mathematics

MDPI

Article

Gradient Method with Step Adaptation

Vladimir Krutikov [1,2,*], Elena Tovbis [1], Svetlana Gutova [2], Ivan Rozhnov [1,3] and Lev Kazakovtsev [1,*]

1 Institute of Informatics and Telecommunications, Reshetnev Siberian State University of Science and Technology, 31 Krasnoyarskii Rabochii Prospekt, 660037 Krasnoyarsk, Russia; sibstu2006@rambler.ru (E.T.); ris2005@mail.ru (I.R.)
2 Department of Applied Mathematics, Kemerovo State University, 6 Krasnaya Street, 650043 Kemerovo, Russia
3 Laboratory "Hybrid Methods of Modeling and Optimization in Complex Systems", Siberian Federal University, 79 Svobodny Prospekt, 660041 Krasnoyarsk, Russia
* Correspondence: krutikovvn@rambler.ru (V.K.); levk@bk.ru (L.K.)

Abstract: The paper solves the problem of constructing step adjustment algorithms for a gradient method based on the principle of the steepest descent. The expansion of the step adjustment principle, its formalization and parameterization led the researchers to gradient-type methods with incomplete relaxation or over-relaxation. Such methods require only the gradient of the function to be calculated at the iteration. Optimization of the parameters of the step adaptation algorithms enables us to obtain methods that significantly exceed the steepest descent method in terms of convergence rate. In this paper, we present a universal step adjustment algorithm that does not require selecting optimal parameters. The algorithm is based on orthogonality of successive gradients and replacing complete relaxation with some degree of incomplete relaxation or over-relaxation. Its convergence rate corresponds to algorithms with optimization of the step adaptation algorithm parameters. In our experiments, on average, the proposed algorithm outperforms the steepest descent method by 2.7 times in the number of iterations. The advantage of the proposed methods is their operability under interference conditions. Our paper presents examples of solving test problems in which the interference values are uniformly distributed vectors in a ball with a radius 8 times greater than the gradient norm.

Keywords: minimization method; relaxation; gradient method; step adaptation; convergence rate

MSC: 90C30

Academic Editor: Janez Žerovnik

Received: 15 November 2024
Revised: 10 December 2024
Accepted: 24 December 2024
Published: 27 December 2024

Citation: Krutikov, V.; Tovbis, E.; Gutova, S.; Rozhnov, I.; Kazakovtsev, L. Gradient Method with Step Adaptation. *Mathematics* **2025**, *13*, 61. https://doi.org/10.3390/math13010061

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Gradient minimization methods are easy to implement, have low iteration costs, and use a small amount of memory, which determines their applicability in solving high-dimensional problems. The advantage of gradient methods is the absence of restrictions on the objective function convexity and their high degree of noise immunity. The noted properties explain their widespread use in solving various applied optimization problems like optimal control, signal processing, robotics [1–4] and, in particular, applications in the field of data analysis, machine learning and deep learning [5–9].

For the problem of minimizing a smooth function, if it is strongly convex, the gradient descent method is known to have a global linear convergence rate [10–12]. However, many fundamental problems of machine learning, such as least squares regression or logistic regression, are reduced to problems of minimizing functions that are non-convex. This has led the researchers to the study of properties of convexity and strong convexity for the objective function of an optimization problem that are suitable for applications of this type. One of the best-known properties is the Polyak–Lojasiewicz gradient dominance condition [10,13].

The Polyak–Lojasiewicz condition is true for a sufficiently large class of non-convex problems. This condition is known to be sufficient to show the global linear convergence rate of gradient descent for sufficiently smooth problems without convexity assumptions. In recent years, the gradient dominance condition has been extensively studied in various areas of optimization and related sciences.

There are a number of ways to adjust the step of gradient methods. From the convergence point of view on a wide set of function classes, the most universal way to adjust the step is by the steepest descent method [11]. However, a possible problem here is the applicability of such settings in the case of noise.

Without denying the merits of the gradient descent method, it must be said that it turns out to be very slow when moving along a ravine, and as the number of variables of the objective function increases, such behavior of the method becomes typical.

A number of step-adjusting methods are based on the use of constants of the function class [10–15]. In many applications, it is not possible to obtain precise information about the gradient and/or the objective function at each iteration of the method. This has led researchers to study the behavior of first-order methods that can operate under noise. In the case of absolute or relative gradient errors, there are a number of ways to adjust the step of gradient methods. Methods for adjusting the step of a gradient method under noise conditions have been studied in a number of works [11,16–21]. The settings of the step of a gradient method here are based on the use and tuning of the corresponding constants of the function class. The influence of relative gradient noise on the convergence rate of the gradient method is studied in [20,21]. Here, as well, the settings of the gradient method step are based on the use and tuning of the corresponding constants of the function class.

In machine learning applications, it is well known that carefully designed learning rate (step size) schedules can significantly improve the convergence of commonly used first-order optimization algorithms. Therefore, the method of choosing the step size adaptively becomes an important research question [22].

Taking into account the convergence of the steepest descent method on a wide variety of function classes, it seems relevant to construct algorithms for adjusting the step of the gradient method based on the principle of the steepest descent method, which is not inferior in efficiency to the steepest descent method and is suitable for solving problems under conditions of significant relative interference on the gradient.

In this paper, we propose algorithms for step adaptation of the gradient method based on the imitation of the principle of the steepest descent method. The main goal of the step adjustment algorithm is to obtain a new point such that the gradient forms an angle of 90 degrees with the previous gradient at this point. We propose several step adaptation algorithms. The proposed methods are studied numerically on a wide range of test problems. The analysis of the proposed algorithms is carried out on a number of multidimensional test functions. We compared the efficiency of the proposed methods with the steepest descent method. To analyze the noise immunity of the methods to relative noise imposed on the function gradient, we carried out a significant number of experimental studies. In some experiments, the noise significantly exceeds the true function gradient.

The main contributions of the work are as follows:

1. The principle of step adaptation of the gradient method is developed.
2. Several step adaptation algorithms are proposed.
3. The proposed methods use only one gradient value per iteration.

4. A step adaptation method is proposed such that in the case without interference, its iteration costs are either equivalent to the number of iterations or significantly less than the steepest descent method costs.
5. The proposed methods were studied under conditions of relative interference on the gradient and their efficiency was established.
6. The obtained algorithms converge at a high rate in the case where the radius of the ball of uniformly distributed interference significantly exceeds the norm of the gradient value.

The rest of the paper is organized as follows: In Section 2, the problem under study is stated. In Section 3, algorithms for step adaptation in the gradient method are presented. In Section 4, methods with incomplete relaxation, super relaxation and mixed relaxation are considered. Section 5 presents the theoretical convergence analysis. In Section 6, the numerical experiment results are given. In Section 7, a brief discussion is provided. Section 8 concludes the work.

2. Problem Statement and Related Work

Let us consider the problem of minimizing a convex function $f(x)$ on R^n. We study gradient methods, in which successive approximations are constructed according to the equations:

$$x_{k+1} = x_k - h_k s_k,\ \ s_k = g_k / \|g_k\|. \tag{1}$$

Here, $g_k = \nabla f(x_k)$ is the descent direction and h_k is the step of one-dimensional search. In the steepest descent method,

$$h_k = \arg\min_{h \in R} f(x_k - h s_k)\ . \tag{2}$$

One of the features of minimization methods is the choice of the step value (learning rate). When choosing a constant step, the method may not converge or have the oscillations near the minimum point. One of the methods for preventing the oscillation of gradient descent is to slow down the parameter updates by decreasing the learning rate. This can be performed by changing the learning rate based on how many epochs through the data have been performed. These approaches typically add additional hyperparameters to control how quickly the learning rate decays [23].

Adaptive methods for selecting a step at each point allow the dynamics of the objective function values to be taken into account and do not contain parameters such as the Lipschitz constant or an estimate of the distance from the starting point to the set of exact solutions to the problem [24]. Adaptive methods that adjust the step size "on the fly" have become widespread in large-scale optimization for their ability to converge robustly and are particularly beneficial when training deep neural networks [25]. Adaptive choices of step sizes allow optimization algorithms to accelerate quickly according to the local curvature and smoothness of the optimization landscape. However, in theory, there are few parameter-free algorithms, and, in practice, there are many search heuristics [26].

The adaptive step was first proposed by Polyak [11]. In his method (which does not need to estimate the smoothness parameter of the objective function), the step was calculated as

$$h_k = \frac{f(x_k) - f^*}{\|\nabla f(x_k)\|_2^2}, \tag{3}$$

where f^* is the optimal function value.

In [25], the authors proposed a stochastic version of the classical Polyak step size:

$$h_k = \frac{f_i(x_k) - f_i^*}{c\|\nabla f_i(x_k)\|_2^2}, \tag{4}$$

where the parameter $0 < c < R$ is usually chosen as $c = 1/2$ for optimal convergence. This version was further improved in [27].

Paper [22] presents a general framework based on the Polyak step size to set the learning rate adaptively for first-order optimization methods with momentum.

Also, Jiang et al. [28] described two stochastic variants of the Polyak step size, AdaSPS and AdaSLS. Berrada et al. [29] designed an extension of the Polyak step size where each update only uses the loss function and its derivative rather than the full objective function and its derivative, the learning rate is clipped to the maximal learning-rate hyperparameter η and the minimum f^* is replaced by the lower-bound of 0. The idea of gradient approximation in step size calculation was used in [30,31]. Loss values were also used to adjust the step size in the method with a moving target [32].

In [33], the authors proposed a method with the adaptation via adjustment of the proximal function itself. Since each dimension has its own dynamic rate, this dynamic rate grows with the inverse of the gradient magnitudes, large gradients have smaller learning rates and small gradients have large learning rates. This method laid the foundation for the AdaGrad family [23,34–39], which has shown good results on large-scale learning problems. Many of them are based on using gradient updates scaled by the square roots of the exponential moving averages of the squared past gradients. For instance, in [23], instead of accumulating the sum of squared gradients over all time, the author restricted the window of past gradients that are accumulated to be some fixed size. AdaGrad-Norm [40,41] was developed with a single step size adaptation based on the gradient norm. Vaswani et al. [35] improved AdaGrad performance using the following step-sizes:

$$h_k = \min\left\{\frac{f_{ik}(x_k) - f_{ik}^*}{c\|\nabla f_{ik}(x_k)\|_2^2}, h_{\max}\right\}, \tag{5}$$

and

$$h_k = \min\left\{\frac{f_{ik}(x_k) - f_{ik}^*}{c\|\nabla f_{ik}(x_k)\|_{A_k^{-1}}^2}, h_{\max}\right\}, \tag{6}$$

where h_{max} is the upper bound on the step size and A_k is a preconditioner matrix.

In [38], the step was chosen as follows:

$$h_k = \frac{\|s_k\|_2^2}{s_k^T y_k}, \tag{7}$$

where $s_k = x_{k+1} - x_k$, $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$.

Generally, adaptive step sizes from the AdaGrad family of methods are particularly successful when training deep neural networks [28]. Theoretical results for the advantage of AdaGrad-like step sizes over the plain stochastic gradient descent in the non-convex setting were presented in [42].

The ADAM method [43] combines classical momentum [44] (using a decaying mean instead of a decaying sum) with RMSProp [45] to improve performance. In [46], the RMSProp was combined with Nesterov's accelerated gradient. Reddi [37] proposed new variants of the ADAM algorithm with "long-term memory" of past gradients. The authors in [47] apply the variance reduction technique to construct the adaptive step size in ADAM.

An ESGD scheme based on the equilibration preconditioner was proposed in [48]. The authors take the absolute value of the Hessian eigenvalues to improve the method's behavior, in particular, in the presence of saddle points. The authors in [49] present two adaptive step length schemes for strongly convex differentiable stochastic optimization problems: a recursive scheme and a cascading scheme. A general nonlinear update rule for the learning rate in batch and stochastic gradient descent was proposed in [50]. This method is shown to achieve robustness in the relationship between the learning rate and the Lipschitz constant, and near optimal convergence rates in both the batch and stochastic settings. An adaptive learning rate rule that employs importance weights was presented in [51].

He et al. [52] proposed a mini-batch semi-stochastic gradient descent (mS2GD) algorithm based on the competitive Barzilai–Borwein step size. An adaptive optimization algorithm with gradient bias correction (AdaGC) was demonstrated in [22]. In this algorithm, the iterative direction is improved using the gradient deviation and momentum, and the step size is adaptively revised using the second-order moment of gradient deviation.

In [53], AdaBelief is proposed to adapt the step size according to the "belief" in the current gradient direction. Using the exponential moving average of the noisy gradient as the prediction of the gradient at the next time step, if the observed gradient greatly deviates from the prediction, this method take a small step; if the observed gradient is close to the prediction, the method take a large step. AdaDerivative [54], in contrast to AdaBelief, can adaptively adjust step sizes by applying the exponential moving average of the derivative term using past gradient information without the smoothing parameter, thereby avoiding the overshoot problem.

Successive piecewise-affine approximations are used for minimization in the works [55–57].

In [5], two adaptive step size estimation methods are proposed for the complex-valued Nesterov accelerated gradient algorithm.

For a class of problems with a sufficiently smooth objective function satisfying the Polyak–Loyasiewicz condition, in paper [16], an adaptive gradient method is proposed that uses the concept of an inexact gradient. However, in this work, it is still necessary to know the exact estimate of the magnitude of the absolute gradient error. In [58], an algorithm is proposed that involves adjusting not only the smoothness constant of the function, but also the magnitude of the absolute error of the gradient. In paper [21], two adaptive algorithms are proposed for problems with objective functions satisfying the Polyak–Loyasevich condition, in the presence of relative inaccuracy in specifying the gradient.

Let us formulate the basic principle of step adjusting in the gradient minimization method with step adaptation. The step adjustment is carried out only on the basis of information about the function gradients. In the steepest descent method (1), (2), an exact one-dimensional search is performed (2). Successive gradients are orthogonal to each other:

$$(g_k, g_{k+1}) = 0. \tag{8}$$

Given the fact that in the steepest descent method, pairs of adjacent gradients are mutually orthogonal, we can construct a step adaptation method using the value of the scalar product (g_k, g_{k+1}):

1. The step is too large if there is an obtuse angle between adjacent gradients, that is, $(g_k, g_{k+1}) \le 0$. Therefore, the step should be reduced.
2. If there is an acute angle between adjacent gradients, that is, $(g_k, g_{k+1}) > 0$, then the step is too small, and it should be increased.

Another adaptation idea is to replace complete relaxation with some degree of incomplete relaxation or over-relaxation. Another possibility for organizing the step adaptation algorithm is to randomize the strategies of incomplete relaxation and over-relaxation.

Each of the listed strategies determines only the moment of decreasing or increasing the step. Another important question is how and how much to increase or decrease the step at the iteration. Each of the noted strategies should be provided with quantitative values of step updating. In this paper, we use some constant coefficients of step decrease and increase, as well as estimates for the current optimal step in order to use them to calculate the step-adjusting coefficients. The article describes the noted step adaptation strategies and formulates adaptation algorithms. In the next section, these ideas will be formulated in the form of algorithms.

3. Algorithms for Step Adaptation in the Gradient Method

The simplest algorithm for adapting the step of the gradient method is based on the idea of adjusting the orthogonality of successive gradients. If an angle between adjacent gradients is obtuse, $(g_k, g_{k+1}) \le 0$, then the step should be reduced. In the case of an acute angle between adjacent gradients, $(g_k, g_{k+1}) > 0$, the step should be increased. The following Algorithm 1 works on this basis.

Algorithm 1 (A1(q))

1. Set $q > 1$, the search step $h_0 > 0$, the initial point x_0.
2. **For** $k = 0,1,2,\ldots$ **do**
   - 2.1 Search for new approximation $x_{k+1} = x_k - h_k s_k$, $s_k = g_k / \|g_k\|$.
   - 2.2 **If** $(s_k, g_{k+1}) > 0$
     then $z_k = q$
     else $z_k = 1/q$.
   - 2.3 Compute the new search step $h_{k+1} = z_k h_k$

Condition (8) $(s_k, g_{k+1}) = 0$ ensures complete relaxation. Here and below, $s_k = g_k / \|g_k\|$. We will consider algorithms with incomplete relaxation and over-relaxation. Denote by $y(h) = (\nabla f(x_k - hs_k), s_k)$ a derivative of the function $\varphi(h) = f(x_k - hs_k)$ with respect to h. If the function is quadratic, then $y(h)$ is a linear function of h. Figure 1 shows this situation.

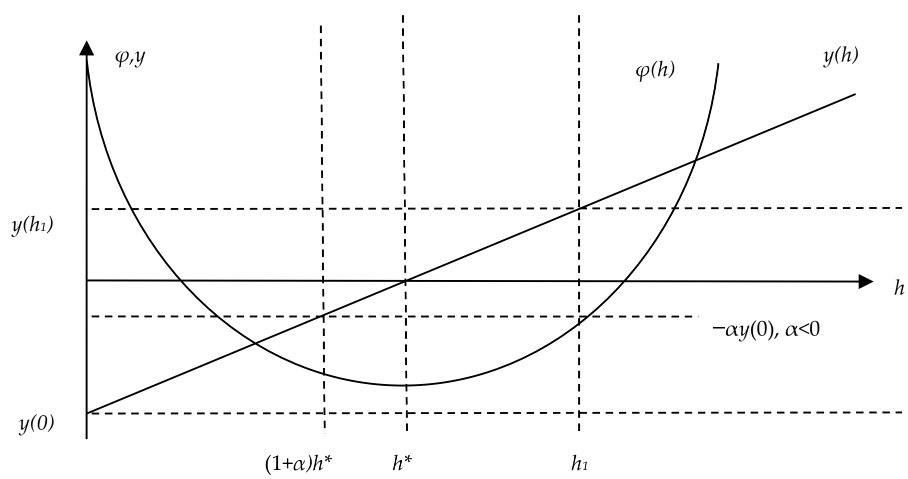


Figure 1. Function $\phi(h)$ and its derivative.

Assuming the function $\phi(h)$ is quadratic, based on two observations $y(0)$ and $y(h_1)$, we compose a linear representation:

$$y(h) = y(0) + h\frac{y(h_1) - y(0)}{h_1}. \tag{9}$$

Denote the step providing (8) by h^*. Condition (8) for (9) has the form:

$$y(h^*) = y(0) + h^*\frac{y(h_1) - y(0)}{h_1} = 0. \tag{10}$$

Solving (10), we find

$$h^* = h_1\frac{y(0)}{y(0) - y(h_1)}. \tag{11}$$

In (11), the discrepancy (error) between the current step h_1 and the optimal step h^* is explicitly given. The error coefficient at the current step can reach large values. Taking into account that the function being minimized is not quadratic, the error coefficient $y(0)/(y(0) - y(h_1))$ in (11) can be used to change the step. This enables us to determine the shift of the current step h_1 towards the predicted h^* using some degree of their convergence, for example,

$$h^+ = h_1\sqrt{\frac{y(0)}{y(0) - y(h_1)}}. \tag{12}$$

In the future, we will use Equation (12) in the adaptation algorithms. To avoid errors associated with noise when calculating $y(h)$, we impose a restriction on the radical expression in (12):

$$\frac{y(0)}{y(0) - y(h_1)} \le q, \quad q > 1. \tag{13}$$

Considering that the numerator and denominator of the last expression are negative, we may rewrite (13), making them positive: $|y(0)|/\left|\left(y(0) - y(h_1)\right)\right| \le q, q > 1$. To avoid division errors, the inequality can be written in the following form:

$$|y(0)| \le q|y(0) - y(h_1)|, \quad q > 1. \tag{14}$$

Using (14), we obtain the setting for the step

$$h^+ = \sqrt{z} \times h_1, \tag{15}$$

where

$$z = \begin{cases} q \quad if \ |y(0)| > q|y(0) - y(h_1)|, \\ \dfrac{|y(0)|}{|y(0) - y(h_1)|} \quad otherwise. \end{cases} \tag{16}$$

We use the step tuning (15), (16) in the adaptation Algorithm 2.

Algorithm 2 (A2(q))
1. Set $q > 1$, the search step $h_0 > 0$, the initial point x_0.
2. **For** $k = 0,1,2,\ldots$ **do**
   2.1 Search for new approximation $x_{k+1} = x_k - h_k s_k$, $s_k = g_k / \|g_k\|$.
   2.2 **If** $|(s_k, g_k)| > q|(s_k, g_k) - (s_k, g_{k+1})|$
   then $z_k = q$
   else $z_k = \dfrac{|(s_k, g_k)|}{|(s_k, g_k) - (s_k, g_{k+1})|}$.
   2.3 Compute the new search step

$$h_{k+1} = \sqrt{z_k}\, h_k. \tag{17}$$

4. Algorithms for the Step Adaptation of the Gradient Method with Incomplete Relaxation, Super Relaxation and Mixed Relaxation

In the previous section, the step was adjusted based on a model h^*. In this section, we will choose a model

$$h = (1+\alpha)h^*,\ \alpha > -1. \tag{18}$$

To organize an algorithm of Algorithm 1 type, we need a bound on the scalar product (s_k, g_{k+1}) to decide whether to increase or decrease the current step.

Assuming the function is quadratic, we use model (15) to calculate the boundary

$$\begin{aligned} y((1+\alpha)h^*) &= y(0) + (1+\alpha)h^* \frac{y(h_1) - y(0)}{h_1} \\ &= y(0) + (1+\alpha)h_1 \frac{y(0)}{y(0) - y(h_1)} \frac{y(h_1) - y(0)}{h_1} \\ &= y(0) - (1+\alpha)y(0) = -\alpha y(0). \end{aligned} \tag{19}$$

Exceeding the boundary $y(h) > -\alpha y(0)$ means that the step h is too small. The case $y(h) \le -\alpha y(0)$ means that the step is too large (in Figure 1, this boundary is shown by a dotted line for a negative value of α).

Based on (19) and the last remark, we formulate Algorithm 3 with a fixed adaptation step.

Algorithm 3 (A3(q, α))
1. Set $q > 1$, the search step $h_0 > 0$, the initial point x_0, parameter $\alpha > -1$.
2. **For** $k = 0,1,2,\ldots$ **do**
   2.1 Search for new approximation $x_{k+1} = x_k - h_k s_k$, $s_k = g_k / \|g_k\|$.
   2.2 **If** $(s_k, g_{k+1}) > -\alpha (s_k, g_k)$
   then $z_k = q$
   else $z_k = 1/q$.
   2.3 Compute the new search step $h_{k+1} = z_k h_k$.

In Step 2.2, the value $-\alpha(s_k, g_k)$ acts as a boundary for the step adaptation parameter (s_k, g_k).

Next, we will consider adaptation based on the predicted step value using a quadratic model of the function. From (12), we obtain

$$h^*_\alpha = (1+\alpha)h^* = h_1 \frac{(1+\alpha)y(0)}{y(0)-y(h_1)}. \tag{20}$$

As an adjusted step, we will take the following:

$$h^+ = h_1 \sqrt{\frac{(1+\alpha)y(0)}{y(0)-y(h_1)}}. \tag{21}$$

Instead of expression (21), we can use dependencies reflecting the tendency $h^+ \to h^*_\alpha$.

Further, in the adaptation algorithms, we will use Equation (21). To avoid errors associated with noise when calculating $y(h)$, we impose a restriction on the radical expression in (12):

$$\frac{(1+\alpha)y(0)}{y(0)-y(h_1)} \le q, \quad q > 1. \tag{22}$$

Considering that the numerator and denominator of the last expression are negative, we may rewrite (22), making them both positive: $((1+\alpha)|y(0)|)/|(y(0)-y(h_1))| \le q, q > 1$. To avoid division errors, the inequality can be written in the following form:

$$(1+\alpha)|y(0)| \le q|y(0)-y(h_1)|, q > 1. \tag{23}$$

Using (23), we obtain step adjusting:

$$h^+ = \sqrt{z}h_1, \tag{24}$$

where

$$z = \begin{cases} q & \text{if } (1+\alpha)|y(0)| > q|y(0)-y(h_1)|, \\ \dfrac{(1+\alpha)|y(0)|}{|y(0)-y(h_1)|} & \text{otherwise}. \end{cases} \tag{25}$$

Let us formulate the algorithm of the gradient method with step adaptation taking into account the predicted value and type of relaxation (21). We use the step tuning (24), (25) in the adaptation algorithm (Algorithm 4).

Algorithm 4 (A4(q, α))

1. Set $q > 1$, the search step $h_0 > 0$, the initial point x_0, parameter $\alpha > -1$.
2. **For** $k = 0,1,2,\ldots$ **do**
   2.1 Search for new approximation $x_{k+1} = x_k - h_k s_k$, $s_k = g_k / \|g_k\|$.
   2.2 **If** $(1+\alpha)|(s_k, g_k)| > q|(s_k, g_k) - (s_k, g_{k+1})|$
   then $z_k = q$
   else $z_k = \dfrac{(1+\alpha)|(s_k, g_k)|}{|(s_k, g_k) - (s_k, g_{k+1})|}$.
   2.3 Compute the new search step

$$h_{k+1} = \sqrt{z_k}h_k. \tag{26}$$

We organize mixed step adaptation by randomizing the parameter α

$$\alpha \in [a, b], a > -1, b > a. \tag{27}$$

In our numerical experiments, we used uniform distribution on the segment when choosing the parameter α in accordance with (27). The algorithm of the gradient method with step adaptation with randomized predicted value and type of relaxation (21) is presented below. We use the step tuning (24), (25) in the adaptation algorithm (Algorithm 5).

Algorithm 5 (A5(q, α[a, b]))

1. Set $q > 1$, the search step $h_0 > 0$, the initial point x_0, parameters of the segment $[a,b]$ $a > -1$, $b > a$.
2. **For** $k = 0,1,2,\ldots$ **do**
   2.1 Search for new approximation $x_{k+1} = x_k - h_k s_k$, $s_k = g_k / \|g_k\|$.
   2.2 Set $\alpha \in [a, b]$.
   2.3 **If** $(1+\alpha)|(s_k, g_k)| > q|(s_k, g_k) - (s_k, g_{k+1})|$
   then $z_k = q$
   else $z_k = \dfrac{(1+\alpha)|(s_k, g_k)|}{|(s_k, g_k) - (s_k, g_{k+1})|}$.

2.4 Compute the new search step

$$h_{k+1} = \sqrt{z_k}\, h_k. \tag{28}$$

In the next section, the efficiency of the proposed step-adaptive gradient method algorithms will be investigated on test functions.

5. Convergence Analysis

Let us study the change in the convergence rate when applying to a gradient a relative interference uniformly distributed on a ball of radius

$$R(x) = \Delta \|\nabla f(x)\|. \tag{29}$$

Approximate estimates of costs for a given Δ are based on a comparison of estimates for $\Delta = 0$. In methods for solving systems of equations, any estimates are based on the use of the boundaries of the matrix eigenvalues. In this case, if a gradient method with a constant step is used, then its step is determined based on the boundaries of the eigenvalues spectrum.

We do not have the boundaries of the matrices of second derivatives for the functions being minimized; however, these matrices also vary significantly depending on the current point. For an approximate estimate of the dependence of costs on Δ, we obtain a relation for the simplest quadratic function and calculate the increase in these costs compared to $\Delta = 0$. Considering that the estimates of the convergence rate on quadratic functions of the gradient method with an optimal step coincide with the estimates of the convergence rate for the steepest descent method, we will use the results for the steepest descent method when correlating the results with noise.

Let us estimate the convergence rate of the gradient method with a constant step in the presence of noise. Consider a one-dimensional function whose gradient is calculated with noise

$$f(x) = x^2/2, \quad \nabla f(x) = g(x) = x + \Delta|x|\eta, \ \eta \in [0,1], \tag{30}$$

where η is a random number. To minimize it, we use a gradient method with a constant step

$$x^+ = x - hg = x - h(x + \Delta|x|\eta) = x(1-h) - h\Delta|x|\eta. \tag{31}$$

Find the expectation

$$\begin{aligned} M(x^+)^2 &= M(x(1-h) - h\Delta|x|\eta)^2 = x^2(1-h)^2 + h^2\Delta^2 x^2 U = \\ &= x^2((1-h)^2 + h^2\Delta^2 U) = x^2(1 - 2h + h^2(1+\Delta^2 U)), \end{aligned} \tag{32}$$

where $U = \mathrm{M}(\eta^2)$.

For a uniform distribution η on a unit ball with $n = 2$, it can be calculated as follows:

$$U = \int_0^1 r^2 2\pi r dr \Big/ \int_0^1 2\pi r dr = 1/2. \tag{33}$$

For a uniform distribution η on a unit ball with large n values, $U \approx 1$.

Expression (32), according to (30), determines the change in the function at the iteration. Let us find the optimal step based on (32)

$$(1 - 2h + h^2(1 + \Delta^2 U))' = -2 + 2h(1 + \Delta^2 U) = 0. \tag{34}$$

Therefore, $h^* = 1/(1 + \Delta^2 U)$.

When minimizing multidimensional functions, the optimal step would be significantly smaller due to the spectrum boundaries of the second derivatives matrix. Therefore, we use the step

$$h = qh^* = q/(1 + \Delta^2 U), \quad q < 1. \tag{35}$$

for the evaluation of the indicator $1 - 2h + h^2(1 + \Delta^2 U)$ from (32). Then,

$$\begin{aligned} Q(\Delta) &= 1 - 2h + h^2(1 + \Delta^2 U) = 1 - 2q/(1 + \Delta^2 U) + q^2(1 + \Delta^2 U)/(1 + \Delta^2 U)^2 = . \\ &= 1 - 2q/(1 + \Delta^2 U) + q^2/(1 + \Delta^2 U) \le 1 - q/(1 + \Delta^2 U) \le \exp(-q/(1 + \Delta^2 U)). \end{aligned} \tag{36}$$

Therefore,

$$Q(0) = Q(\Delta)^{1 + \Delta^2 U}. \tag{37}$$

Denote by $N(\Delta)$ the number of iterations of the method to achieve a given accuracy for the function. Then, according to (37), the cost ratio will be as follows:

$$N(\Delta) = (1 + \Delta^2 U) N(0). \tag{38}$$

We use the dependence on the magnitude of the noise (38) to estimate the number of iterations in our problems. Since the estimates for the steepest descent method and the gradient method with the choice of the optimal step for minimizing quadratic functions coincide, we will use the number of iterations of the steepest descent method as $N(0)$.

6. Numerical Experiment

Our experiments were performed on smooth test functions, where the usual steepest descent method converges with the geometric progression rate. The minimum of test functions is uniquely defined. Test functions include functions with curvilinear ravines and functions that differ significantly in properties from quadratic ones. The tests take into account nonlinearities, non-quadraticity and the curvature of ravines with different degrees of problem determination. The calculations are carried out for a number of dimensions.

The local behavior of algorithms in a local region of some nonlinear function where there is a bounded matrix of second derivatives is reflected in tests on a quadratic function at different degrees of conditionality. These data can be extrapolated to non-quadratic functions with an existing matrix of second derivatives, where the number of iterations required is several times greater.

The objectives of the numerical experiment are as follows:

1. Evaluate the efficiency of the proposed algorithms and compare their efficiency with the efficiency of the steepest descent method under conditions without interference.
2. Determine the effects of convergence acceleration in the proposed methods and identify modifications that have accelerated convergence.
3. Study the effect on the convergence rate when applying a gradient of relative interference uniformly distributed on a ball of radius $R(x) = \Delta\|\nabla f(x)\|$.

4. Make estimates of the iteration costs given (29).

Denote the steepest descent method by GR. Among the methods presented above, the following algorithms were used: Algorithm 1 (A1(q)), Algorithm 2 (A2(q)), Algorithm 4 (A4(q, α)), Algorithm 5 (A5(q, $\alpha[a, b]$)).

In all methods, the function and gradient were calculated simultaneously. In step-adaptive algorithms, function calculations are not required. The stopping criterion was $f(x^k) - f^* \le \varepsilon$.

Tables A1–A23 show the number of iterations (the number of gradient calculations) for the methods with step adaptation. For the gradient method (GR), the number of iterations and the number of calculations of the function and gradient used to form the descent direction and one-dimensional minimization are given. We will compare the efficiency of the methods only by the number of iterations. The values of x_0 and ε are given in the description of the corresponding function.

6.1. Rosenbrock Function

The Rosenbrock function has the form:

$$f_R(x) = 100(x_2 - x_1^2)^2 + (x_1 - 1)^2. \quad (39)$$

Its minimum point is $x^* = (1, 1)^T$. Two points were selected as starting points: $x_1^T = (0, 0)$, $x_2^T = (-1.2, 1)$. The stopping criterion was $f(x^k) - f^* \le \varepsilon = 10^{-10}$.

In Tables A1 and A2, the number of iterations of the algorithms A4(q, α)), A5(q, $\alpha[a, b]$) and GR method required to achieve a given accuracy for different α values is presented. One-dimensional search in the steepest descent method was performed based on cubic interpolation using function and gradient information.

Tables A1 and A2 and Figure 2 show calculations for large values of q, where the convergence rate turned out to be higher. But, as we will see below, for functions with a lower degree of conditionality, the costs will be just as high with such parameters, and for small q, these costs will be significantly lower. This problem of equal efficiency for different degrees of conditionality is solved by randomization of the A5 algorithm. The application of this algorithm is equally effective at different levels of conditionality. Also, this algorithm allows obtaining more effective results at a low interference level.

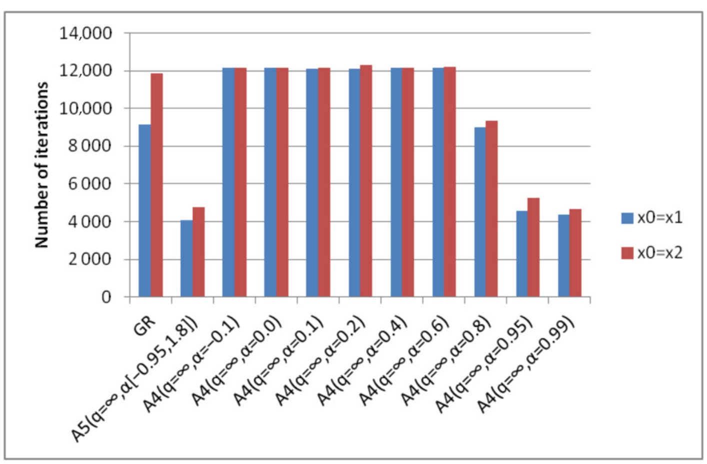


Figure 2. Number of iterations of the algorithms GR, A4(q, α), A5(q, $\alpha[a, b]$) required to achieve a given accuracy for different α values from initial points x_1 and x_2. Function $f_R(x) = 100(x_2 - x_1^2)^2 + (x_1 - 1)^2$.

In Table A3 and Figure 3, the results of minimization with interference are presented. The first column of the table indicates the interference parameter Δ imposed in accordance with (29). It is assumed that the interference is uniformly distributed in the sphere. For some functions, the interference is distributed on the surface of the sphere, which will be specifically discussed. Empty cells here and below mean that the algorithm does not converge as the noise level increases.

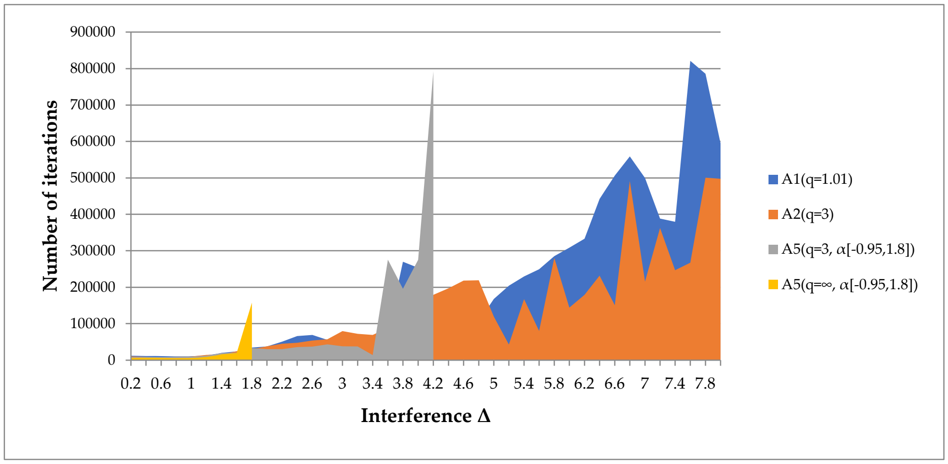


Figure 3. Number of iterations of the algorithms A1(q), A2(q), A4(q, α), A5(q, $\alpha[a, b]$) required to achieve a given accuracy from initial point x_1 with interference. Function $f_R(x) = 100(x_2 - x_1^2)^2 + (x_1 - 1)^2$.

On this function, the algorithms A1(q), A2(q) are approximately the same and can withstand significant interference. Here is an example where the radius of the interference ball is 8 times greater than the gradient norm. In the case of interference, the algorithm A5(q, $\alpha[a, b]$) turned out to be effective, but only for small interference values.

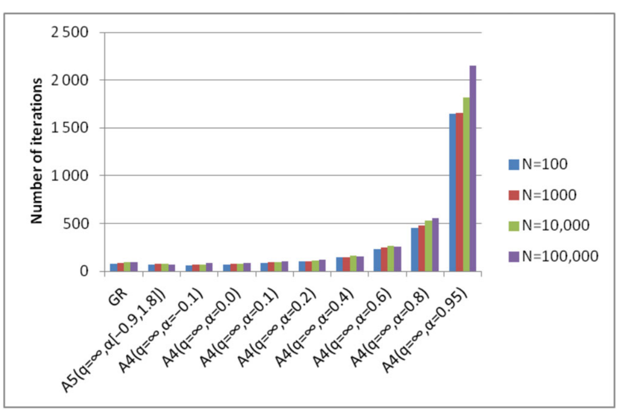
6.2. Quadratic Function

The following quadratic function is tested:

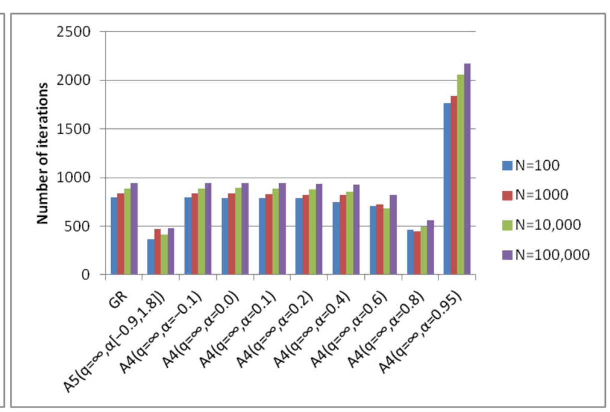
$$f_Q(x,[a\max]) = \frac{1}{2}\sum_{i=1}^{n} a_i\ x_i^2,\ a_i\ = a\max^{\frac{i-1}{n-1}}. \tag{40}$$

The eigenvalues a_i of this function have the boundaries $\lambda_{min} = 1$ and $\lambda_{max} = a_{max}$. The starting point was $x_0^T = (100, 100, \ldots, 100)$. The stopping criterion was $f(x^k) - f^* \le \varepsilon = 10^{-10}$.

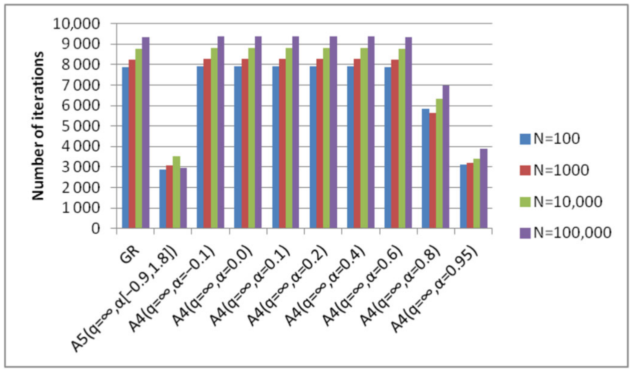
Tables A4–A6 and Figure 4 show the results of function minimization for different degrees of conditionality.



(**a**)



(**b**)



(**c**)

Figure 4. Number of iterations of the algorithms GR, A4(q, α), A5(q, $\alpha[a, b]$) required to achieve a given accuracy for different α values (**a**) Function $f_Q(x) = \frac{1}{2}\sum_{i=1}^{n} 10^{\frac{i-1}{n-1}} x_i^2$; (**b**) Function $f_Q(x) = \frac{1}{2}\sum_{i=1}^{n} 100^{\frac{i-1}{n-1}} x_i^2$; (**c**) $f_Q(x) = \frac{1}{2}\sum_{i=1}^{n} 1000^{\frac{i-1}{n-1}} x_i^2$.

Depending on the conditionality of the problem, the algorithm A4($q = \infty$,α) has good results for different values of α. The algorithm A5($q = \infty$,$\alpha[a, b]$) has equally good results, surpassing the results of the steepest descent method.

Conclusions can be drawn regarding the convergence rate of algorithms:

1. Algorithm A4($q = \infty$, α) achieves good results with different parameters α for different degrees of conditionality. This parameter can only be determined experimentally.
2. Algorithm A5($q = \infty$, $\alpha[a, b]$) achieves good results with fixed parameters of the algorithm for different degrees of conditionality. From this point of view, it can be considered universal.
3. The best versions of algorithm A4($q = \infty$, α) and algorithm A5($q = \infty$, $\alpha[a, b]$) are less expensive in terms of the number of iterations compared to the gradient method.

Table A7 and Figure 5 shows the results of minimizing a quadratic function under gradient interference for dimension $N = 1000$.

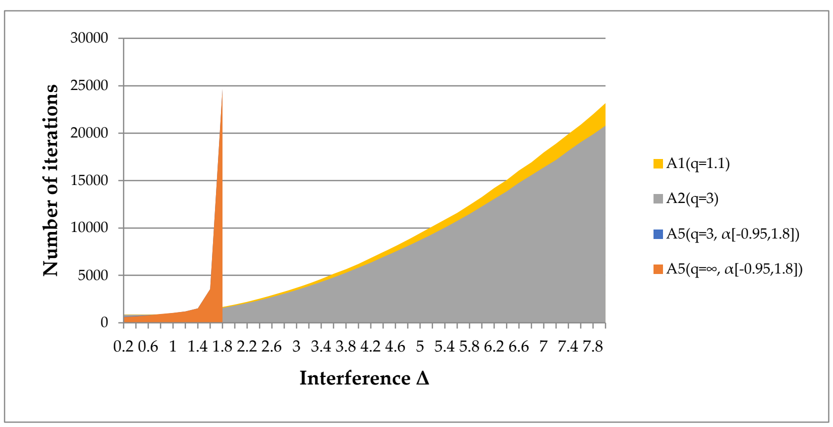


Figure 5. Number of iterations of the algorithms A1(q), A2(q), A4(q, α), A5(q, $\alpha[a, b]$) required to achieve a given accuracy with interference. Function $f_Q(x) = \frac{1}{2}\sum_{i=1}^{n} 100^{\frac{i-1}{n-1}} x_i^2$.

On this function, the algorithms A1(q), A2(q) are approximately the same and can withstand significant interference. Here is an example where the radius of the interference uniformly distributed in the ball is 8 times greater than the gradient norm. In the case of interference, the algorithm A5(q, $\alpha[a, b]$) turned out to be not so effective even for small values of the error level.

6.3. Functions with Ellipsoidal Ravine

The following function has a multidimensional ellipsoidal ravine. Minimization occurs when moving along a curvilinear ravine to the minimum point.

$$f_{EEL}(x,[a\max,b\max])=(1-x_1)^2+a\max\left(1-\sum_{i=1}^{n}x_i^2/b_i\right)^2,\quad b_i=b\max^{\frac{i-1}{n-1}},\tag{41}$$

The starting points were $x_1 = (-1, 0.1, \ldots, 0.1)$, $x_2 = (-1, 2, 3, \ldots, n)$. The stopping criterion was $f(x^k)-f^*\le\varepsilon=10^{-4}$.

Tables A8–A10 and Figures 6 and 7 demonstrate the results of function f_{EEL} minimization.

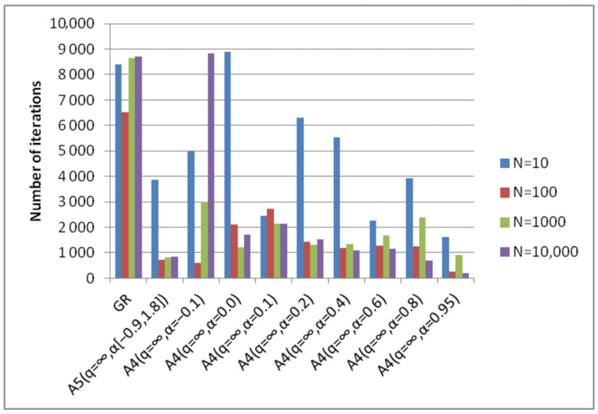


Figure 6. Number of iterations of the algorithms GR, A4(q, α), A5(q, $\alpha[a, b]$) required to achieve a given accuracy for different α values from the initial point x_2. Function $f_{EEL}(x)=(1-x_1)^2+10(1-\sum_{i=1}^{n}\frac{x_i^2}{10^{\frac{i-1}{n-1}}})^2$.

Algorithm A4($q = \infty$, α) has good results for different values of α for different initial points and dimensions of the problem. That is, a preliminary experiment is required to select the optimal parameters. Algorithm A5($q = \infty$, $\alpha[a, b]$) has good results regardless of changes in the dimension and degree of conditionality of the problem. Its results in terms of the number of iterations exceed the results of the steepest descent method.

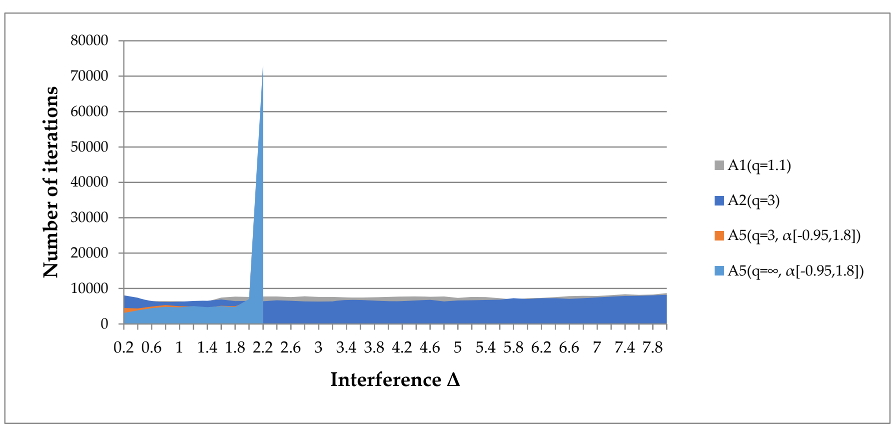


Figure 7. Number of iterations of the algorithms A1(q), A2(q), A4(q, α), A5(q, $\alpha[a, b]$) required to achieve a given accuracy with interference. $N = 1000$, function $f_{EEL}(x)=(1-x_1)^2+10(1-\sum_{i=1}^{n}\frac{x_i^2}{10^{\frac{i-1}{n-1}}})^2$.

On this function, the algorithms A1(q), A2(q) are approximately the same and withstand significant interference. Here, the interference is uniformly distributed in the ball, the radius of which is 8 times greater than the gradient norm. In the case of interference, the algorithm A5(q, $\alpha[a, b]$) shows good results only at a low level of interference.

The convergence rate of the algorithms A1(q), A2(q) depends little on the interference. This is explained by the presence of a ravine, where interference creates the possibility of moving not to the bottom of the ravine, but along it.

In order to make the dependence of the iterations number on the magnitude of interference noticeable, in the next test, we reduced the dimension, and the interference on the gradient was made uniformly distributed on the surface of the ball for different values of the gradient interference parameter (29). Tables A11–A13 and Figures 8 and 9 demonstrate the results.

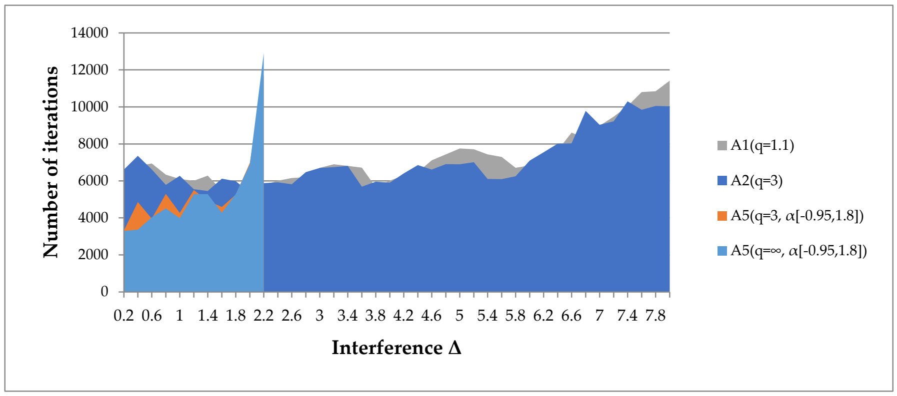


Figure 8. Number of iterations of the algorithms A1(q), A2(q), A4(q, α), A5(q, $\alpha[a, b]$) required to achieve a given accuracy with interference from initial point x_2. N = 100, function $f_{EEL}(x) = (1 - x_1)^2 + 10(1 - \sum_{i=1}^{n} \frac{x_i^2}{10^{\frac{i-1}{n-1}}})^2$.

On this function, the algorithms A1(q), A2(q) are approximately equal in efficiency and can withstand significant interference. Here, the interference is uniformly distributed over the surface of the sphere. In the case of interference, the algorithm A5(q, $\alpha[a, b]$) shows good results at a low level of interference. Here, the dependence of the convergence rate of the algorithms A1(q), A2(q) on the magnitude of interference appears more clearly.

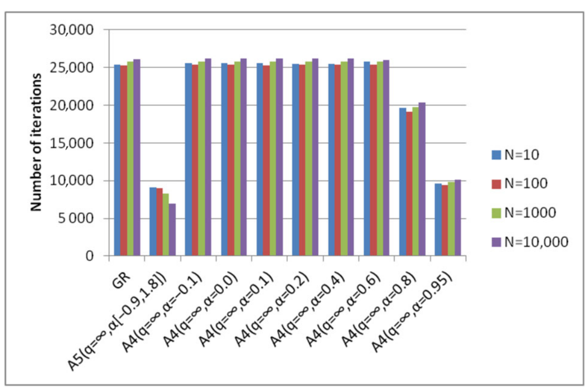


Figure 9. Number of iterations of the algorithms GR, A4(q, α), A5(q, $\alpha[a, b]$) required to achieve a given accuracy for different α values from the initial point x_2. Function $f_{EEL}(x) = (1 - x_1)^2 + 30(1 - \sum_{i=1}^{n} \frac{x_i^2}{10^{\frac{i-1}{n-1}}})^2$.

In Figure 9, the degree of degeneracy of the ravine has increased compared to the previous example. Algorithm A4($q = \infty$, α) has good results for α = 0.95. Algorithm A5($q = \infty$,

$\alpha[a, b]$) has good results, significantly exceeding the results of the steepest descent method, which confirms its universality.

In Figure 10, algorithms A1(q), A2(q) are approximately equal in efficiency at the interference level $\Delta \leq 5$. At higher values, algorithm A2(q) shows better results. Both algorithms withstand significant interference. Here, the interference is uniformly distributed on the surface of the sphere. In the case of interference, algorithm A5(q, $\alpha[a, b]$) shows good results at a low interference level.

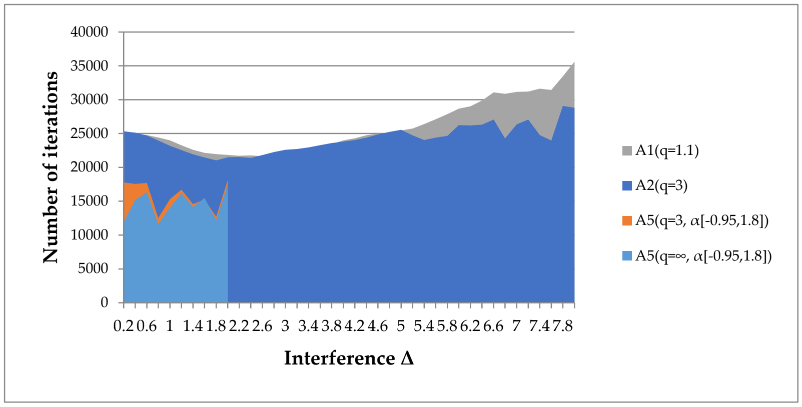


Figure 10. Number of iterations of the algorithms A1(q), A2(q), A4(q, α), A5(q, α[a, b]) required to achieve a given accuracy with interference from the initial point x_2. N = 100, function $f_{EEL}(x) = (1 - x_1)^2 + 30(1 - \sum_{i=1}^{n} \frac{x_i^2}{10^{\frac{i-1}{n-1}}})^2$.

The next function also has a multi-dimensional ellipsoidal ravine.

$$f_{EELX}(x,[a\max]) = (1 - x_1)^2 + a\max\left(1 - \sum_{i=1}^{n} \frac{x_i^{\,2}}{b_i}\right)^2 + \frac{1}{2}\sum_{i=1}^{n} \frac{x_i^{\,2}}{b_i}, \quad b_i = b\max^{\frac{i-1}{n-1}},\ b\max = 10\,. \tag{42}$$

The starting points were $x_1 = (-1, 0.1, \ldots, 0.1)$, $x_2 = (-1, 2, 3, \ldots, n)$. The stopping criterion was $f(x^k) - f^* \leq \varepsilon = 10^{-10}$. Due to the additional term in f_{EELX}, the minimum point ceases to be singular. This allows the gradient method to find the minimum of the function at higher ravine coefficients a_{max} with higher accuracy compared to the function f_{EEL}. Tables A14–A16 and Figures 11 and 12 show the results of function f_{EELX} minimizing.

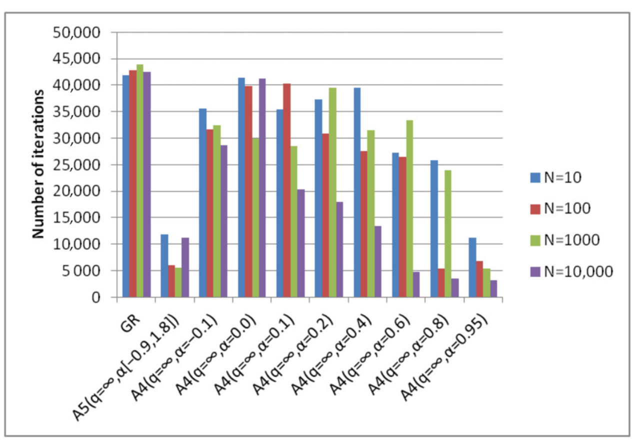


Figure 11. Number of iterations of the algorithms GR, A4(q, α), A5(q, $\alpha[a, b]$) required to achieve a given accuracy for different α values from initial point x_2. Function $f_{EELX}(x) = (1 - x_1)^2 + 100\left(1 - \sum_{i=1}^{n} \frac{x_i^2}{10^{\frac{i-1}{n-1}}}\right)^2 + \frac{1}{2}\sum_{i=1}^{n} \frac{x_i^2}{10^{\frac{i-1}{n-1}}}$.

On this function, the degree of the ravine degeneracy has increased compared to the previous function f_{EEL}. Algorithm A4($q = \infty$, α) has good results for $\alpha = 0.95$. Algorithm A5($q = \infty$, $\alpha[a, b]$) has good results, significantly exceeding the results of the steepest descent method, which, together with the results of minimization of the previous functions, confirms its universality.

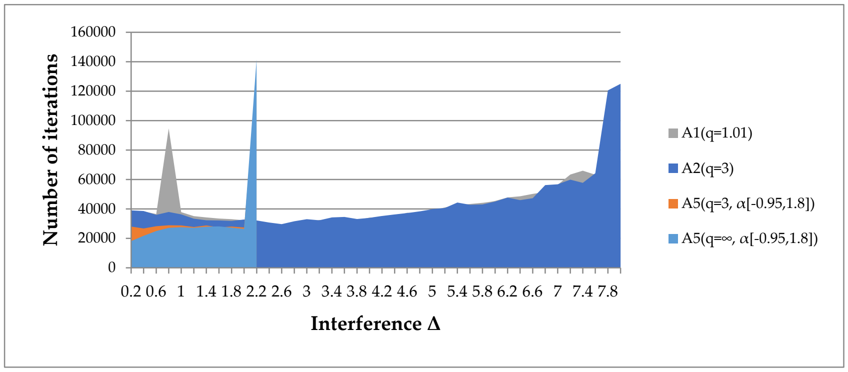


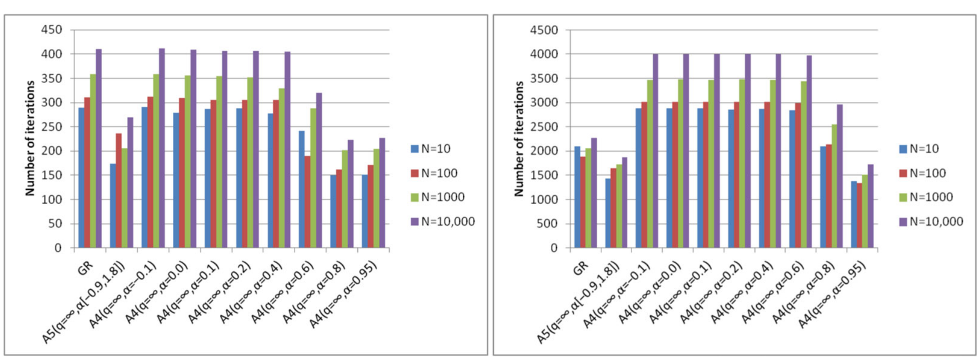
Figure 12. Number of iterations of the algorithms A1(q), A2(q), A4(q, α), A5(q, $\alpha[a, b]$) required to achieve a given accuracy with interference from initial point x_2. $N = 100$, function $f_{EELX}(x) = (1-x_1)^2 + 100\left(1 - \sum_{i=1}^{n} \frac{x_i^2}{10^{\frac{i-1}{n-1}}}\right)^2 + \frac{1}{2}\sum_{i=1}^{n} \frac{x_i^2}{10^{\frac{i-1}{n-1}}}$.

On this function, the A1(q) algorithm slightly outperforms the A2(q) algorithm. This is due to the decrease in the step tuning value in A1($q = 1.01$) from the previous $q = 1.1$. Both algorithms withstand significant interference. Unlike the previous results, here, the interference is uniformly distributed over the sphere. In the case of interference, the A5(q, $\alpha[a, b]$) algorithm shows good results at a low interference level. As the interference level increases, the A5(q, $\alpha[a, b]$) algorithm ceases to converge.

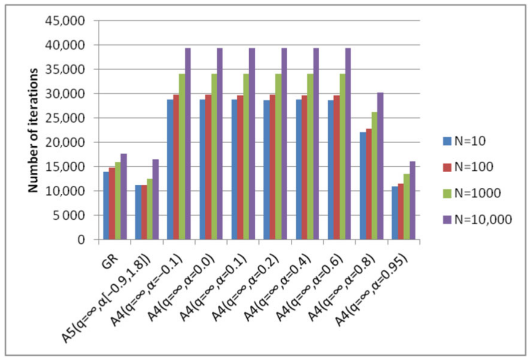
The following function was used to analyze the effect of noise on the gradient components.

$$f_{Q^{\wedge}2}(x,[a\max]) = \left(\sum_{i=1}^{n} a_i\ x_i^2\right)^2, \quad a_i = a\max^{\frac{i-1}{n-1}}, \quad x_0 = (1,1,\ldots,1)\,. \tag{43}$$

The matrix of second derivatives of this function tends to zero as it approaches the minimum. The stopping criterion was $f(x^k) - f^* \le \varepsilon = 10^{-10}$. Tables A17–A20 and Figures 13 and 14 show the results of function $f_{Q^{\wedge}2}$ minimization for different degrees of elongation of the level surfaces.



(**a**) (**b**)



(**c**)

Figure 13. Number of iterations of the algorithms GR, A4(q, α), A5(q, α[a, b]) required to achieve a given accuracy for different α values (**a**) Function $f_{Q^\wedge 2}(x) = \left(\sum_{i=1}^{n} 100^{\frac{i-1}{n-1}} x_i^2\right)^2$; (**b**) Function $f_{Q^\wedge 2}(x) = \left(\sum_{i=1}^{n} 1000^{\frac{i-1}{n-1}} x_i^2\right)^2$; (**c**) $f_{Q^\wedge 2}(x) = \left(\sum_{i=1}^{n} 10000^{\frac{i-1}{n-1}} x_i^2\right)^2$.

Depending on the elongation of the function level surfaces, the algorithm A4($q = \infty$, α) has good results for the same values of $\alpha = 0.95$. The algorithm A5($q = \infty$, α[a, b]) has equally good results, surpassing the results of the steepest descent method.

The following conclusions can be drawn regarding the convergence rate of the algorithms:

1. Algorithm A4($q = \infty$, α) achieves good results at $\alpha = 0.95$ for various degrees of elongation of the level surfaces. This parameter can only be determined experimentally.
2. Algorithm A5($q = \infty$, α[a, b]) achieves good results with fixed algorithm parameters for various degrees of elongation of the level surfaces. From this point of view, it confirms its universality.
3. The best versions of algorithm A4($q = \infty$, α) and algorithm A5($q = \infty$, α[a, b]) are less expensive in terms of the number of iterations compared to the steepest descent method.

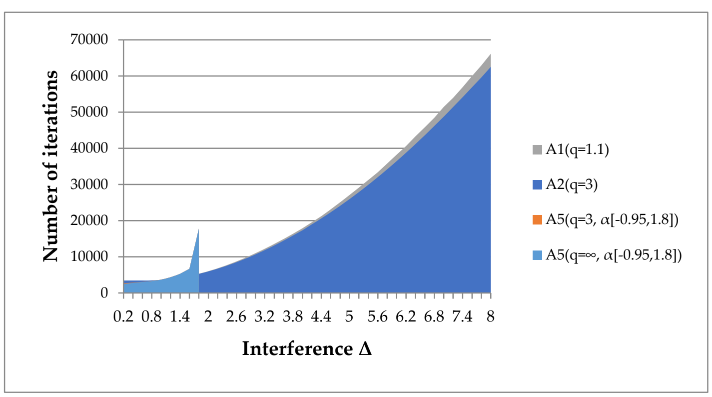


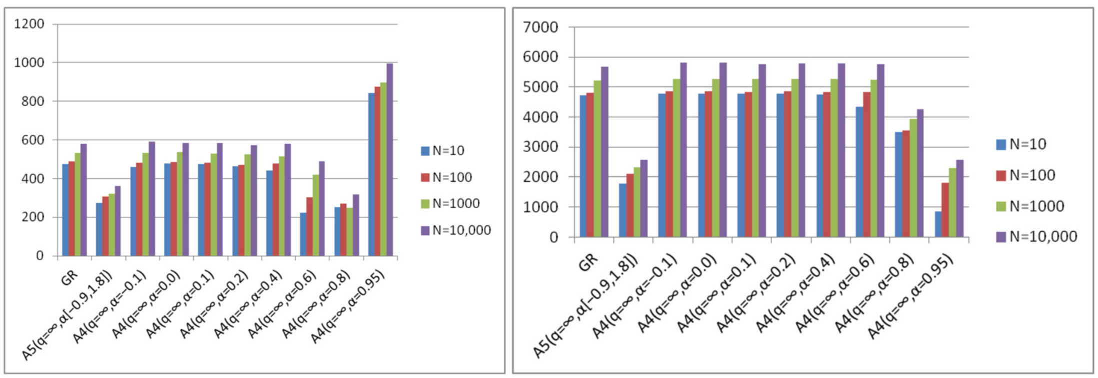
Figure 14. Number of iterations of the algorithms A1(q), A2(q), A4(q, α), A5(q, α[a, b]) required to achieve a given accuracy with interference. $N = 1000$, function $f_{Q^\wedge 2}(x) = \left(\sum_{i=1}^{n} 1000^{\frac{i-1}{n-1}} x_i^2\right)^2$.

On this function, the algorithms A1(q), A2(q) are approximately equivalent in efficiency. In the case of interference, the algorithm A5(q, α[a, b]) turned out to be more efficient only for small values of the noise level. In the case of a strongly elongated curvilinear ravine (at $a_{max} = 10{,}000$), algorithms A1(q), A2(q) with their given parameters failed to obtain a solution at interference level $\Delta > 6$. As experience in testing the algorithms A1(q), A2(q) shows, reducing the step boundary q allows to obtain a solution for large values of interference. At the same time, with small interference, there is some slowdown in the convergence rate.

The next for testing was the Raydan1 function. It is biased to obtain a new function with a zero minimum value:

$$f_R(x,[a\max]) = \sum_{i=1}^{n} \frac{a_i}{10}(\exp(x_i) - x_i - 1),\ \ a_i = a\max^{\frac{i-1}{n-1}},\ \ x_0 = (\ 2,\ 2,\ldots,2) \tag{44}$$

The stopping criterion was $f(x^k) - f^* \le \varepsilon = 10^{-10}$. In this function, the banks of the ravine differ significantly in steepness. The behavior of the gradient method with step adaptation under such conditions is of interest. Tables A21–A23 and Figures 15 and 16 demonstrate the results of function f_R minimization.



(**a**) (**b**)

Figure 15. Number of iterations of the algorithms GR, A4(q, α), A5(q, $\alpha[a, b]$) required to achieve a given accuracy for different α values (**a**) Function $f_{\mathrm{R}}(x) = \sum_{\mathrm{i=1}}^{\mathrm{n}} \frac{100^{\frac{\mathrm{i-1}}{\mathrm{n-1}}}}{10}(e^{x_i} - x_i - 1)$; (**b**) Function $f_{\mathrm{R}}(x) = \sum_{\mathrm{i=1}}^{\mathrm{n}} \frac{1000^{\frac{\mathrm{i-1}}{\mathrm{n-1}}}}{10}(e^{x_i} - x_i - 1)$.

Figure 15 shows the results of function minimization for different degrees of elongation of the level surfaces. Depending on the elongation of the function level surfaces, the A4(q = ∞, α) algorithm has good results for different values of α. The A5(q = ∞, $\alpha[a, b]$) algorithm has equally good results, surpassing the results of the steepest descent method.

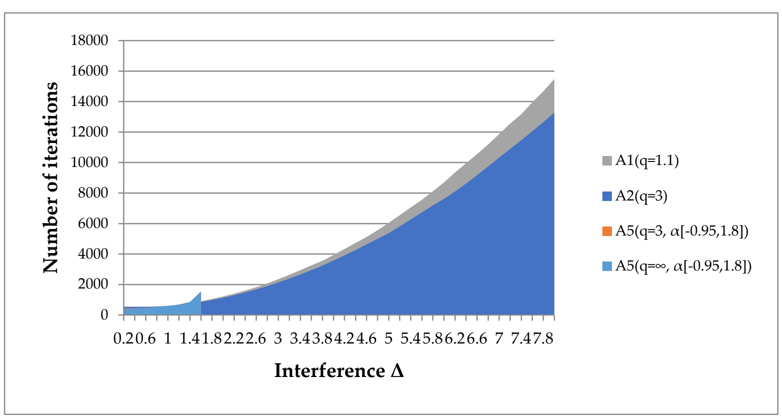


Figure 16. Number of iterations of the algorithms A1(q), A2(q), A4(q, α), A5(q, $\alpha[a, b]$) required to achieve a given accuracy with interference. N = 1000, function $f_{\mathrm{R}}(x) = \sum_{\mathrm{i=1}}^{\mathrm{n}} \frac{100^{\frac{\mathrm{i-1}}{\mathrm{n-1}}}}{10}(e^{x_i} - x_i - 1)$.

On this function, algorithms A1(q), A2(q) are approximately equivalent in efficiency. In the case of interference, algorithm A5(q, $\alpha[a, b]$) turned out to be not so efficient even for small values of the error level.

7. Discussion

To illustrate the convergence of the abovementioned algorithms, we consider the logarithm of the minimization error $f(x^k) - f^*$ against iterations for the quadratic function f_Q (Figure 17).

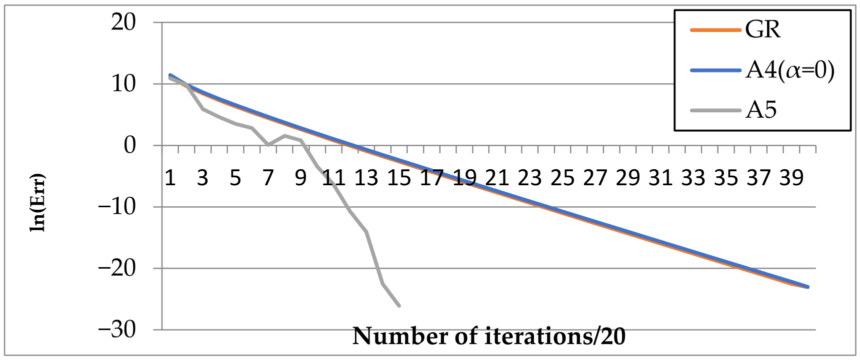


Figure 17. Minimization error in logarithmic scale against number of iterations with 20-iteration step.

As can be seen, a linear convergence rate takes place. Methods GR and A4($q = \infty$, α) are almost equivalent, although the GR method uses a one-dimensional search. Moreover, method A5(q, $\alpha[a, b]$) has a higher linear convergence rate.

In real minimization problems, the costs of methods are proportional to the dimension and are small compared to the time to calculate the function and gradient. Since the main costs are incurred in calculating the function and gradient, the steepest descent method requires at least calculating the function and gradient. The proposed method calculates only one gradient, which means that compared to the steepest descent method, the time is reduced by more than 2 times.

The iteration runtime for the considered methods is presented in Table 1 and Figure 18. The runtime for the A4($q = \infty$, α) and A5(q, $\alpha[a, b]$) methods is equivalent.

Table 1. Runtime in seconds per iteration for the gradient method (GR) and new methods with step adaptation (A4, A5).

Function	$N = 10$		$N = 100$		$N = 1000$		$N = 10{,}000$	
	GR	A4, A5	GR	A4, A5	GR	A4, A5	GR	A4, A5
$f_Q(x, [a_{max} = 1000])$	-	-	2.018×10^{-5}	1.083×10^{-5}	9.341×10^{-5}	3.258×10^{-5}	8.446×10^{-4}	3.236×10^{-4}
$f_{EEL}(x, [a_{max} = 10, b_{max} = 10])$	6.782×10^{-6}	4.400×10^{-6}	1.174×10^{-5}	7.592×10^{-6}	6.195×10^{-5}	3.738×10^{-5}	5.610×10^{-4}	3.409×10^{-4}
$f_{EELX}(x, [a_{max} = 100])$	4.632×10^{-6}	3.006×10^{-6}	9.637×10^{-6}	6.233×10^{-6}	6.971×10^{-5}	4.116×10^{-5}	6.050×10^{-4}	3.973×10^{-4}
$f_{Q\char`^2}(x, [a_{max} = 100])$	5.554×10^{-6}	4.556×10^{-6}	1.026×10^{-5}	1.011×10^{-5}	1.079×10^{-4}	5.624×10^{-5}	1.302×10^{-3}	5.873×10^{-4}
$f_R(x, [a_{max} = 100])$	2.345×10^{-5}	1.426×10^{-5}	6.288×10^{-5}	3.306×10^{-5}	2.910×10^{-4}	1.472×10^{-4}	3.099×10^{-3}	1.640×10^{-3}

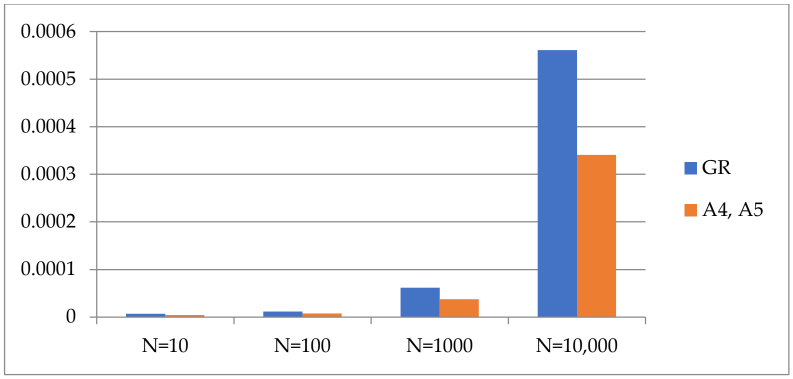


Figure 18. Iteration runtime (s) for the function $f_{EEL}(x, [a_{max} = 10, b_{max} = 10])$. GR is gradient method; A4 and A5 are the new methods with step adaptation.

Let us test the theoretical convergence analysis in practice. For the Rosenbrock function, the cost of the number of iterations of the steepest descent method for different initial points is 9150 and 11,865 iterations. For $\Delta = 3$, taking into account (33) and (38), we obtain $N(\Delta) = (1 + \Delta^2 U)N(0) = (1 + 9/2)N(0) = 5.5N(0)$. For different initial points, we obtain $N(\Delta) =$ 50,325 and $N(\Delta) =$ 65,260, while the actual costs are 59,616 and 43,264.

For $\Delta = 8$, we obtain $N(\Delta) = 33N(0)$. For different initial points $N(\Delta) =$ 30,1950 and $N(\Delta)$ = 351,543, the actual costs are 97,469 and 379,943. For the Rosenbrock function, we obtained a good match between the calculated and real data, although the Rosenbrock function is far from a quadratic function.

For a quadratic function, the cost of the number of iterations of the steepest descent method for different function coefficients is 835 and 8239. For $\Delta = 3$ $N(\Delta) = (1 + \Delta^2 U)N(0) = (1 + 9)N(0) = 10N(0)$. The theoretical values are $N(\Delta)=$ 8350 and $N(\Delta)=$ 82,390, while the actual costs are 3440 and 28,925. For $\Delta = 8$, we obtain $N(\Delta) = 65N(0)$. The theoretical values are $N(\Delta)=$ 54,275 and $N(\Delta)=$ 535,535, while the actual costs are 23,166 and 153,001.

For the quadratic function, the actual results were even better than the estimates. At the same time, the actual data are more consistent with the use of the A5($q = \infty$, α[a, b]) algorithm as $N(0)$ data.

The question arises as to why in many cases deviation from the step selection of the steepest descent method yields better results. In linear algebra, there is a multi-step optimal process for solving systems of linear equations, the parameters of which are calculated based on the boundaries of the matrix spectrum. But it turns out that this process can be implemented as a gradient method, the steps of which are calculated based on Chebyshev polynomials. The steepest descent method descends into a ravine and moves toward a minimum in small steps. In the case of variability of steps, there is a move away from the ravine, which allows moving toward a minimum with large steps. In our case, in the A5($q = \infty$, α[a, b]) algorithm, there is a change in the step, which makes it oscillatory to a certain extent, which facilitates movement with large steps. In the case of interference due to randomness in the direction, the method has large oscillations relative to the ravine, which facilitates faster movement toward the minimum. This is probably why real results for a quadratic function give better results.

Let us summarize the features of the studied algorithms with step adaptation in the gradient method in the cases without interference and with interference.

Without interference:

1. Algorithm A4($q = \infty$, α) achieves the best (minimal) results with different parameters α, which depend on the degree of conditionality of the problem and the choice of the starting point (Figure 19).

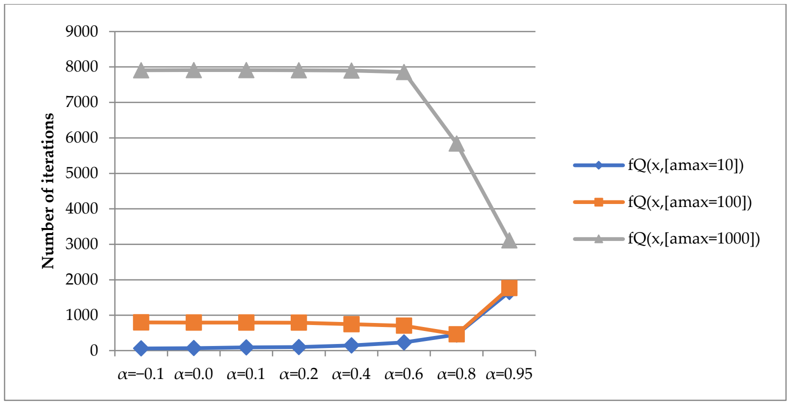


Figure 19. The effect of parameter α on convergence rate in Algorithm A4($q = \infty$, α) for the function f_Q.

2. The best results of algorithm A4($q = \infty$, α) are either comparable or significantly exceed the results of the steepest descent method in the number of iterations.
3. The results of algorithm A5($q = \infty$, $\alpha[a, b]$) with fixed parameters correspond to the results of the optimal algorithm A4($q = \infty$, α) (Figure 20). This means that there is no need to preliminarily choose the parameters for the A4($q = \infty$, α) algorithm. To obtain optimal results one can use the A5($q = \infty$, $\alpha[a, b]$) algorithm, the parameters of which are fixed.

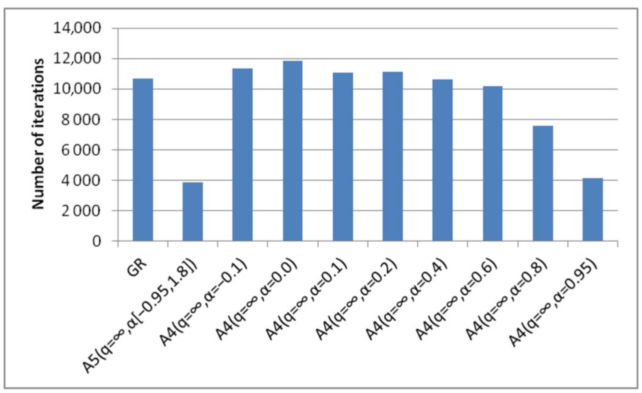
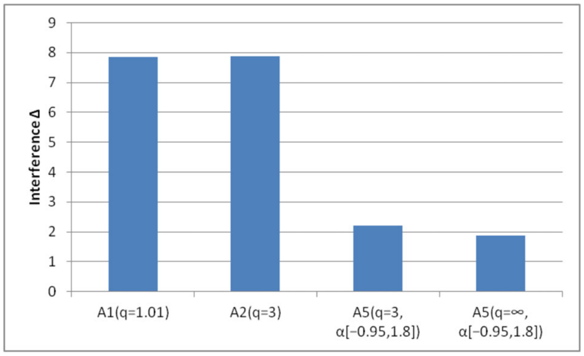


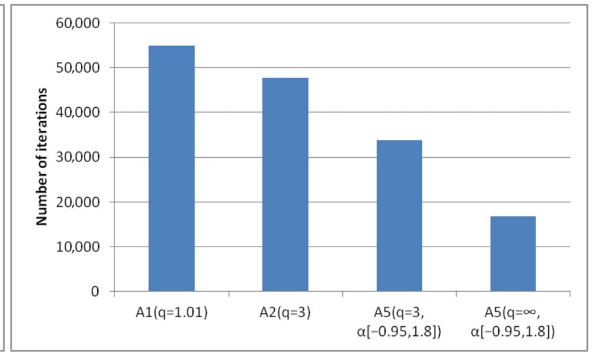
Figure 20. Average number of iterations among all test functions without interference.

With interference:

1. Algorithm A5(q, $\alpha[a, b]$) is applicable only for minor interference. However, its results are not always more significant than the results of algorithms A1(q), A2(q).
2. Algorithms A1(q), A2(q) are applicable for high interference levels. We have given examples where the radius of the interference uniformly distributed in the sphere exceeds the gradient norm by 8 times (Figure 21a).
3. The convergence of algorithms A1(q), A2(q) depends on the restrictions imposed on the parameter q (Figure 21b). For smaller values of the parameter q, the algorithms are efficient at a higher interference level. However, the convergence rate slows down. For smaller values of the boundary q, results can be obtained even with a 10-fold excess of the interference radius over the gradient norm.



(**a**)



(**b**)

Figure 21. Analysis of algorithms A1(q), A2(q), A4(q, α), A5(q, $\alpha[a, b]$) under interference conditions (**a**) Average interference level Δ that algorithm can handle; (**b**) Average number of iterations required to achieve a given accuracy.

8. Conclusions

The paper solves the problem of constructing step adaptation algorithms for a gradient method based on the principle of the steepest descent method. Expanding the step adjustment principle, its formalization and parameterization led to gradient-type methods with incomplete relaxation or over-relaxation. In such methods, only the function gradient needs to be calculated at the iteration. Optimization of the step adaptation algorithm parameters enables us to obtain methods that significantly exceed the steepest descent method in convergence rate.

We present a universal step adjustment algorithm that does not require selecting optimal parameters, and its convergence rate corresponds to algorithms with optimization of the step adaptation algorithm parameters. The advantage of the proposed method is its operability under interference conditions. Our paper presents examples of solving test problems in which the interference level is in the form of a uniformly distributed vector in a ball whose radius is 8 times greater than the gradient norm.

Author Contributions: Conceptualization, V.K.; methodology, V.K., E.T. and S.G.; software, V.K.; validation, L.K., E.T., I.R. and S.G.; formal analysis, L.K. and S.G.; investigation, V.K.; resources, L.K.; data curation, S.G.; writing—original draft preparation, V.K. and S.G.; writing—review and editing, E.T., I.R. and L.K.; visualization, V.K. and E.T.; supervision, V.K. and L.K.; project administration, L.K.; funding acquisition, L.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Ministry of Science and Higher Education of the Russian Federation, project no. FEFE-2023-0004.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Appendix A

Table A1. Number of iterations for Rosenbrock function minimization without interference from initial point x_1.

N	*GR* *	*A5*($q = \infty$, $\alpha[a, b]$)	*A4*($q = \infty$, α)								
		$\alpha \in [-0.95, 1.8]$	$\alpha = -0.1$	$\alpha = 0.0$	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.4$	$\alpha = 0.6$	$\alpha = 0.8$	$\alpha = 0.95$	$\alpha = 0.99$
2	9150 (22,788)	4077	12,152	12,149	12,122	12,105	12,144	12,134	9008	4553	4391

* The number of function and gradient calculations is given in parentheses.

Table A2. Number of iterations for Rosenbrock function minimization without interference from initial point x_2.

N	*GR* *	*A5*($q = \infty$, $\alpha[a, b]$)	*A4*($q = \infty$, α)								
		$\alpha \in [-0.95, 1.8]$	$\alpha = -0.1$	$\alpha = 0.0$	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.4$	$\alpha = 0.6$	$\alpha = 0.8$	$\alpha = 0.95$	$\alpha = 0.99$
2	11,865 (29,610)	4761	12,159	12,133	12,149	12,294	12,150	12,196	9340	5263	4657

* The number of function and gradient calculations is given in parentheses.

Table A3. Number of iterations for Rosenbrock function minimization with gradient interference Δ from initial points x_1, x_2.

Δ	x_1				x_2			
	A1(q)	*A2*(q)	*A5*(q, $\alpha[a, b]$)	*A5*(q, $\alpha[a, b]$)	*A1*(q)	*A2*(q)	*A5*(q, $\alpha[a, b]$)	*A5*(q, $\alpha[a, b]$)
	$\alpha = 0.0$	$\alpha = 0.0$	$\alpha \in [-0.95, 1.8]$	$\alpha \in [-0.95, 1.8]$	$\alpha = 0.0$	$\alpha = 0.0$	$\alpha \in [-0.95, 1.8]$	$\alpha \in [-0.95, 1.8]$
	$q = 1.01$	$q = 3$	$q = 3$	$q = \infty$	$q = 1.01$	$q = 3$	$q = 3$	$q = \infty$
0.2	12,184	10,722	8567	5906	12,234	10,939	8637	6147
0.4	11,465	8439	8402	6466	11,465	8541	8635	5847
0.6	11,023	7676	8284	6561	11,092	8458	8694	6696
0.8	11,120	8510	8721	5382	10,782	9086	7935	7145
1	10,491	10,111	9446	6121	11,269	9635	12,340	7319
1.2	13,315	14,285	12,368	8728	10,686	7985	16,987	3271
1.4	20,098	15,445	19,649	16,118	18,720	18,911	34,700	9366
1.6	24,043	24,570	20,621	19,482	28,600	24,022	22,241	25,043
1.8	34,352	29,378	33,314	157,679	44,400	32,019	41,962	
2	37,942	38,146	30,051		31,477	37,510	29,335	
2.2	50,690	45,354	29,812		51,192	42,941	28,450	
2.4	66,219	47,895	35,579		89,059	51,962	35,754	
2.6.	69,348	53,748	36,969		77,062	60,003	86,217	
2.8	55,604	57,246	43,107		84,679	65,490	30,715	
3	59,616	79,305	38,298		29,050	43,264	65,511	
3.2	60,337	72,013	37,955		57,805	85,334	157,592	
3.4	64,974	69,090	14,077		64,798	94,479	122,119	
3.6	78,793	88,460	275,656		110,826	90,339	389,925	
3.8	269,995	97,980	195,847		222,006	117,191	291,453	
4	253,321	101,563	275,651		147,051	130,206		
4.2	94,456	179,195	792,694		47,864	116,564		
4.4	24,622	197,242			21,570	128,049		
4.6	86,633	218,273			64,747	124,935		
4.8	106,416	219,263			110,989	105,653		
5	167,580	119,307			168,840	146,901		
5.2	204,223	42,896			196,347	214,080		
5.4	229,839	167,171			244,947	135,973		
5.6	249,466	80,196			282,396	164,611		
5.8	285,200	280,227			285,186	272,935		

6	308,540	144,408		429,597	461,874
6.2	333,173	179,384		334,031	440,997
6.4	442,650	231,839		450,688	346,296
6.6	506,562	150,984		562,969	335,605
6.8	558,934	491,455		624,308	371,494
7	499,818	215,790		625,358	250,578
7.2	388,358	362,532		502,964	120,357
7.4	379,735	247,071		568,432	730,441
7.6	821,177	267,345		651,425	257,868
7.8	785,639	500,870		386,720	651,424
8	594,816	497,469		379,943	388,342

Table A4. Number of iterations for function $f_Q(x, [a_{max} = 10])$ minimization without interference from initial point x_0.

N	***GR****	***A5*(*q* = ∞, *α*[*a*, *b*])**	***A4*(*q* = ∞, *α*)**							
		***α* ∈ [−0.9, 1.8]**	***α* = −0.1**	***α* = 0.0**	***α* = 0.1**	***α* = 0.2**	***α* = 0.4**	***α* = 0.6**	***α* = 0.8**	***α* = 0.95**
100	81 (186)	71	60	71	90	102	147	230	451	1652
1000	86 (200)	82	68	74	95	105	146	247	478	1656
10,000	91 (214)	78	73	78	97	111	162	265	526	1822
100,000	97 (230)	72	84	89	103	122	154	257	556	2147

* The number of function and gradient calculations is given in parentheses.

Table A5. Number of iterations for function $f_Q(x, [a_{max} = 100])$ minimization without interference from initial point x_0.

N	***GR****	***A5*(*q* = ∞, *α*[*a*, *b*])**	***A4*(*q* = ∞, *α*)**							
		***α* ∈ [−0.9, 1.8]**	***α* = −0.1**	***α* = 0.0**	***α* = 0.1**	***α* = 0.2**	***α* = 0.4**	***α* = 0.6**	***α* = 0.8**	***α* = 0.95**
100	793 (1618)	364	795	791	792	788	746	704	461	1764
1000	835 (1708)	468	839	837	833	823	819	725	445	1836
10,000	888 (1819)	413	888	891	886	881	855	686	494	2061
100,000	944 (1936)	476	947	943	947	937	926	822	561	2170

* The number of function and gradient calculations is given in parentheses.

Table A6. Number of iterations for function $f_Q(x, [a_{max} = 1000])$ minimization without interference from initial point x_0.

N	***GR****	***A5*(*q* = ∞, *α*[*a*, *b*])**	***A4*(*q* = ∞, *α*)**							
		***α* ∈ [−0.9, 1.8]**	***α* = −0.1**	***α* = 0.0**	***α* = 0.1**	***α* = 0.2**	***α* = 0.4**	***α* = 0.6**	***α* = 0.8**	***α* = 0.95**
100	7869 (15,768)	2874	7908	7914	7912	7910	7900	7858	5841	3110
1000	8239 (16,514)	3079	8287	8277	8278	8275	8264	8250	5636	3202
10,000	8770 (17,582)	3532	8820	8821	8810	8813	8806	8785	6342	3391
100,000	9324 (18,697)	2966	9376	9371	9376	9370	9362	9350	7000	3881

* The number of function and gradient calculations is given in parentheses.

Table A7. Number of iterations for f_Q function minimization with gradient interference Δ from initial point x_0, $N = 1000$.

Δ	***f*Q(*x*, [*a*max = 100])**				***f*Q(*x*, [*a*max = 1000])**			
	***A1*(*q*) *α* = 0.0 *q* = 1.1**	***A2*(*q*) *α* = 0.0 *q* = 3**	***A5*(*q*, *α*[*a*, *b*]) *α* ∈ [−0.95, 1.8] *q* = 3**	***A5*(*q*, *α*[*a*, *b*]) *α* ∈ [−0.95, 1.8] *q* = ∞**	***A1*(*q*) *α* = 0.0 *q* = 1.1**	***A2*(*q*) *α* = 0.0 *q* = 3**	***A5*(*q*, *α*[*a*, *b*]) *α* ∈ [−0.95, 1.8] *q* = 3**	***A5*(*q*, *α*[*a*, *b*]) *α* ∈ [−0.95, 1.8] *q* = ∞**
0.2	873	843	647	604	8234	8251	6858	6618
0.4	874	843	703	678	8227	8238	7320	7204

0.6	876	846	778	777	8252	8278	8005	8029
0.8	897	868	898	920	8499	8553	8964	9028
1	968	935	1030	1037	9085	9115	9982	10,030
1.2	1090	1043	1191	1196	9989	9929	11,219	11,241
1.4	1245	1181	1513	1518	11,181	11,020	20,713	20,795
1.6	1446	1358	3555	3565	12,621	12,498		
1.8	1671	1580	24,613	24,677	14,340	14,253		
2	1934	1822			16,243	16,155		
2.2	2217	2089			18,380	18,282		
2.4	2541	2395			20,647	20,540		
2.6.	2897	2714			23,195	22,954		
2.8	3287	3063			25,941	25,606		
3	3695	3440			28,925	28,431		
3.2	4142	3860			32,035	31,511		
3.4	4632	4307			35,494	34,834		
3.6	5164	4779			39,093	38,320		
3.8	5663	5283			42,881	41,972		
4	6216	5810			46,834	45,830		
4.2	6828	6345			50,922	49,877		
4.4	7445	6921			55,372	54,136		
4.6	8095	7498			59,873	58,421		
4.8	8748	8090			64,598	62,862		
5	9454	8715			69,391	67,532		
5.2	10,139	9365			74,271	72,317		
5.4	10,897	10,018			79,538	77,524		
5.6	11,600	10,752			85,077	82,805		
5.8	12,428	11,474			90,453	88,153		
6	13,268	12,264			96,130	93,746		
6.2	14,206	13,060			102,006	99,503		
6.4	15,051	13,850			108,469	105,460		
6.6	16,051	14,769			114,860	111,749		
6.8	16,927	15,580			121,053	117,931		
7	17,956	16,385			119,322	124,245		
7.2	18,906	17,191			126,186	130,862		
7.4	19,912	18,136			132,999	137,335		
7.6	20,884	19,041			139,836	137,335		
7.8	21,982	19,885			147,162	143,897		
8	23,166	20,781			153,001	150,746		

Table A8. Number of iterations for function $f_{EEL}(x, [a_{max} = 10, b_{max} = 10])$ minimization without interference from initial point x_2.

N	GR *	$A5(q = \infty, \alpha[a, b])$	$A4(q = \infty, \alpha)$							
		$\alpha \in [-0.9, 1.8]$	$\alpha = -0.1$	$\alpha = 0.0$	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.4$	$\alpha = 0.6$	$\alpha = 0.8$	$\alpha = 0.95$
10	8393 (20,648)	3861	4969	8897	2442	6305	5520	2275	3919	1628
100	6505 (16,234)	724	584	2105	2712	1440	1186	1271	1256	250
1000	8661 (20,498)	822	3005	1215	2131	1292	1326	1676	2386	917
10,000	8715 (21,186)	847	8826	1712	2153	1531	1081	1161	681	207

* The number of function and gradient calculations is given in parentheses.

Table A9. Number of iterations for function $f_{EEL}(x, [a_{max} = 10, b_{max} = 10])$ minimization without interference from initial point x_1.

N	*GR* *	*A5*($q = \infty$, $\alpha[a, b]$) $\alpha \in$ [−0.9, 1.8]	*A4*($q = \infty$, α) α = −0.1	α = 0.0	α = 0.1	α = 0.2	α = 0.4	α = 0.6	α = 0.8	α = 0.95
10	2058 (4547)	872	2022	2056	2079	2027	2047	2041	1516	881
100	1547 (3817)	1672	3667	3634	3579	3584	3607	3666	2896	1555
1000	5592 (13,929)	1901	5838	5879	5898	5829	5937	6085	4769	2873
10,000	2339 (5807)	1898	7323	7316	6354	7463	1728	5389	3923	3556

* The number of function and gradient calculations is given in parentheses.

Table A10. Number of iterations for $f_{EEL}(x, [a_{max} = 10, b_{max} = 10])$ function minimization with gradient interference Δ from initial point x_2, $N = 1000$.

Δ	$f_{EEL}(x, [a_{max} = 10, b_{max} = 10])$ *A1*(q) α = 0.0 q = 1.1	*A2*(q) α = 0.0 q = 3	*A5*(q, $\alpha[a, b]$) $\alpha \in$ [−0.95, 1.8] q = 3	*A5*(q, $\alpha[a, b]$) $\alpha \in$ [−0.95, 1.8] $q = \infty$
0.2	6428	8070	4546	3120
0.4	6138	7443	4364	3870
0.6	6389	6497	4927	4468
0.8	6468	6175	5354	4822
1	6390	6298	4983	4723
1.2	6275	6507	4966	5030
1.4	6367	6547	4619	4746
1.6	7473	6898	5136	5063
1.8	7734	6466	5004	4809
2	7625	6496	6478	7026
2.2	7764	6417	71,359	73,154
2.4	7763	6709		
2.6.	7598	6530		
2.8	7802	6369		
3	7649	6341		
3.2	7625	6372		
3.4	7515	6837		
3.6	7541	6749		
3.8	7505	6536		
4	7586	6476		
4.2	7749	6443		
4.4	7826	6638		
4.6	7661	6821		
4.8	7777	6378		
5	7393	6674		
5.2	7655	6711		
5.4	7599	6783		
5.6	7265	6873		
5.8	6927	7237		
6	7256	7058		
6.2	7404	7237		
6.4	7605	7276		
6.6	7868	7101		
6.8	7 947	7252		
7	7876	7508		
7.2	8147	7722		

7.4	8420	7904
7.6	8237	7924
7.8	8328	8067
8	8712	8239

Table A11. Number of iterations for $f_{EEL}(x, [a_{max} = 10, b_{max} = 10])$ function minimization with gradient interference Δ, uniformly distributed on the surface of the ball, from initial points x_1, x_2, $N = 100$.

Δ	x_2 A1(q) α = 0.0 q = 1.1	A2(q) α = 0.0 q = 3	A5(q, α[a, b]) α ∈ [−0.95, 1.8] q = 3	A5(q, α[a, b]) α ∈ [−0.95, 1.8] q = ∞	x_1 A1(q) α = 0.0 q = 1.1	A2(q) α = 0.0 q = 3	A5(q, α[a, b]) α ∈ [−0.95, 1.8] q = 3	A5(q, α[a, b]) α ∈ [−0.95, 1.8] q = ∞
0.2	6463	6619	3324	3296	3500	3558	2566	1969
0.4	6811	7356	4862	3369	3469	3502	2670	2307
0.6	6945	6627	3963	4032	3452	3478	2943	2741
0.8	6338	5778	5297	4512	3450	3461	3010	2919
1	6102	6272	4268	3980	3428	3408	3191	3102
1.2	6009	5548	5503	5287	3550	3399	3243	3284
1.4	6286	5458	4914	5281	3433	3428	3529	3693
1.6	5480	6119	4593	4288	3538	3395	3396	3436
1.8	5462	5979	5262	5293	3551	3487	3363	3314
2	5508	5375	6975	6902	3769	3568	3609	4003
2.2	5780	5876	11,733	12,909	3787	3698	5778	5922
2.4	6006	5926			3966	3810		
2.6.	6152	5823			3977	3844		
2.8	6199	6479			4625	3959		
3	6703	6696			4381	4072		
3.2	6899	6739			4835	4277		
3.4	6809	6813			4993	4440		
3.6	6715	5691			5177	4457		
3.8	5665	5962			5316	4619		
4	6015	5901			5320	4742		
4.2	6288	6401			5498	4669		
4.4	6468	6850			5520	4886		
4.6	7110	6609			5743	4802		
4.8	7440	6907			5827	5374		
5	7751	6903			5771	5469		
5.2	7715	7009			6210	5242		
5.4	7432	6108			6284	5316		
5.6	7293	6097			5791	5582		
5.8	6703	6248			5923	5715		
6	6833	7096			6040	5433		
6.2	7098	7544			6314	6100		
6.4	7660	8011			6703	6387		
6.6	8610	8026			6894	5967		
6.8	8306	9781			6684	7017		
7	8984	9033			6868	6968		
7.2	9472	9227			7161	8292		
7.4	10,039	10,303			7202	6709		
7.6	10,803	9840			7743	6972		
7.8	10,844	10,057			7923	9873		
8	11,414	10,028			8267	9668		

Table A12. Number of iterations for function $f_{EEL}(x, [a_{max} = 30, b_{max} = 10])$ minimization without interference from initial point x_2.

N	*GR* *	*A5*($q = \infty$, $\alpha[a, b]$) $\alpha \in [-0.9, 1.8]$	*A4*($q = \infty$, α) $\alpha = -0.1$	$\alpha = 0.0$	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.4$	$\alpha = 0.6$	$\alpha = 0.8$	$\alpha = 0.95$
10	25,334 (61,902)	9128	25,586	25,539	25,591	25,481	25,492	25,820	19,615	9624
100	25,237 (61,712)	9045	25,346	25,329	25,305	25,376	25,371	25,355	19,130	9430
1000	25,742 (62,524)	8302	25,801	25,819	25,768	25,824	25,826	25,823	19,788	9835
10,000	26,125 (64,321)	6972	26,209	261,99	26,173	26,201	26,210	25,958	20,315	10,157

* The number of function and gradient calculations is given in parentheses.

Table A13. Number of iterations for $f_{EEL}(x, [a_{max} = 30, b_{max} = 10])$ function minimization with gradient interference Δ, uniformly distributed on the surface of the ball, from initial point x_2, $N = 100$.

Δ	$f_{EEL}(x, [a_{max} = 30, b_{max} = 10])$ *A1*(q) $\alpha = 0.0$ $q = 1.1$	*A2*(q) $\alpha = 0.0$ $q = 3$	*A5*(q, $\alpha[a, b]$) $\alpha \in [-0.95, 1.8]$ $q = 3$	*A5*(q, $\alpha[a, b]$) $\alpha \in [-0.95, 1.8]$ $q = \infty$
0.2	25,099	25,341	17,787	11,910
0.4	24,972	25,119	17,560	15,195
0.6	24,759	24,693	17,692	16,312
0.8	24,413	23,964	12,537	11,609
1	23,998	23,179	15,310	14,063
1.2	23,250	22,544	16,715	16,184
1.4	22,614	21,935	14,574	14,142
1.6	22,135	21,448	15,230	15,481
1.8	21,966	21,038	12,699	12,213
2	21,828	21,469	18,127	17,482
2.2	21,723	21,538		
2.4	21,771	21,396		
2.6.	21,661	21,814		
2.8	22,065	22,283		
3	22,205	22,580		
3.2	22,320	22,711		
3.4	22,289	22,945		
3.6	22,716	23,266		
3.8	23,446	23,596		
4	23,977	23,819		
4.2	24,292	24,022		
4.4	24,747	24,399		
4.6	24,994	24,852		
4.8	24,781	25,264		
5	25,453	25,549		
5.2	25,746	24,729		
5.4	26,405	24,031		
5.6	27,098	24,399		
5.8	27,870	24,650		
6	28,674	26,238		
6.2	29,028	26,200		
6.4	29,874	26,299		
6.6	31,088	27,064		
6.8	30,879	24,282		
7	31,144	26,347		
7.2	31,194	27,062		

7.4	31,607	24,769
7.6	31,432	23,974
7.8	33,436	29,079
8	35,627	28,815

Table A14. Number of iterations for function $f_{EELX}(x, [a_{max} = 100])$ minimization without interference from initial point x_2.

N	***GR*** *	***A*5(*q* = ∞, *α*[*a*, *b*])**	***A*4(*q* = ∞, *α*)**							
		***α* ∈ [−0.9, 1.8]**	***α* = −0.1**	***α* = 0.0**	***α* = 0.1**	***α* = 0.2**	***α* = 0.4**	***α* = 0.6**	***α* = 0.8**	***α* = 0.95**
10	41,854 (73,933)	11,819	35,579	41,396	35,471	37,251	39,603	27,322	25,806	11,276
100	42,888 (74,275)	5968	31,693	39,836	40,243	30,840	27,556	26,535	5382	6887
1000	43,867 (78,317)	5636	32,430	30,004	28,578	39,560	31,555	33,324	23,921	5343
10,000	42,537 (75,562)	11,292	28,718	41,286	20,289	18,044	13,411	4721	3481	3134

* The number of function and gradient calculations is given in parentheses.

Table A15. Number of iterations for function $f_{EELX}(x, [a_{max} = 100])$ minimization without interference from initial point x_1.

N	***GR*** *	***A*5(*q* = ∞, *α*[*a*, *b*])**	***A*4(*q* = ∞, *α*)**							
		***α* ∈ [−0.9, 1.8]**	***α* = −0.1**	***α* = 0.0**	***α* = 0.1**	***α* = 0.2**	***α* = 0.4**	***α* = 0.6**	***α* = 0.8**	***α* = 0.95**
10	19,327 (48,234)	14,202	32,401	32,412	32,409	32,402	32,402	32,387	24,936	13,102
100	18,936 (47,183)	14,427	33,407	33,410	33,408	33,318	33,398	33,396	25,724	13,528
1000	34,541 (86,318)	15,549	35,864	35,917	35,944	35,727	36,008	36,288	28,274	15,046
10,000	25,837 (64,455)	15,542	37,922	37,878	36,342	38,117	38,931	37,235	23,886	15,969

* The number of function and gradient calculations is given in parentheses.

Table A16. Number of iterations for $f_{EELX}(x, [a_{max} = 100])$ function minimization with gradient interference Δ from initial points x_1, x_2, $N = 100$.

Δ	***x*₂**				***x*₁**			
	***A*1(*q*)** ***α* = 0.0** ***q* = 1.01**	***A*2(*q*)** ***α* = 0.0** ***q* = 3**	***A*5(*q*, *α*[*a*, *b*])** ***α* ∈ [−0.95, 1.8]** ***q* = 3**	***A*5(*q*, *α*[*a*, *b*])** ***α* ∈ [−0.95, 1.8]** ***q* = ∞**	***A*1(*q*)** ***α* = 0.0** ***q* = 1.01**	***A*2(*q*)** ***α* = 0.0** ***q* = 3**	***A*5(*q*, *α*[*a*, *b*])** ***α* ∈ [−0.95, 1.8]** ***q* = 3**	***A*5(*q*, *α*[*a*, *b*])** ***α* ∈ [−0.95, 1.8]** ***q* = ∞**
0.2	37,261	39,007	28,008	18,091	33,231	33,419	23,600	16,216
0.4	38,355	38,541	26,724	21,720	33,299	33,231	24,268	19,993
0.6	36,569	36,039	28,297	25,003	33,241	32,905	24,610	22,325
0.8	94,629	38,007	28,867	27,261	32,891	32,579	24,709	23,412
1	37,825	36,312	28,740	27,674	32,374	31,916	25,320	24,206
1.2	35,119	33,383	27,729	27,199	31,789	30,820	25,689	24,913
1.4	34,335	32,358	28,724	27,925	31,176	30,068	24,969	24,793
1.6	33,424	32,123	27,567	28,159	30,461	29,453	24,479	24,254
1.8	33,018	31,864	28,051	27,285	29,711	29,106	24,255	24,720
2	32,288	32,887	27,469	26,362	29,359	28,734	28,527	28,601
2.2	31,293	32,121	127,756	141,449	28,950	28,555	133,978	159,208
2.4	29,560	30,786			28,804	28,586		
2.6.	29,053	29,665			28,902	28,272		
2.8	29,282	31,642			29,397	28,963		
3	30,602	33,052			29,036	29,448		
3.2	32,505	32,337			29,585	29,523		
3.4	32,430	34,266			30,025	30,582		
3.6	32,912	34,555			30,618	30,898		
3.8	32,362	33,138			30,836	31,461		
4	34,169	33,837			31,586	32,495		

4.2	34,408	35,353	32,236	33,122
4.4	35,033	36,173	32,768	34,311
4.6	37,325	37,278	34,021	34,872
4.8	37,019	38,411	34,451	36,339
5	38,425	39,939	34,712	37,565
5.2	40,954	40,611	36,631	38,074
5.4	42,431	44,346	36,522	39,336
5.6	43,359	42,846	37,023	41,404
5.8	44,149	43,068	39,175	41,123
6	45,440	45,009	40,235	42,563
6.2	47,913	47,721	41,789	43,140
6.4	48,651	45,961	42,449	42,268
6.6	50,238	47,260	44,234	45,318
6.8	51,542	56,151	45,820	45,302
7	56,361	56,762	49,367	46,206
7.2	63,478	59,872	52,129	47,642
7.4	65,989	57,709	55,812	50,892
7.6	63,248	64,288	59,237	60,767
7.8	70,764	12,0629	69,242	58,808
8	72,668	124,948	66,135	54,755

Table A17. Number of iterations for function $f_{Q^2}(x, [a_{max} = 100])$ minimization without interference from initial point x_0.

N	*GR* *	***A*5(*q* = ∞, *α*[*a*, *b*])**	***A*4(*q* = ∞, *α*)**							
		***α* ∈ [−0.9, 1.8]**	***α* = −0.1**	***α* = 0.0**	***α* = 0.1**	***α* = 0.2**	***α* = 0.4**	***α* = 0.6**	***α* = 0.8**	***α* = 0.95**
10	289 (591)	174	291	279	287	288	278	241	150	151
100	311 (635)	236	312	309	306	305	305	190	162	171
1000	358 (729)	206	359	356	354	352	329	288	202	204
10,000	411 (835)	269	412	409	407	406	405	320	223	227

* The number of function and gradient calculations is given in parentheses.

Table A18. Number of iterations for function $f_{Q^2}(x, [a_{max} = 1000])$ minimization without interference from initial point x_0.

N	*GR* *	***A*5(*q* = ∞, *α*[*a*, *b*])**	***A*4(*q* = ∞, *α*)**							
		***α* ∈ [−0.9, 1.8]**	***α* = −0.1**	***α* = 0.0**	***α* = 0.1**	***α* = 0.2**	***α* = 0.4**	***α* = 0.6**	***α* = 0.8**	***α* = 0.95**
10	2104 (4789)	1431	2883	2882	2879	2862	2868	2845	2095	1382
100	1878 (4363)	1643	3016	3021	3017	3019	3010	2993	2142	1335
1000	2059 (4811)	1719	3466	3477	3474	3475	3472	3444	2552	1515
10,000	2267 (5331)	1877	3998	4006	4003	4004	4004	3977	2966	1727

* The number of function and gradient calculations is given in parentheses.

Table A19. Number of iterations for function $f_{Q^2}(x, [a_{max} = 10000])$ minimization without interference from initial point x_0.

N	*GR* *	***A*5(*q* = ∞, *α*[*a*, *b*])**	***A*4(*q* = ∞, *α*)**							
		***α* ∈ [−0.9, 1.8]**	***α* = −0.1**	***α* = 0.0**	***α* = 0.1**	***α* = 0.2**	***α* = 0.4**	***α* = 0.6**	***α* = 0.8**	***α* = 0.95**
10	14,019 (34,549)	11,329	28,788	28,782	28,783	28,763	28,781	28,769	22,130	11,040
100	14,785 (36,151)	11,293	29,777	29,776	29,769	29,775	29,766	29,759	22,891	11,554
1000	15,958 (39,107)	12,489	34,124	34,122	34,115	34,121	34,119	34,105	26,240	13,563
10,000	17,636 (43,300)	16,481	39,386	39,384	39,378	39,384	39,380	39,375	30,278	16,122

* The number of function and gradient calculations is given in parentheses.

Table A20. Number of iterations for $f_{Q^{\wedge}2}$ function minimization with gradient interference Δ from initial point x_0, $N = 1000$.

Δ	$f_{Q^{\wedge}2}(x, [a_{max} = 1000])$				$f_{Q^{\wedge}2}(x, [a_{max} = 10000])$			
	$A1(q)$ $\alpha = 0.0$ $q = 1.1$	$A2(q)$ $\alpha = 0.0$ $q = 3$	$A5(q, \alpha[a, b])$ $\alpha \in [-0.95, 1.8]$ $q = 3$	$A5(q, \alpha[a, b])$ $\alpha \in [-0.95, 1.8]$ $q = \infty$	$A1(q)$ $\alpha = 0.0$ $q = 1.1$	$A2(q)$ $\alpha = 0.0$ $q = 3$	$A5(q, \alpha[a, b])$ $\alpha \in [-0.95, 1.8]$ $q = 3$	$A5(q, \alpha[a, b])$ $\alpha \in [-0.95, 1.8]$ $q = \infty$
0.2	3452	3465	2651	2453	33,790	34,023	26,031	22,973
0.4	3452	3459	2875	2782	33,756	33,927	27,882	26,563
0.6	3451	3451	3078	3036	33,701	33,801	29,704	29,109
0.8	3468	3463	3294	3276	33,693	33,719	31,545	31,256
1	3580	3567	3720	3659	34,041	33,998	33,506	33,427
1.2	3826	3812	4371	4353	35,222	35,226	38,304	38,566
1.4	4211	4192	5342	5314	37,552	37,519	47,043	46,838
1.6	4724	4691	6698	6693	40,786	40,792	56,293	56,151
1.8	5329	5289	17,758	17,770	44,952	44,909	242,613	275,452
2	6057	5974			50,041	49,760		
2.2	6865	6742			55,762	55,277		
2.4	7770	7596			62,273	61,395		
2.6.	8740	8529			69,538	68,166		
2.8	9839	9553			77,268	75,707		
3	10,969	10,663			86,043	83,887		
3.2	12,180	11,837			95,119	92,698		
3.4	13,525	13,124			104,769	102,139		
3.6	14,912	14,504			115,146	112,302		
3.8	16,354	15,912			125,313	122,981		
4	17,914	17,370			136,547	134,183		
4.2	19,566	18,980			148,435	145,973		
4.4	21,214	20,566			161,241	158,268		
4.6	23,088	22,306			174,628	171,257		
4.8	25,059	24,091			188,113	184,685		
5	27,136	26,017			202,016	198,775		
5.2	29,175	27,933			217,963	213,433		
5.4	31,189	29,910			233,419	228,438		
5.6	33,379	32,004			249,645	244,233		
5.8	35,822	34,176			266,880	260,484		
6	38,209	36,466			284,429	277,197		
6.2	40,653	38,781				294,631		
6.4	43,332	41,217						
6.6	45,830	43,705						
6.8	48,444	46,221						
7	51,438	48,728						
7.2	53,950	51,308						
7.4	56,921	54,157						
7.6	60,000	56,912						
7.8	62,919	59,614						
8	66,154	62,535						

Table A21. Number of iterations for function $f_R(x, [a_{max} = 100])$ minimization without interference from initial point x_0.

N	GR *	A5(q = ∞, α[a, b]) α ∈ [−0.9, 1.8]	A4(q = ∞, α) α = −0.1	α = 0.0	α = 0.1	α = 0.2	α = 0.4	α = 0.6	α = 0.8	α = 0.95
10	475 (951)	272	459	478	476	464	441	221	253	843
100	488 (979)	305	480	484	482	472	477	303	271	874
1000	533 (1073)	321	534	535	529	524	514	418	248	898
10,000	579 (1172)	363	590	583	584	572	582	489	319	996

* The number of function and gradient calculations is given in parentheses.

Table A22. Number of iterations for function $f_R(x, [a_{max} = 1000])$ minimization without interference from initial point x_0.

N	GR *	A5(q = ∞, α[a, b]) α ∈ [−0.9, 1.8]	A4(q = ∞, α) α = −0.1	α = 0.0	α = 0.1	α = 0.2	α = 0.4	α = 0.6	α = 0.8	α = 0.95
10	4718 (11,463)	1790	4781	4772	4776	4767	4755	4353	3490	864
100	4796 (11,348)	2101	4852	4846	4840	4849	4843	4824	3550	1798
1000	5222 (12,372)	2328	5276	5274	5271	5263	5261	5246	3941	2298
10,000	5682 (13,901)	2570	5802	5801	5763	5798	5780	5750	4258	2562

* The number of function and gradient calculations is given in parentheses.

Table A23. Number of iterations for f_R function minimization with gradient interference Δ from initial point x_0, $N = 1000$.

Δ	$f_R(x, [a_{max} = 100])$ A1(q) α = 0.0 q = 1.1	A2(q) α = 0.0 q = 3	A5(q, α[a, b]) α ∈ [−0.95, 1.8] q = 3	A5(q, α[a, b]) α ∈ [−0.95, 1.8] q = ∞	$f_R(x, [a_{max} = 1000])$ A1(q) α = 0.0 q = 1.1	A2(q) α = 0.0 q = 3	A5(q, α[a, b]) α ∈ [−0.95, 1.8] q = 3	A5(q, α[a, b]) α ∈ [−0.95, 1.8] q = ∞
0.2	535	538	425	373	5228	5276	4012	3599
0.4	537	539	447	437	5223	5271	4340	4174
0.6	537	538	482	477	5213	5259	4588	4511
0.8	552	552	527	527	5225	5265	4873	4826
1	599	591	587	587	5387	5412	5239	5203
1.2	672	657	679	679	5747	5758	5712	5692
1.4	774	747	850	850	6315	6308	6383	6371
1.6	901	857	1541	1542	7069	7042	8801	8805
1.8	1047	992			7986	7934		
2	1214	1140			9064	8986		
2.2	1392	1300			10,306	10,175		
2.4	1598	1478			11,627	11,488		
2.6.	1827	1676			13,103	12,917		
2.8	2062	1892			14,756	14,507		
3	2350	2123			16,519	16,199		
3.2	2630	2381			18,372	17,988		
3.4	2948	2654			20,363	19,950		
3.6	3246	2942			22,582	22,027		
3.8	3557	3241			24,705	24,211		
4	3931	3566			27,056	26,490		
4.2	4330	3904			29,433	28,891		
4.4	4729	4262			32,155	31,422		
4.6	5105	4618			34,719	34,101		
4.8	5567	4995			37,581	36,837		
5	6053	5371			40,432	39,688		

5.2	6525	5812	43,607	42,683
5.4	7057	6268	46,818	45,809
5.6	7560	6729	50,183	49,014
5.8	8104	7190	53,747	52,398
6	8688	7608	57,493	55,898
6.2	9356	8090	61,392	59,515
6.4	9945	8612	65,251	63,122
6.6	10,547	9166	69,389	66,927
6.8	11,186	9769	73,205	70,865
7	11,874	10,325	77,342	74,619
7.2	12,552	10,877	81,762	78,900
7.4	13,176	11,458	86,101	82,977
7.6	13,946	12,045	90,935	87,288
7.8	14,670	12,640	95,621	91,805
8	15,483	13,282	100,846	96,240

Appendix B

Table A24. Frequently used designations.

Designation	Meaning
h_k	Step of minimization method
h^*	Optimal step
g_k, ∇f	Gradient of a function
f^*	Optimal function value
$(.\,,.)$	Scalar product
$\|\ \|$	Vector norm
s_k	New direction for minimization
z_k	Step change
α, q	Step adaptation parameters
Δ	Interference level
N	Dimension

References

1. Lyu, F.; Xu, X.; Zha, X. An adaptive gradient descent attitude estimation algorithm based on a fuzzy system for UUVs. *Ocean Eng.* **2022**, *266*, 113025.
2. Rivas, J.M.; Gutierrez, J.J.; Guasque, A.; Balbastre, P. Gradient descent algorithm for the optimization of fixed priorities in real-time systems. *J. Syst. Archit.* **2024**, *153*, 103198.
3. Silaa, M.; Bencherif, A.; Barambones, O. A novel robust adaptive sliding mode control using stochastic gradient descent for PEMFC power system. *Int. J. Hydrogen Energy* **2023**, *48*, 17277–17292.
4. Li, S.; Wang, J.; Zhang, H.; Feng, Y.; Lu, G.; Zhai, A. Incremental accelerated gradient descent and adaptive fine-tuning heuristic performance optimization for robotic motion planning. *Expert Syst. Appl.* **2024**, *243*, 122794. https://doi.org/10.1016/j.eswa.2023.122794.
5. Zhao, W.; Huang, H. Adaptive stepsize estimation based accelerated gradient descent algorithm for fully complex-valued neural networks. *Expert Syst. Appl.* **2024**, *236*, 121166. https://doi.org/10.1016/j.eswa.2023.121166.
6. Chen, H.; Natsuaki, R.; Hirose, A. Polarization-aware prediction of mobile radio wave propagation based on complex-valued and quaternion neural networks. *IEEE Access* **2022**, *10*, 66589–66600.
7. Zeng, Z.; Sun, J.; Han, Z.; Hong, W. SAR automatic target recognition method based on multi-stream complex-valued networks. *IEEE Trans. Geosci. Remote Sens.* **2022**, *60*, 5228618.
8. Ke, T.; Hwang, J.; Guo, Y.; Wang, X.; Yu, S. Unsupervised hierarchical semantic segmentation with multiview cosegmentation and clustering transformers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 19–24 June 2022; pp. 2571–2581.

9. Li, C.; Yao, K.; Wang, J.; Diao, B.; Xu, Y.; Zhang, Q. Interpretable generative adversarial networks. In Proceedings of the AAAI Conference on Artificial Intelligence, Washington, DC, USA, 7–14 February 2022.
10. Polyak, B.T. Gradient methods for the minimisation of functional. *USSR Comput. Math. Math. Phys.* **1963**, *3*, 864–878. https://doi.org/10.1016/0041-5553(63)90382-3.
11. Polyak, B.T. *Introduction to Optimization*; Optimization Software: New York, NY, USA, 1987.
12. Nesterov, Y. *Introductory Lectures on Convex Optimization*; Kluwer Academic Publisher: Boston, MA, USA, 2004.
13. Lojasiewicz, S. Une propriété topologique des sous-ensembles analytiques réels. *Les Équations Aux Dérivées Partielles* **1963**, *117*, 87–89.
14. Karimi, H.; Nutini, J.; Schmidt, M. Linear convergence of gradient and proximal-gradient methods under the Polyak- Lojasiewicz condition. In Proceedings of the Machine Learning and Knowledge Discovery in Databases Conference, Riva del Garda, Italy, 19–23 September 2016; p. 795811.
15. Yue, P.; Fang, C.; Lin, Z. On the lower bound of minimizing Polyak-Lojasiewicz functions. In Proceedings of the 36th Annual Conference on Learning Theory, Bangalore, India, 12–15 July 2023; pp. 2948–2968.
16. Stonyakin, F.; Kuruzov, I.; Polyak, B. Stopping rules for gradient methods for nonconvex problems with additive noise in gradient. *J. Optim. Theory Appl.* **2023**, *198*, 531–551. https://doi.org/10.1007/s10957-023-02245-w.
17. Devolder, O. Exactness, Inexactness and Stochasticity in First-Order Methods for Largescale Convex Optimization. Ph.D. Thesis, CORE UCLouvain, Louvain-la-Neuve, Belgium, 2013.
18. d'Aspremont, A. Smooth optimization with approximate gradient. *SIAM J. Optim*. **2008**, *19*, 11711183.
19. Vasin, A.; Gasnikov, A.; Spokoiny, V. *Stopping Rules for Accelerated Gradient Methods with Additive Noise in Gradient*; Preprint No. 2812; WIAS: Berlin, Germany, 2021. https://doi.org/10.20347/WIAS.PREPRINT.2812.
20. Nesterov, Y. Universal gradient methods for convex optimization problems. *Math. Program. Ser. A* **2015**, *152*, 381–404.
21. Puchinin, S.; Stonyakin, F. Gradient-Type Method for Optimization Problems with Polyak-Lojasiewicz Condition: Relative Inexactness in Gradient and Adaptive Parameters Setting. Available online: https://arxiv.org/abs/2307.14101 (accessed on 30 October 2024).
22. Wang, Q.; Su, F.; Dai, S.; Lu, X.; Liu, Y. AdaGC: A Novel Adaptive Optimization Algorithm with Gradient Bias Correction. *Expert Syst. Appl.* **2024**, *256*, 124956. https://doi.org/10.1016/j.eswa.2024.124956.
23. Zeiler, M.D. ADADELTA: An Adaptive Learning Rate Method. Available online: https://arxiv.org/abs/1212.5701 (accessed on 30 October 2024).
24. Ablaev, S.; Beznosikov, A.; Gasnikov, A.; Dvinskikh, D.; Lobanov, A.; Puchinin, S.; Stonyakin, F. About Some Works of Boris Polyak on Convergence of Gradient Methods and Their Development. Available online: https://arxiv.org/abs/2311.16743 (accessed on 30 October 2024).
25. Loizou, N.; Vaswani, S.; Laradji, I.; Lacoste-Julien, S. Stochastic Polyak step-size for SGD: An adaptive learning rate for fast convergence. In Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS-2021), Virtual, 13–15 April 2021; PMLR, Volume 130, pp. 1306–1314. https://doi.org/10.48550/arXiv.2002.10542.
26. Hazan, E.; Kakade, S. Revisiting the Polyak Step Size. Available online: https://arxiv.org/abs/1905.00313 (accessed on 30 October 2024).
27. Orvieto, A.; Lacoste-Julien, S.; Loizou, N. Dynamics of SGD with stochastic Polyak stepsizes: Truly adaptive variants and convergence to exact solution. In *Advances in Neural Information Processing Systems*; Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., Oh, A., eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2022; Volume 35, pp. 26943–26954.
28. Jiang, X.; Stich, S.U. Adaptive SGD with Polyak stepsize and Line-search: Robust Convergence and Variance Reduction. In Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS 2023), New Orleans, LA, USA, 10–16 December 2023; pp. 1–29.
29. Berrada, L.; Zisserman, A.; Kumar, M.P. Training neural networks for and by interpolation. In Proceedings of the 37th International Conference on Machine Learning (ICML'20), Virtual, 13–18 July 2020; PLMR, Volume 119.
30. Prazeres, M.; Oberman, A.M. Stochastic Gradient Descent with Polyak's Learning Rate. *J. Sci. Comput.* **2021**, *89*, 25. https://doi.org/10.1007/s10915-021-01628-3.
31. Rolinek, M.; Martius, G. L4: Practical loss-based stepsize adaptation for deep learning. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS 2018), Montreal, QC, Canada, 3–8 December 2018; Volume 31, Curran Associates, Inc.: Red Hook, NY, USA, 2018; pp. 6434–6444.
32. Gower, R.; Defazio, A.; Rabbat, M. Stochastic Polyak Stepsize with a Moving Target. Available online: https://arxiv.org/abs/2106.11851 (accessed on 30 October 2024).

33. Duchi, J.; Hazan, E.; Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res*. **2011**, *12*, 2121–2159.
34. Orabona, F.; Pál, D. Scale-Free Algorithms for Online Linear Optimization. In *Algorithmic Learning Theory*; Chaudhuri, K., Gentile, C., Zilles, S., Eds.; ALT 2015, Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2015. Volume 9355. https://doi.org/10.1007/978-3-319-24486-0_19.
35. Vaswani, S.; Laradji, I.; Kunstner, F.; Meng, S.Y.; Schmidt, M.; Lacoste-Julien, S. Adaptive Gradient Methods Converge Faster with Over-Parameterization (But You Should Do a Line-Search). Available online: https://arxiv.org/abs/2006.06835 (accessed on 30 October 2024).
36. Vaswani, S.; Mishkin, A.; Laradji, I.; Schmidt, M.; Gidel, G.; Lacoste-Julien, S. Painless Stochastic Gradient: Interpolation, Line-Search, and Convergence Rates. Available online: https://arxiv.org/abs/1905.09997 (accessed on 30 October 2024).
37. Reddi, S.J.; Kale, S.; Kumar, S. On the convergence of Adam and beyond. In Proceedings of the 6th International Conference on Learning Representations (ICLR), Vancouver, BC, Canada, 30 April–3 May 2018.
38. Tan, C.; Ma, S.; Dai, Y.-H.; Qian, Y. Barzilai-Borwein step size for stochastic gradient descent. In Proceedings of the Advances in Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; pp. 685–693.
39. Chen, J.; Zhou, D.; Tang, Y.; Yang, Z.; Cao, Y.; Gu, Q. Closing the Generalization Gap of Adaptive Gradient Methods in Training Deep Neural Networks. Available online: https://arxiv.org/abs/1806.06763 (accessed on 30 October 2024).
40. Ward, R.; Wu, X.; Bottou, L. AdaGrad stepsizes: Sharp convergence over nonconvex landscapes. In Proceedings of the 36th International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; PMLR; Volume 97, pp. 6677–6686.
41. Xie, Y.; Wu, X.; Ward, R. Linear convergence of adaptive stochastic gradient descent. In Proceedings of the 33 International Conference on Artificial Intelligence and Statistics, Online, 26–28 August 2020; PMLR; Volume 108, pp. 1475–1485.
42. Li, X.; Orabona, F. On the convergence of stochastic gradient descent with adaptive stepsizes. In Proceedings of the 22 International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Japan, 16–18 April 2019; Volume 89, pp. 983–992.
43. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. Available online: https://arxiv.org/abs/1412.6980 (accessed on 30 October 2024).
44. Polyak, B.T. Some methods of speeding up the convergence of iteration methods. *USSR Comp. Math. Mat. Phys*. **1964**, *4*, 1–17.
45. Tieleman, T.; Hinton, G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA Neural Netw. Mach. Learn.* **2012**, *4*, 26.
46. Dozat, T. Incorporating Nesterov Momentum into Adam. In Proceedings of the ICLR 2016 workshop, San Juan, Puerto Rico, 2–4 May 2016. Available online: https://openreview.net/forum?id=OM0jvwB8jIp57ZJjtNEZ (accessed on 16 December 2024).
47. Liu, L.; Jiang, H.; He, P.; Chen, W.; Liu, X.; Gao, J.; Han, J. On the Variance of the Adaptive Learning Rate and Beyond. Available online: https://arxiv.org/abs/1908.03265 (accessed on 30 October 2024).
48. Dauphin, Y.; Vries, H.D.; Bengio, Y. RMSProp and Equilibrated Adaptive Learning Rates for Non-Convex Optimization. Available online: https://arxiv.org/pdf/1502.04390v1 (accessed on 30 October 2024).
49. Yousefian, F.; Nedic, A.; Shanbhag, U. On stochastic gradient and subgradient methods with adaptive steplength sequences. *Automatica* **2012**, *48*, 56–67.
50. Wu, X.; Ward, R.; Bottou, L. WNGrad: Learn the Learning Rate in Gradient Descent. Available online: https://arxiv.org/abs/1803.02865 (accessed on 30 October 2024).
51. Levy, K.Y.; Yurtsever, A.; Cevher, V. Online Adaptive Methods, Universality and Acceleration. Available online: https://arxiv.org/abs/1809.02864 (accessed on 30 October 2024).
52. He, C.; Zhang, Y.; Zhu, D.; Cao, M.; Yang, Y. A mini-batch algorithm for large-scale learning problems with adaptive step size. *Digit. Signal Process*. **2023**, *143*, 104230. https://doi.org/10.1016/j.dsp.2023.104230.
53. Zhuang, J.; Tang2, T.; Ding, Y.; Tatikonda, S.; Dvornek, N.; Papademetris, X.; Duncan, J. AdaBelief Optimizer: Adapting Stepsizes by the Belief in Observed Gradients. In Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, BC, Canada, 6–12 December 2020; Volume 33, pp. 1–12.
54. Zou, W.; Xia, Y.; Cao, W. AdaDerivative optimizer: Adapting step-sizes by the derivative term in past gradient information. *Eng. Appl. Artif. Intell.* **2023**, *119*, 105755. https://doi.org/10.1016/j.engappai.2022.105755.
55. Gaudioso, M.; Giallombardo, G.; Miglionico, G.; Vocaturo, E. Classification in the multiple instance learning framework via spherical separation. *Soft Comput.* **2020**, *24*, 5071–5077. https://doi.org/10.1007/s00500-019-04255-1.
56. Astorino, A.; Fuduli, A.; Gaudioso, M. A Lagrangian Relaxation Approach for Binary Multiple Instance Classification. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 2662–2671. https://doi.org/10.1109/TNNLS.2018.2885852.

57. Fuduli, A.; Gaudioso, M.; Khalaf, W.; Vocaturo, E. A heuristic approach for multiple instance learning by linear separation. *Soft Comput*. **2022**, 26, 3361–3368. https://doi.org/10.1007/s00500-021-06713-1.
58. Kuruzov, I.A.; Stonyakin, F.S.; Alkousa, M.S. Gradient-type methods for optimization problems with Polyak–Lojasiewicz condition: Early stopping and adaptivity to inexactness parameter. In *Advances in Optimization and Applications*; Olenev, N., Evtushenko, Y., Jaćimović, M., Khachay, M., Malkova, V., Pospelov, I., Eds.; OPTIMA 2022, Communications in Computer and Information Science; Springer: Berlin/Heidelberg, Germany, 2022; Volume 1739, pp. 18–32.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.