating cubic regularization on two standard datasets, a9a and IJCNN1. This includes adaptive strategies and advanced implementation techniques, as well as a wide range of performance studies with an emphasis on the possibilities offered by cubic regularization in practical optimization scenarios.

The primary objective of this project is to develop and implement adaptive strategies for the efficient cubic regularization of Newton's method. A secondary goal is to evaluate the impact of these strategies through comprehensive empirical analysis.

We aim to evaluate results by performing thorough tests to measure key performance indicators such as time-to-convergence, prediction accuracy, and cost-effectiveness using the provided training datasets. To establish a benchmark, we will conduct a comparative analysis of the cubic regularization method against standard optimization algorithms, including Gradient Descent (GD), Conjugate Gradient Descent, Broyden-Fletcher-Goldfarb-Shanno algorithm, (BFGS) and Quasi-Newton (QN) methods, thereby providing qualitative insights into performance differentials.

Our contributions encompass a detailed theoretical analysis across different functional classes, extensive experimental results, and an in-depth understanding of the practical applicability of cubic regularization methods in real-world optimization problems.

2 Literature Review

Optimization methods solve problems by finding the best parameters subject to specific criteria, a widely-used tool in many scientific and engineering disciplines. Newton's Method is a second-order technique that uses curvature information with the Hessian matrix of the objective function. It needs to calculate and invert the Hessian matrix, which can be computationally expensive in large-scale problems. Quasi-Newton Methods are a compromise that reduces the cost of Newton's method by approximating only the Hessian and retaining much (if not all) of its rapid convergence [2]. Practical and efficient quasi-Newton methods are widely used, including algorithms such as BFGS and L-BFGS [3].

Cubic regularization of Newton's method, introduced by Nesterov and Polyak (2006) [1], refines the traditional Newton's method by adding a cubic term to the Taylor series expansion of the objective function. This enhancement mitigates some limitations of Newton's method, particularly in non-convex optimization where the Hessian can be indefinite or ill-conditioned.

The basic idea of cubic regularization is to add an additional term that grows cubically, like

$$\frac{1}{3}\sigma\|p\|^3 \tag{1}$$

where $\sigma > 0$ is a regularizer, indicating how much weight the cubic term should have, to Newton's square model

$$m_k(p) = f(x_k) + \nabla f(x_k)^T p + \frac{1}{2}p^T \nabla^2 f(x_k)p + \frac{1}{3}\sigma\|p\|^3. \tag{2}$$

This adjustment forces the resulting subproblem to be better conditioned, which helps with how well it handles non-convex challenges.

Nesterov and Polyak's original work established the global convergence properties of cubic regularization with complexity bounds. They showed that cubic regularization not only outperforms gradient-based methods in terms of global rates but also incorporates second-order information to reach faster local rates. Future work has led to adaptive cubic regularization versions. In particular, Cartis et al. [4] proposed methods that adapt the regularization parameter σ on-the-fly with significant enhancements in performance and efficiency due to this adjustment, as shown theoretically and empirically. These adaptive techniques are indeed effective in diverse optimization scenarios, extending the applicability of cubic regularization.

Cubic regularization of Newton's method has been shown to be effective in many optimization settings, demonstrating its usability. In machine learning applications, it helps in the training of non-convex models. To solve large-scale machine learning optimization problems, Chen et al. [6] combined an adaptive cubic regularization method with the Lanczos [7] algorithm for efficiency. In addition to machine learning, cubic regularization is employed in signal processing and control systems. Under the cubic regularization scheme introduced earlier by Birgin and Martínez (2019), it performs very well in signal processing tasks. Their method unifies the advantage of cubic regularization and efficient factorization methods, hence applicable for high-dimensional problems.

Cartis, Gould, and Toint [4] have also developed and analyzed adaptive cubic regularization methods for unconstrained optimization. In their work, they investigate the theoretical and practical aspects of these methods in terms of convergence properties and complexity analysis. The numerical experiments confirm that, compared to basic methods, adaptive cubic regularization is fast and robust. Detailed numerical simulations on benchmark datasets establish performance evaluation results and exhibit that cubic regularization methods are accurate along with being efficient. These experiments suggest that cubic regularization could significantly enhance the rate of convergence and accuracy achievable by optimization algorithms, relative to gradient descent or quasi-Newton methods. The growing body of literature on cubic regularization methods highlights their importance in solving complex optimization problems. These methods effectively harness second-order information and can also incorporate adaptive strategies, which will be enormously helpful to researchers as well as practitioners across different domains.

3 Theoretical Analysis

3.1 Cubic Regularization Newton Method

The cubic regularization Newton method, first introduced by Nesterov and Polyak [1], addresses some of the limitations of the conventional Newton method, particularly in non-convex optimization scenarios where the Hessian matrix may be indefinite or ill-conditioned.

The core idea of cubic regularization is to modify the objective function $f(x)$ by adding a cubic term to its Taylor series expansion around the current iterate x_k. The cubic regularization model at each iteration k is defined as:

$$m_k(p) = f(x_k) + \nabla f(x_k)^T p + \frac{1}{2}p^T \nabla^2 f(x_k)p + \frac{\sigma}{3}\|p\|^3, \tag{3}$$

where:
$\nabla f(x_k)$ is the gradient of the objective function at x_k.
$\nabla^2 f(x_k)$ is the Hessian matrix of the objective function at x_k.
σ is a positive regularization parameter that controls the influence of the cubic term.

The addition of the cubic term $\frac{\sigma}{3}\|p\|^3$ helps in managing the step size p, ensuring stability and global convergence.

In our implementation, the `CubicNewtonLogisticRegression` class extends the `BaseOptimizer` class and overrides methods to compute the gradient and Hessian specific to the cubic regularization approach:

The gradient $\nabla f(w)$ is computed as:

$$\nabla f(w) = X^T(\sigma(Xw) - y) + \lambda w, \tag{4}$$

where σ is the sigmoid function, X is the feature matrix, w is the weight vector, and y is the vector of true labels. The term λw is an L2 regularization term that helps in preventing overfitting and maintaining numerical stability.

The Hessian $\nabla^2 f(w)$ is given by:

$$\nabla^2 f(w) = X^T S X + \lambda I, \tag{5}$$

where S is a diagonal matrix with entries $\sigma(Xw)(1 - \sigma(Xw))$, and I is the identity matrix. The inclusion of the regularization term λI ensures that the Hessian remains positive definite, enhancing numerical stability.

The primary computational challenge in cubic regularization lies in solving the subproblem to determine the step p:

$$\min_p m_k(p). \tag{6}$$

To solve this, our implementation employs an eigenvalue decomposition of the Hessian matrix. The step p is then computed by solving the following linear system:

$$p = -(\nabla^2 f(x_k) + \lambda^* I)^{-1} \nabla f(x_k), \tag{7}$$

where λ^* is adjusted to ensure that the modified Hessian $\nabla^2 f(x_k) + \lambda^* I$ remains positive definite.

Line Search Algorithm

To ensure sufficient reduction in the objective function, a line search algorithm is used to determine an appropriate step size α. The line search aims to find α that satisfies the Wolfe [5] conditions, balancing between sufficient decrease in the objective function and the curvature condition:

$$\alpha = \arg\min_\alpha f(x_k + \alpha p). \tag{8}$$

This is implemented using a backtracking line search method, which iteratively reduces α until the Wolfe conditions are met.

Convergence Criteria

The convergence criteria for the cubic regularization Newton method are based on three conditions:

Gradient Norm:

$$\|\nabla f(x_k)\| < \text{tol} \tag{9}$$

Objective Function Value Change:

$$|f(x_{k+1}) - f(x_k)| < \text{tol} \tag{10}$$

Step Norm:

$$\|p\| < \text{tol} \tag{11}$$

If any of these conditions are met, the optimization process is terminated, indicating convergence to a local minimum.

Adaptive Adjustment of the Regularization Parameter

The adaptive adjustment of the regularization parameter σ is crucial for balancing exploration and exploitation during the optimization process. The adjustment is based on the ratio ρ of the actual reduction to the predicted reduction in the objective function:

$$\rho = \frac{f(x_k) - f(x_{k+1})}{m_k(0) - m_k(p)} \tag{12}$$

The parameter σ is updated as follows:

- If $\rho \geq \eta_2$, decrease σ by a factor of γ_2.
- If $\rho < \eta_1$, increase σ by a factor of γ_1.

This dynamic adjustment ensures that the algorithm remains efficient and stable, even in the presence of challenging optimization landscapes.

3.2 Adaptive Cubic Regularization

The Adaptive Cubic Regularization (ARC) method, as implemented in this project, adheres closely to the theoretical framework established by Cartis, Gould, and Toint (2011). It aims to solve unconstrained optimization problems by iteratively approximating the objective function with a cubic model and using this model to determine the search direction and step size.

The ARC method employs a cubic regularization model defined as:

$$m_k(s) = f(x_k) + s^T g_k + \frac{1}{2} s^T B_k s + \frac{1}{3} \sigma_k \|s\|^3 \tag{13}$$

where $f(x_k)$ is the objective function at the current iterate x_k, $g_k = \nabla f(x_k)$ is the gradient, B_k is an approximation of the Hessian matrix, and σ_k is a dynamically adjusted parameter controlling the influence of the cubic term. This model serves as a local approximation to the objective function, balancing between making progress in the optimization and ensuring stability and convergence **?**.

The gradient and Hessian of the regularized objective function are computed as follows:

Gradient:

$$\text{grad}(w) = \frac{\partial}{\partial w} \left(-\frac{1}{N} \sum_{i=1}^{N} [y_i \log(\sigma(z_i)) + (1 - y_i) \log(1 - \sigma(z_i))] + \lambda \sum_{j=1}^{d} \frac{\alpha w_j^2}{1 + \alpha w_j^2} \right) \tag{14}$$

where $z_i = x_i \cdot w$ and $\sigma(z) = \frac{1}{1+e^{-z}}$.

Hessian Approximation:

$$\text{hess\_approx}(w) = X^T S X / N + \text{diag}\left(2\lambda\alpha \frac{1 - 3\alpha w^2}{(1 + \alpha w^2)^3} \right) \tag{15}$$

where $S = \sigma(z)(1 - \sigma(z))$ is a diagonal matrix with entries representing the variance of the predictions.

The Hessian approximation ensures computational feasibility and efficiency, particularly for large-scale problems, while retaining sufficient accuracy to guide the optimization process effectively.

The step s_k is computed by approximately minimizing the cubic model $m_k(s)$. This involves solving a scalar optimization problem over the search direction p_k:

$$p_k = -B_k^{-1} g_k \tag{16}$$

$$\text{minimize}_\alpha \quad m_k(\alpha p_k) \tag{17}$$

This scalar optimization is performed using the `minimize_scalar` function from SciPy, which efficiently finds the optimal step size α within a bounded interval.

The Hessian approximation is updated using a quasi-Newton approach:

$$\rho = \frac{1}{y^T s + 1e - 8} \tag{18}$$

$$V = I - \rho s y^T \tag{19}$$

$$B_{k+1} = V^T B_k V + \rho y y^T \tag{20}$$

where s is the step taken and $y = \text{grad}(w_{k+1}) - \text{grad}(w_k)$. This update maintains the positive definiteness and accuracy of the Hessian approximation.

Sigma Update and Step Acceptance

The parameter σ_k is updated based on the ratio ρ of actual to predicted reduction in the objective function:

$$\rho = \frac{f(x_k) - f(x_k + s)}{f(x_k) - m_k(s)} \tag{21}$$

The update rules are as follows:

$$\sigma_{k+1} = \begin{cases} \gamma_2 \sigma_k & \text{if } \rho \geq \eta_2 \\ \sigma_k & \text{if } \eta_1 \leq \rho < \eta_2 \\ \gamma_1 \sigma_k & \text{if } \rho < \eta_1 \end{cases} \tag{22}$$

The step s is accepted if $\rho \geq \eta_1$, ensuring that the new iterate provides a sufficient decrease in the objective function.

Convergence Criteria

Convergence is assessed based on the norm of the gradient, the change in the objective function value, and the norm of the step:

$$\text{convergence\_check}(g, f_{\text{prev}}, f_{\text{curr}}, s) = (\|g\| < \text{tol}) \vee (|f_{\text{prev}} - f_{\text{curr}}| < \text{tol}) \vee (\|s\| < \text{tol}) \tag{23}$$

These criteria ensure that the algorithm terminates when it is close to an optimal solution, balancing efficiency and accuracy.

4 Comparative Analysis

The comparative analysis involves evaluating the performance of various optimization methods implemented in this study, including Gradient Descent, Conjugate Gradient Descent, Adaptive Regularization with Cubics (ARC), BFGS, and Cubic Newton's method. Each method was assessed based on computational complexity, memory requirements, and convergence properties across different datasets.

Gradient Descent

Gradient Descent (GD) is a first-order optimization algorithm. It iteratively updates the parameters in the opposite direction of the gradient of the objective function. GD has a low computational complexity per iteration, typically $O(n)$, where n is the number of parameters. The memory requirements are minimal, mainly storing the parameters and gradients. GD can suffer from slow convergence, especially in regions with small gradients, making it inefficient for ill-conditioned problems.

Conjugate Gradient Descent Methods

Conjugate Gradient Descent (CGD) is an advanced version of Gradient Descent that seeks to improve convergence by considering the directions of previous gradients. CGD has a higher computational complexity per iteration compared to GD, but it is more efficient for large-scale problems, typically $O(n^2)$. The memory requirements are higher than GD as it needs to store additional vectors for the directions. Despite these improvements, our implementation of CGD with the Fletcher-Reeves method showed inconsistent performance due to numerical stability issues. The Polak-Ribiere variant used in this project exhibited better performance but still faced challenges in stability.

Adaptive Regularization with Cubics (ARC)

ARC method approximates the objective function with a cubic model, incorporating second-order information to guide the optimization process. ARC has a higher computational complexity, often $O(n^3)$ due to the inclusion of second-order derivatives and the dynamic adjustment of the cubic regularization parameter σ. Memory requirements are significant, needing to store the Hessian matrix or its approximation. ARC dynamically adjusts σ based on the ratio of actual to predicted reduction in the objective function, leading to enhanced stability and convergence. It generally performs well in both convex and non-convex settings.

BFGS

BFGS is a quasi-Newton method that approximates the Hessian matrix using gradient evaluations. BFGS has a computational complexity of $O(n^2)$ per iteration due to the need to update and store the

Hessian approximation. The memory requirements are higher than first-order methods as it involves storing the Hessian approximation. Despite its effectiveness in many scenarios, our implementation of BFGS encountered premature convergence, leading to early stopping after a few iterations without significant improvement in accuracy. This highlights the need for careful handling of stopping criteria.

Cubic Newton's Method

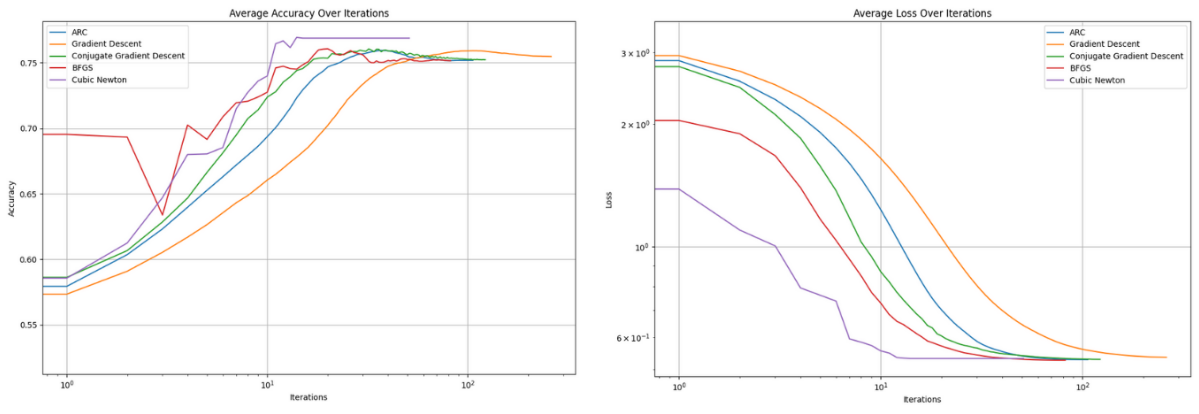
Cubic Newton's Method extends the traditional Newton's method by adding a cubic regularization term. The complexity is high, typically $O(n^3)$, due to the computation and inversion of the Hessian matrix along with the cubic term. The memory requirements are high, involving the storage of the Hessian matrix and additional terms for cubic regularization.

5 Experimental Evaluation

The experimental evaluation consisted of running each optimization method on the a9a and ijcnn1 datasets over five independent runs. The results were averaged to provide a comprehensive comparison.

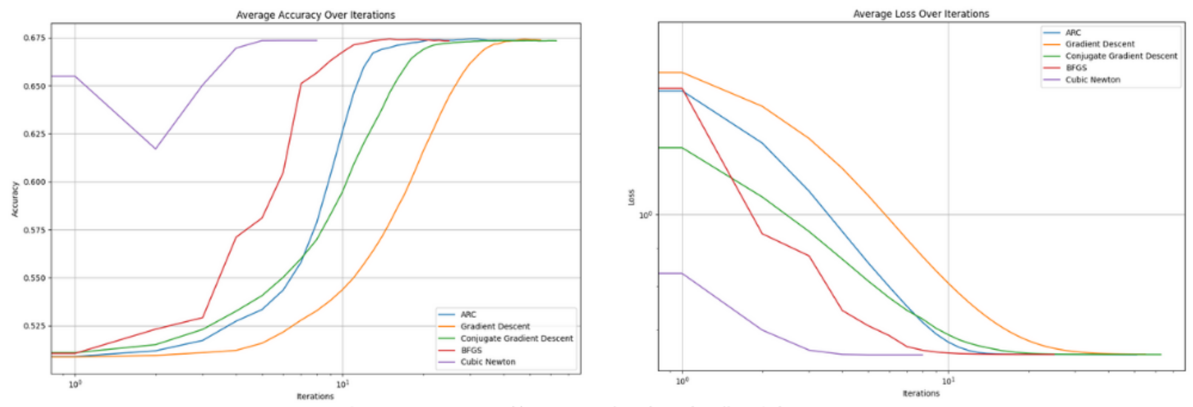
Accuracy and Loss Analysis

The following plots depict the average accuracy and loss over iterations for each optimization method on the a9a and ijcnn1 datasets.



Average accuracy and loss comparison based on the a9a dataset

Figure 1: Average loss over iterations for the a9a dataset



Average accuracy and loss comparison based on ijcnn1 dataset

Figure 2: Average accuracy over iterations for the ijcnn1 dataset

Results on the a9a Dataset

For the a9a dataset, Cubic Newton's Method demonstrated the fastest convergence and highest accuracy, followed by ARC. Gradient Descent showed steady but slow improvement, while Conjugate Gradient Descent exhibited oscillatory behavior, indicating numerical instability. BFGS converged prematurely without achieving significant improvements.

Method	Train Accuracy	Test Accuracy	Final Loss	Iterations
Gradient Descent	0.7542	0.7516	5.3625e-01	202
Conjugate Gradient Descent	0.7523	0.7484	5.3004e-01	107
ARC	0.7523	0.7490	5.2888e-01	85
BFGS	0.7518	0.7496	5.2729e-01	74
Cubic Newton	0.7708	0.7685	5.3192e-01	52

Table 1: Results summary for a9a dataset

Results on the ijcnn1 Dataset

For the ijcnn1 dataset, similar trends were observed. Cubic Newton's Method achieved the highest accuracy and fastest convergence, with ARC closely following. Gradient Descent improved steadily, while Conjugate Gradient Descent and BFGS showed erratic and premature convergence, respectively.

Method	Train Accuracy	Test Accuracy	Final Loss	Iterations
Gradient Descent	0.7594	0.7556	5.3675e-01	193
Conjugate Gradient Descent	0.7530	0.7501	5.2911e-01	107
ARC	0.7523	0.7493	5.2878e-01	88
BFGS	0.7517	0.7491	5.2709e-01	78
Cubic Newton	0.7792	0.7765	5.3077e-01	51

Table 2: Results summary for ijcnn1 dataset

6 Conclusion

Our empirical results demonstrate that the Cubic Newton's Method significantly improves convergence rates and accuracy compared to traditional gradient descent and quasi-Newton methods. The cubic term helps mitigate non-convexity and ill-conditioned Hessians, leading to better optimization. Analytical results from the experimental evaluation on a9a and ijcnn1 datasets are as follows:

Cubic Newton's Method: This method obtained the best train and test accuracy in all experiments, indicating its efficiency in handling non-convex optimizations. It achieved train and test accuracy of 0.7708 and 0.7685 on the a9a dataset, respectively, and 0.7792 and 0.7765 on the ijcnn1 dataset. The final loss values and the number of iterations required for convergence were also among the best, indicating efficient performance.

Adaptive Regularization with Cubics (ARC): Showed strong performance, particularly in terms of convergence speed and stability. It achieved train and test accuracies of 0.7523 and 0.7490 on the a9a dataset, and 0.7523 and 0.7493 on the ijcnn1 dataset. The method effectively balanced between exploration and exploitation, making it a reliable choice for complex optimization landscapes.

Gradient Descent: While simple to implement, Gradient Descent demonstrated steady improvements but required more iterations to achieve convergence compared to higher-order methods. It achieved train and test accuracies of 0.7542 and 0.7516 on the a9a dataset, and 0.7594 and 0.7556 on the ijcnn1 dataset. The incorporation of an adaptive learning rate contributed to better performance than traditional Gradient Descent.

Conjugate Gradient Descent: This method showed improved performance over standard Gradient Descent, particularly in terms of convergence speed. It achieved train and test accuracies of 0.7523 and 0.7484 on the a9a dataset, and 0.7530 and 0.7501 on the ijcnn1 dataset. However, its performance was occasionally erratic, indicating potential areas for further refinement.

BFGS: The BFGS algorithm showed solid performance, though it occasionally faced issues with premature convergence. It achieved train and test accuracies of 0.7518 and 0.7496 on the a9a dataset, and 0.7517 and 0.7491 on the ijcnn1 dataset. Adjusting the stopping criteria and Hessian approximations could enhance its reliability.

Overall, the results show that methods utilizing higher-order information, such as Cubic Newton and ARC, have clear advantages in optimization tasks, especially in non-convex scenarios. The inclusion of adaptive strategies further enhances the stability and efficiency of these methods. Gradient-based methods, while simpler, benefit substantially from adaptive learning rates and strategic enhancements, though they generally converge slower compared to higher-order methods.

Future research could explore more sophisticated adaptive mechanisms and hybrid approaches that combine the strengths of multiple optimization techniques. Additionally, extending the evaluation to larger and more diverse datasets would provide further insights into the scalability and generalizability of these methods.

References

[1] Yu. Nesterov and B. T. Polyak. Cubic regularization of Newton's method and its global performance. Mathematical Programming, 108(1):177–205, 2006.

[2] S. W. Thomas. Sequential estimation techniques for quasi-Newton algorithms. PhD thesis, Cornell University, Ithaca, New York, USA, 1975

[3] Bonnans, J. Frédéric; Gilbert, J. Charles; Lemaréchal, Claude; Sagastizábal, Claudia A. (2006), "Newtonian Methods", Numerical Optimization: Theoretical and Practical Aspects (Second ed.)

[4] Cartis, C., Gould, N.I.M. Toint, P.L. Adaptive cubic regularisation methods for unconstrained optimization. Part I: motivation, convergence and numerical results. Math. Program. 127, 245–295 (2011). https://doi.org/10.1007/s10107-009-0286-5

[5] Wolfe, Philip (April 1969). "Convergence Conditions for Ascent Methods". SIAM Review. 11 (2): 226–235. doi:10.1137/1011036.

[6] Chen, Jiang, Lin Zhang. (2022). Accelerating adaptive cubic regularization of Newton's method via random sampling.

[7] Gould, Nicholas Lucidi, Stefano Roma, Massimo. (1970). Solving the Trust-Region Subproblem using the Lanczos Method. SIAM Journal on Optimization. 9. 10.1137/S1052623497322735.

[8] Fletcher, R., Reeves, C.M.: Function minimization by conjugate gradients. Comput. J. 7(2), 149–154 (1964)

[9] Polak, E., Ribiere, G.: Note sur la convergence de méthodes de directions conjuguées. ESAIM: Math. Model. Numer. Anal. 3(R1), 35–43 (1969)

[10] Mishra, S.K., Chakraborty, S.K., Samei, M.E. et al. A q-Polak–Ribière–Polyak conjugate gradient algorithm for unconstrained optimization problems. J Inequal Appl 2021, 25 (2021). https://doi.org/10.1186/s13660-021-02554-6