ELSEVIER

Journal of Computational and Applied Mathematics 114 (2000) 367–386

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

www.elsevier.nl/locate/cam

A class of gradient unconstrained minimization algorithms with adaptive stepsize

M.N. Vrahatis[a,*], G.S. Androulakis[a], J.N. Lambrinos[a], G.D. Magoulas[b]

[a]*Department of Mathematics, University of Patras Artificial Intelligence Research Center (UPAIRC), University of Patras, GR-261.10 Patras, Greece*

[b]*Department of Informatics, University of Athens, GR-157.71 Athens, Greece*

Received 16 April 1997; received in revised form 22 January 1999

Abstract

In this paper the development, convergence theory and numerical testing of a class of gradient unconstrained minimization algorithms with adaptive stepsize are presented. The proposed class comprises four algorithms: the first two incorporate techniques for the adaptation of a common stepsize for all coordinate directions and the other two allow an individual adaptive stepsize along each coordinate direction. All the algorithms are computationally efficient and possess interesting convergence properties utilizing estimates of the Lipschitz constant that are obtained without additional function or gradient evaluations. The algorithms have been implemented and tested on some well-known test cases as well as on real-life artificial neural network applications and the results have been very satisfactory. © 2000 Elsevier Science B.V. All rights reserved.

MSC: 65K05; 65K10; 49D37; 65C20; 82C32; 68T05

Keywords: Unconstrained optimization; Steepest descent; Gradient method; Lipschitz constant; Line search strategies; Armijo's method; Globally convergent method; Artificial neural network; Training algorithm

1. Introduction

A well-known class of algorithms for unconstrained minimization of functions $f(x)$ in n real variables

$$f : \mathscr{D} \subset \mathbb{R}^n \to \mathbb{R}, \tag{1.1}$$

having Lipschitz continuous first partial derivatives whose gradient $\nabla f(x)$ is available, is the steepest descent methods [12,13,20,31,33,34,36] first proposed by Cauchy in 1847 [11]. The iterations are

* Corresponding author. http://www.math.upatras.gr/˜vrahatis; Tel.: +30-61-997374; fax: +30-61-992965.
E-mail address: vrahatis@math.upatras.gr (M.N. Vrahatis)

made according to the following equation:

$$x^{k+1} = x^k - \lambda^k \nabla f(x^k), \quad k = 0, 1, 2, \ldots, \tag{1.2}$$

where λ^k is the smallest nonnegative value of λ that locally minimizes f along the direction $-\nabla f(x^k)$ starting from x^k. Carry in 1944 [10] showed that any limit point x^* of the sequence $\{x^k\}_{k=0}^{\infty}$ is a stationary point; that is, $\nabla f(x^k) = 0$.

The iterative scheme (1.2) is not a computational method, because the stepsize rule at each step contains an exact one-dimensional minimization problem. This scheme can be implemented by doing an inexact minimization, utilizing a more efficient stepsize rule, first proposed by Armijo [3]. It can be shown that, under mild assumptions the iterative scheme (1.2) converges to a local minimizer x^* or saddle point of f, but its convergence is only linear and sometimes slowly linear. Specifically if it converges to a x^* where the Hessian matrix $H^* \equiv \nabla^2 f(x^*)$ of f at x^* is positive definite and $e_{\max}$ and $e_{\min}$ are the largest and smallest eigenvalues of H^*, then it can be shown that the sequence $\{x^k\}_{k=0}^{\infty}$ satisfies [13]

$$\lim_{k\to\infty} \sup \frac{\| x^{k+1} - x^* \|}{\| x^k - x^* \|} \leqslant c, \quad c = \frac{e_{\max} - e_{\min}}{e_{\max} + e_{\min}}. \tag{1.3}$$

The steepest descent method is particularly useful when the dimension of the problem is very large. On the other hand its main disadvantages are: (a) each iteration is calculated independently of the others; that is, no information is stored and used that might accelerate convergence, (b) it is not generally a finite procedure for minimizing a positive-definite quadratic form and (c) the rate of convergence depends strongly on the morphology of the objective function; if the ratio of the maximum to the minimum eigenvalue of the Hessian matrix H of f at any local minimizer x^* is large, the steepest descent generates short zigzagging moves in a neighborhood of x^* [20].

In the algorithmic framework of steepest descent methods, Goldstein proposed in 1962 [16] the iterative formula:

$$x^{k+1} = x^k - \lambda \varphi^k, \quad k = 0, 1, 2, \ldots, \tag{1.4}$$

where φ denotes a bounded map defined on the level set $S(x^0) = \{x\colon f(x) \leqslant f(x^0)\}$ satisfying the relation:

$$\langle \nabla f(x), \varphi(x) \rangle \geqslant 0, \tag{1.5}$$

so that for a given $\varepsilon > 0$, there exists $\delta > 0$ for which

$$\langle \nabla f(x), \varphi(x) \rangle < \delta \quad \text{implies} \quad \| \nabla f(x) \| < \varepsilon, \tag{1.6}$$

where $\langle \cdot, \cdot \rangle$ stands for the usual inner product in $\mathbb{R}^n$. He has also proved that the sequence $\{x^k\}_{k=0}^{\infty}$, generated by (1.4), converges to a local minimizer x^* of f. Following this approach Goldstein [17] has derived a formula to determine the stepsize λ at the kth iteration for the convergence of the sequence $\{x^k\}_{k=0}^{\infty}$ to a local minimizer x^* of f. Furthermore, by making the assumptions that $f \in C^2$ on a bounded $S(x^0)$, convergence has been established in case $\varphi(x) \equiv \nabla f(x)$. Finally, an estimate for the ultimate rate of convergence of the gradient vector has been given when a bound for the norm of the Hessian matrix H of f on $S(x^0)$ is known.

Using only values of the objective function f and its gradient, Goldstein and Price have extended previous results [18] by considering the case $\varphi(x) = Q_k^{-1} \nabla f(x)$ where Q_k is an approximation of the

Hessian matrix at x^k. Global convergence of their algorithm is provided at a rate which is eventually superlinear.

Alternatively, Armijo [3] has provided a modification of the steepest descent method which automatically adapts the stepsize and has proved convergence under less restrictive assumptions than those imposed by Goldstein [16]. For an efficient implementation of the Armijo's adaptation procedure as well as the corresponding convergence results we refer to [34]. Some important global convergence results for various methods using specific line search procedures have been given by [13,29,34,35,46,47,49].

The numerical algorithms proposed in this contribution exploit the local information regarding the local descent direction and the local Lipschitz constant estimates to provide better convergence. The paper is organized as follows. In Section 2 the modified steepest descent method due to Armijo and various line search strategies are briefly presented. In Section 3 our methods are derived and the corresponding algorithms presented. In Section 4 we present numerical results in order to compare and evaluate the performance of the new algorithms with several unconstrained optimization methods and finally end, in Section 5, with some concluding remarks.

2. The modified steepest descent and line search algorithms

Armijo provided in 1966 [3] a modification of the steepest descent method which automatically adapts the stepsize λ of the iterative scheme (1.2). His method and the corresponding convergence result are as follows:

Theorem 2.1 (Armijo [3]). *Suppose that the objective function $f:\mathbb{R}^n \to \mathbb{R}$ is continuous on $\mathbb{R}^n$ and bounded below on $\mathbb{R}^n$. Assume that for a given $x^0 \in \mathbb{R}^n$ the function f is continuously differentiable on the bounded level set $\mathscr{S}(x^0) = \{x\colon f(x) \leqslant f(x^0)\}$ and that there exists a unique point $x^* \in \mathbb{R}^n$ which minimizes f. Suppose further that the equation $\nabla f(x) = 0$ is satisfied for $x \in \mathscr{S}(x^0)$ if and only if $x = x^*$ and that ∇f is Lipschitz continuous on $\mathscr{S}(x^0)$, i.e., there exists a Lipschitz constant $K > 0$, such that*

$$\| \nabla f(x) - \nabla f(y) \| \leqslant K \| x - y \|, \tag{2.1}$$

for every pair $x, y \in S(x^0)$. Let λ^0 be an arbitrary assigned positive number, and consider the sequence $\lambda^m = \lambda^0/2^{m-1}$, $m = 1, 2, \ldots$. Then for the sequence of points $\{x^k\}_{k=0}^{\infty}$ defined by

$$x^{k+1} = x^k - \lambda^{m_k} \nabla f(x^k), \quad k = 0, 1, 2, \ldots, \tag{2.2}$$

where m_k is the smallest positive integer for which

$$f(x^k - \lambda^{m_k} \nabla f(x^k)) - f(x^k) \leqslant -\tfrac{1}{2} \lambda^{m_k} \| \nabla f(x^k) \|^2, \tag{2.3}$$

it holds that $\lim_{k\to\infty} x^k = x^*$.

The iterative scheme (2.2) and relation (2.3) constitute the *Armijo's method* or *the Modified Steepest Descent*. Next, we present a high-level description of the Armijo's algorithm, in which the corresponding parameters indicate: x^0 initial point, λ^0 an arbitrary large initial stepsize, MIT the maximum number of iterations required and ε the predetermined desired accuracy.

Algorithm 1. The modified steepest descent

```
1. Input $\{f; x^0; \lambda^0; \mathrm{MIT};\ \varepsilon\}$.
2. Set $k = -1$.
3. If $k < \mathrm{MIT}$, replace $k$ by $k+1$, set $\lambda = \lambda^0$, $m = 1$ and go to the next step; otherwise, go to Step 8.
4. If $f(x^k - \lambda \nabla f(x^k)) - f(x^k) \leqslant -\frac{1}{2}\lambda \| \nabla f(x^k) \|^2$, go to Step 6; otherwise, set $m = m + 1$ and go to the next step.
5. Set $\lambda = \lambda^0/2^{m-1}$ and return to Step 4.
6. Set $x^{k+1} = x^k - \lambda \nabla f(x^k)$.
7. If $\| \nabla f(x^k) \| \leqslant \varepsilon$, go to Step 8; otherwise go to Step 3.
8. Output $\{x^k; f(x^k); \nabla f(x^k)\}$.
```

Although Algorithm 1 provides an effective and very useful stepsize adaptation procedure for various applications [1,2,19,44], its exponential schedule is too fast for certain problems [26,27]. In [34, p. 30] an alternative Armijo's stepsize adaptation procedure is proposed applicable to any descent direction φ^k. This procedure uses two parameters $\alpha, \beta \in (0,1)$ and can be implemented in two versions depending on the input value of the parameter s.

Algorithm 2. Algorithm model with Armijo line search — ALS

```
1. Input $\{f; x^0; \alpha, \beta \in (0,1); s \in \{0,1\}; m^* \in \mathbb{Z}; \mathrm{MIT};\ \varepsilon\}$.
2. Set $k = 0$.
3. If $\| \nabla f(x^k) \| \leqslant \varepsilon$ go to Step 6. Else, compute a descent direction $\varphi^k$.
4. If $s = 0$, set $M^* = \{m \in \mathbb{Z} \mid m \geqslant m^*\}$, and compute the stepsize
   (a) $\lambda^k = \beta^{m_k} = \arg\max_{m \in M^*} \{\beta^m \mid f(x^k + \beta^m \varphi^k) - f(x^k) \leqslant \beta^m \alpha \langle \nabla f(x^k), \varphi^k \rangle\}$.
       Else ($s = 1$) compute the stepsize $\lambda^k = \beta^{m_k}$, where $m_k \in \mathbb{Z}$ is any integer such that
   (b) $f(x^k + \beta^{m_k} \varphi^k) - f(x^k) \leqslant \beta^{m_k} \alpha \langle \nabla f(x^k), \varphi^k \rangle$ and
   (c) $f(x^k + \beta^{m_k - 1} \varphi^k) - f(x^k) > \beta^{m_k - 1} \alpha \langle \nabla f(x^k), \varphi^k \rangle$.
5. Set $x^{k+1} = x^k + \lambda^k \varphi^k$. If $k < \mathrm{MIT}$, replace $k$ by $k+1$, and go to Step 3; otherwise go to Step 6.
6. Output $\{x^k; f(x^k); \nabla f(x^k)\}$.
```

The selection $s=0$ is normally used with Newton-like algorithms, with $m^*=0$ to ensure superlinear convergence. The selection $s = 0$ is not very good for first-order algorithms because, on average, it requires considerably more function evaluations than the selection $s = 1$. So, $s = 1$ is used in first-order algorithms.

If the objective function f is bounded from below the following subprocedure can be used to find an m_k satisfying relations (b) and (c) of Step 4 of Algorithm 2. This subprocedure uses the last used step length $\lambda^{k-1} = \beta^{m_{k-1}}$ as the starting point for the computation of the next step [34].

Stepsize subprocedure

```
1. If $k = 0$, set $m' = m^*$. Else set $m' = m_{k-1}$.
2. If $m_k = m'$ satisfies relations (b) and (c) of Step 4 of Algorithm 2, stop.
3. If $m_k = m'$ satisfies (b) but not (c), replace $m'$ by $m' - 1$, and go to Step 2.
   If $m_k = m'$ satisfies (c) but not (b), replace $m'$ by $m' + 1$, and go to Step 2.
```

In practice, only a very small number of iterations of the above subprocedure are required to compute the Armijo stepsize. When a very small stepsize occurs for several iterations, causing slow convergence, the user can revert to setting $s=0$ for one or two iterations.

The search strategy of Algorithm 2 allows us to establish the following useful convergence theorem due to Polak et al. [34, p. 33]. This theorem requires the search direction φ^k to be bounded from above, it imposes a restriction on the angle between $\nabla f(x^k)$ and φ^k (see relation (2.6) below) and states that Algorithm 2 is well defined in the sense that whenever $\nabla f(x^k) \neq 0$, the search for a stepsize λ^k is a finite process, whether $s=0$ or 1.

Theorem 2.2 (Polak–Sargent–Sebastian [34]). *Assume that* (i) *the objective function* $f : \mathbb{R}^n \to \mathbb{R}$ *is Lipschitz continuously differentiable on bounded sets*; (ii) *the sequences* $\{x^k\}_{k=0}^{\infty}$ *and* $\{\varphi^k\}_{k=0}^{\infty}$ *are constructed by Algorithm* 2; (iii) *there exist two continuous functions* $N_1 : \mathbb{R}^n \to \mathbb{R}$ *and* $N_2 : \mathbb{R}^n \to \mathbb{R}$ *such that*
(1) *for all* x *satisfying* $\nabla f(x) \neq 0$, $N_1(x) > 0$, $N_2(x) > 0$ *and* $N_1(x)=0$ *if and only if* $\nabla f(x)=0$ *and*
(2) *for all* $k \in \mathbb{N}$, *the* x^k *and* φ^k *satisfy the inequalities* $\langle \nabla f(x^k), \varphi(x^k)\rangle \leqslant -N_1(x^k)$, *and* $\| \varphi^k \| \leqslant N_2(x^k)$.
Under these assumptions, (a) *if* x^k *is such that* $\nabla f(x^k) \neq 0$, *then* λ^k *is computed by Algorithm* 2 *using a finite number of function evaluations and* (b) *any accumulation point* x^* *of the sequence* $\{x^k\}_{k=0}^{\infty}$ *satisfies* $\nabla f(x^*)=0$.

A relative strategy consists in accepting a positive stepsize λ^k if it satisfies the *Wolfe conditions*:

$$f(x^k+\lambda^k\varphi^k) - f(x^k) \leqslant \sigma_1 \lambda^k \langle \nabla f(x^k), \varphi^k\rangle, \tag{2.4}$$

$$\langle \nabla f(x^k+\lambda^k\varphi^k), \varphi^k\rangle \geqslant \sigma_2 \langle \nabla f(x^k), \varphi^k\rangle, \tag{2.5}$$

where $0<\sigma_1<\sigma_2<1$. Also, in this case the first inequality ensures that the function is reduced sufficiently, and the second prevents the steps from being too small. It can be shown that if φ^k is a descent direction, if f is continuously differentiable and if f is bounded below along the ray $\{x^k+\lambda\varphi^k \mid \lambda>0\}$, then there always exists a stepsize satisfying (2.4)–(2.5) [29,46,47].

The following theorem, due to Wolfe [13,29,46,47], states that if f is bounded below, then the sequence $\{x^k\}_{k=0}^{\infty}$ generated by any algorithm that follows a descent direction φ^k whose angle θ_k with $-\nabla f(x^k)$ is such that

$$\cos\theta_k = \frac{\langle -\nabla f(x^k), \varphi^k\rangle}{\| \nabla f(x^k) \| \| \varphi^k \|} \geqslant \delta > 0, \tag{2.6}$$

and satisfy the Wolfe's conditions, will obey $\lim_{k\to\infty} \nabla f(x^k)=0$.

Theorem 2.3 (Dennis and Schnabel [13], Nocedal [29], and Wolfe [46,47]). *Suppose that the objective function* $f : \mathbb{R}^n \to \mathbb{R}$ *is continuously differentiable on* $\mathbb{R}^n$ *and assume that* ∇f *is Lipschitz continuous on* $\mathbb{R}^n$. *Then, given any* $x^0 \in \mathbb{R}^n$, *either* f *is unbounded below, or there exists a sequence* $\{x^k\}_{k=0}^{\infty}$ *obeying the Wolfe's conditions* (2.4) *and* (2.5) *and either*
(i) $\langle \nabla f(x^k), (x^{k+1}-x^k)\rangle < 0$, *or*
(ii) $\nabla f(x^k)=0$, *and* $x^{k+1}-x^k=0$, *for each* $k>0$. *Furthermore, for any such sequence, either*

(a) $\nabla f(x) \neq 0$ *for some* $k \geqslant 0$, *or*
(b) $\lim_{k\to\infty} f(x^k) = -\infty$, *or*
(c) $\lim_{k\to\infty} \langle \nabla f(x^k), (x^{k+1} - x^k)\rangle / \parallel x^{k+1} - x^k \parallel = 0$.

For a relative convergence result where the sequence $\{x^k\}_{k=0}^{\infty}$ converges q-superlinearly to a minimizer x^* see [13, p. 123]

In practice, condition (2.5) generally is not needed because the use of a backtracking strategy avoids very small steps. Also, it can be proved (see [13]) that if (2.5) is replaced by

$$f(x^k + \lambda^k \varphi^k) - f(x^k) \geqslant \sigma_2 \lambda^k \langle \nabla f(x^k), \varphi^k \rangle, \quad \sigma_2 \in (\sigma_1, 1), \tag{2.7}$$

then Theorem 2.3 still holds (cf. relation (c) of Step 4 of Algorithm 2).

3. Derivation of the proposed minimization algorithms

The value of the stepsize λ in (1.2) has been related to the value of the Lipschitz constant K (cf. relation (2.1)), [3]. In this case, the well-known *Cauchy's method* or *the steepest descent algorithm* [11] states that the sequence $\{x^k\}_{k=0}^{\infty}$, defined by

$$x^{k+1} = x^k - \frac{1}{2K} \nabla f(x^k), \quad k = 0, 1, 2, \ldots, \tag{3.1}$$

converges to the point x^* which minimizes f (see [3] for a proof).

Note that when the objective function is "steep", i.e., K is large, then, according to iterative scheme (3.1), we have to take a small value for the stepsize in order to guarantee convergence. On the other hand, when f is "flat", i.e., K is small, we have to take a large stepsize to accelerate the convergence. However, in general optimization problems neither the morphology of the objective function surface nor the value of K are known a priori. Thus, in practice, a "small" stepsize has to be chosen in order to avoid oscillations and to guarantee the steepest descent convergence. On the other hand, choosing a "small" stepsize leads to slow convergence. Attempts to find a proper stepsize usually result in a trade-off between the convergence rate and the stability of the optimization algorithm.

In this section we propose unconstrained minimization algorithms with adaptive stepsize derived from the classical method of steepest descent.

3.1. Stepsize adaptation using local estimation of the Lipschitz constant

In this subsection two steepest descent algorithms with adaptive stepsize are described. These algorithms follow the steepest descent direction, $-\nabla f(x^k)$, and they use a local estimation of the Lipschitz constant Λ^k in order to estimate the stepsize $0.5/K$ at the kth iteration.

The local estimation of the Lipschitz constant Λ^k can be easily obtained without any additional function and gradient evaluations by relation (2.1) for a pair of consecutive updates x^k, x^{k-1}:

$$\Lambda^k = \frac{\parallel \nabla f(x^k) - \nabla f(x^{k-1}) \parallel}{\parallel x^k - x^{k-1} \parallel}. \tag{3.2}$$

In this way, the stepsize $0.5/\Lambda^k$ would be sensitive to the local shape of the objective function. In other words, when $\Lambda^k \neq 0$ relation (3.2) provides us with the following iterative scheme, named *steepest descent with adaptive stepsize* (SDAS):

$$x^{k+1} = x^k - \frac{1}{2\Lambda^k}\nabla f(x^k), \quad k = 0, 1, \ldots, \tag{3.3}$$

where Λ^k is given by relation (3.2). Clearly, the iterative scheme (3.3) is related to the iterative scheme (3.1) and it converges when $0 \leqslant 0.5/\Lambda^k \leqslant K^{-1}$. This can be easily justified following, for instance, the proof in [3].

In practice, the above-mentioned requirements are not easily checked. Thus, in order to ensure global convergence (i.e., convergence to a solution from any initial point, if a solution exists), it is necessary to have some criterion of acceptance of any approximation x^{k+1} of a minimizer x^*, generated by the iterative scheme (3.3). To this end, relations (2.4) and (2.5) or (2.4) and (2.7) as well as the corresponding relations (b) and (c) of Algorithm 2 can be used to obtain an appropriate stepsize. In these cases the value of $0.5/\Lambda^k$ can assist in the initialization of the input parameters of the corresponding algorithms. For instance, in Algorithm 2 by choosing constant values of α and β, say $0 < \alpha < \frac{1}{2}$ and $\frac{1}{2} < \beta < 1$, (see [13] for a discussion of the usefulness of these values) the parameter m^* can be initialized by setting $0.5/\Lambda^k = \alpha\beta^{m^*}$ and taken

$$m^* = -\left\lceil \frac{\log(2\Lambda^k\alpha)}{\log(\beta)} \right\rceil, \tag{3.4}$$

where $\lceil\cdot\rceil$ defines the ceiling of the real number quoted.

The following high-level algorithm, named SDAS-2, is the composition of the SDAS method with a *stepsize tuning subprocedure* consisting of any of the above pairs of relations. If Step 5 is ignored, SDAS-2 reduces to the SDAS algorithm.

Algorithm 3. The Steepest Descent with Adaptive Stepsize — SDAS-2

```
1. Input $\{f; x^0; \lambda^0; \mathrm{MIT};\ \varepsilon\}$.
2. Set $k = -1$.
3. If $k < \mathrm{MIT}$, replace $k$ by $k+1$ and go to the next step; otherwise, go to Step 8.
4. If $k \geqslant 1$ and $\Lambda^k = \| \nabla f(x^k) - \nabla f(x^{k-1}) \| / \| x^k - x^{k-1} \| \neq 0$, set $\lambda = 0.5/\Lambda^k$; otherwise set $\lambda = \lambda^0$.
5. Tune $\lambda$ by means of a stepsize tuning subprocedure.
6. Set $x^{k+1} = x^k - \lambda\nabla f(x^k)$.
7. If $\| \nabla f(x^k) \| \leqslant \varepsilon$ go to Step 8; otherwise go to Step 3.
8. Output $\{x^k; f(x^k); \nabla f(x^k)\}$.
```

Assume now that the stepsize tuning subprocedure of Step 5 of Algorithm 3 consists of the pair of relations (2.4) and (2.5) that define the Wolfe's conditions. The following theorem states that if f is bounded below, then the sequence $\{x^k\}_{k=0}^{\infty}$ generated by Algorithm 3 converges to a point x^* for which $\nabla f(x^*) = 0$.

Theorem 3.1. *Suppose that the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and bounded below on $\mathbb{R}^n$ and assume that ∇f is Lipschitz continuous on $\mathbb{R}^n$. Then, given any $x^0 \in \mathbb{R}^n$,* *for any sequence* $\{x^k\}_{k=0}^{\infty}$, *generated by Algorithm* 3, *satisfying the Wolfe's conditions* (2.4) *and* (2.5) *implies that* $\lim_{k\to\infty} \nabla f(x^k) = 0$.

Proof. Obviously, the sequence $\{x^k\}_{k=0}^{\infty}$ follows a descent direction and the restriction on the angle θ_k is fulfilled since $\cos\theta_k = 1 > 0$ (cf. relation (2.6)). Thus, by Theorem 2.3 it holds that $\lim_{k\to\infty} \nabla f(x^k) = 0$. Thus the theorem is proved. □

A relative convergence result can be proved, using Theorem 2.2, for any sequence $\{x^k\}_{k=0}^{\infty}$ satisfying relations (b) and (c) of Step 4 of Algorithm 2. A detailed justification is provided in [34].

3.2. Stepsize adaptation along each coordinate direction using estimates of the Lipschitz constant

In this subsection, unconstrained minimization algorithms with an adaptive stepsize for each coordinate direction are analyzed as composite *nonlinear Jacobi* methods applied to the gradient of the objective function f. The class of nonlinear Jacobi methods is widely used for the numerical solution of a system of nonlinear equations:

$$F(x) = \Theta^n = (0, 0, \ldots, 0), \tag{3.5}$$

where $F = (f_1, f_2, \ldots, f_n) : \mathscr{D} \subset \mathbb{R}^n \to \mathbb{R}^n$ is a continuously differentiable mapping on an open neighborhood $\mathscr{D}^* \subset \mathscr{D}$ of a solution $x^* \in \mathscr{D}$ of (3.5). The main feature of the nonlinear Jacobi process is that it is a parallel algorithm [23,31,32], i.e., it applies a parallel update of the variables.

In function minimization problems, it is well known that all the local minima x^* of a continuously differentiable function f satisfy the necessary conditions

$$\nabla f(x^*) = \Theta^n. \tag{3.6}$$

Eq. (3.6) represents a set of n nonlinear equations which must be solved to obtain x^*. Therefore, one approach to the minimization of a function f is to seek the solutions of the set of Eq. (3.6) by including a provision to ensure that the solution found does indeed correspond to a local minimizer. This is equivalent to solving the following system of equations:

$$\begin{aligned} \partial_1 f(x_1, x_2, \ldots, x_n) &= 0, \\ \partial_2 f(x_1, x_2, \ldots, x_n) &= 0, \\ &\vdots \\ \partial_n f(x_1, x_2, \ldots, x_n) &= 0, \end{aligned} \tag{3.7}$$

by applying the class of nonlinear Jacobi methods, where $\partial_i f$ denotes the ith coordinate of ∇f.

So, starting from an arbitrary initial point $x^0 \in \mathscr{D}$, one can subminimize, at the kth iteration, the one-dimensional function:

$$f(x_1^k, \ldots, x_{i-1}^k, x_i, x_{i+1}^k, \ldots, x_n^k), \tag{3.8}$$

along the ith direction and a corresponding subminimizer $\hat{x}_i$ is obtained. This is done in parallel for all $i = 1, \ldots, n$. Obviously for this $\hat{x}_i$

$$\partial_i f(x_1^k, \ldots, x_{i-1}^k, \hat{x}_i, x_{i+1}^k, \ldots, x_n^k) = 0. \tag{3.9}$$

This is a one-dimensional subminimization because all other components of the vector x, except the ith, are kept constant.

Then each variable is updated according to the following equation:

$$x_i^{k+1} = x_i^k + \omega^k(\hat{x}_i - x_i^k), \tag{3.10}$$

for some relaxation coefficient $\omega^k \in (0,1]$.

When exact one-dimensional subminimization is applied and $\omega^k = 1$ for all k the following convergence result is provided.

Theorem 3.2 (Brewster and Kannan [7]). *Suppose that the objective function $f : \mathscr{D} \subset \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable on a convex domain $\mathscr{D}$ and that f is a strictly convex function. Assume that there exists $\gamma \in \mathbb{R}$ such that $\mathscr{S}_\gamma = \{x \in \mathscr{D}\colon f(x) \leqslant \gamma\}$ is nonempty and compact and that $\partial_{ii}^2 f(y) \neq 0$ for $i = 1,2,\ldots,n$ and $y \in \mathscr{S}_\gamma$, unless y is the point at which f attains its minimum, where $\partial_{ij}^2 f(y)$ denotes the h_{ij} element of the Hessian matrix of f at y, $H = [h_{ij}]$. Suppose further, that from any point $x^0 = (x_1^0, x_2^0, \ldots, x_n^0) \in \mathscr{S}_\gamma$ a sequence $\{x^k\}_{k=0}^\infty$ is generated as follows:*

$$x_j^{k+1} = x_j^k,\ j \neq i_k,$$

and

$$x_{i_k}^{k+1} \text{ is the solution of } \partial_{i_k} f(x_1^k, x_2^k, \ldots, x_{i_k-1}^k, x_{i_k}, x_{i_k+1}^k, \ldots, x_n^k) = 0,$$

where i_k is any one of the integers $1,2,\ldots,n$. Such a sequence $\{x^k\}_{k=0}^\infty$ is uniquely defined and converges to x^*, the unique global minimizer of f, provided that in the above iterative process every coordinate direction i is chosen an infinite number of times.

Depending on the one-dimensional subminimization method [6,13,31,36], used for the subminimization of (3.8), we can obtain various composite nonlinear Jacobi algorithms. When inexact one-dimensional subminimization is applied, the number of iterations of the subminimization method is related to the requested accuracy in obtaining the subminimizer approximations. Thus, significant computational effort is needed in order to find very accurate approximations of the subminimizer in each coordinate direction at each iteration. Furthermore, the total computational effort for the subminimization method is increased when the dimension of the problem is very high. In addition, it is not certain that the large computational effort speeds up the minimization process for nonconvex functions when far from a minimizer x^*. Thus, in practice, only a single iteration of the one-dimensional method in each variable direction is usually suggested for the iterative solution of nonlinear equations [31,43].

By applying the ideas of the previous subsection in each direction we come up with two algorithms for unconstrained minimization. In this case, the updates are based on the components $\partial_i f(x^k)$ of $\nabla f(x^k)$ and on the following estimates for the Lipschitz constant along the ith direction, $i = 1,\ldots,n$, at the kth iteration:

$$\Lambda_i^k = \frac{|\partial_i f(x^k) - \partial_i f(x^{k-1})|}{|x_i^k - x_i^{k-1}|}. \tag{3.11}$$

Relation (3.11) can be considered as a local estimation of the Lipschitz constant K and its inverse can be used in order to estimate the stepsize along the ith direction. This means that for a large value of K a small stepsize is used and vice versa.

As a consequence, we propose the following iterative scheme

$$x^{k+1} = x^k - \operatorname{diag}\{\omega_1/\Lambda_1^k, \omega_2/\Lambda_2^k, \ldots, \omega_n/\Lambda_n^k\}\, \nabla f(x^k), \quad k = 0, 1, \ldots \tag{3.12}$$

for some relaxation coefficients $\omega_i^k \in (0, 1]$.

The iterative scheme (3.12) formulates a new method for unconstrained optimization named *gradient descent with adaptive multi-stepsize* (GDAM).

In order to give a convergence result for the new method, the following concepts and theorems are needed [4,31,43].

First, we present the notion of the A^π property: Young discovered in 1971 [48] a class of matrices, described as having the *property A*, that can be partitioned into block-tridiagonal form possibly after a suitable permutation. In Young's original presentation, the elements of a matrix $A = [a_{ij}]$ are partitioned into two groups. In general, any partitioning of an n-dimensional vector $x = (x^{(1)}, \ldots, x^{(m)})$ into block components $x^{(p)}$ of dimensions n_p, $p = 1, \ldots, m$ (with $\sum_{p=1}^m n_p = n$) is uniquely determined by a partitioning $\pi = \{\pi_p\}_{p=1}^m$ of the set of the first n integers, where π_p contains the integers $s_p + 1, \ldots, s_p + n_p$, $s_p = \sum_{j=1}^{k-1} n_j$. The same partitioning π also induces a partitioning of any $n \times n$ matrix A into block matrix components A_{ij} of dimensions $n_i \times n_j$. Note that the matrices A_{ii} are square.

Definition 3.3 (*Axelsson* [4]). The matrix A has the property A^π if A can be permuted by $\mathrm{PAP}^\top$ into a form that can be partitioned into a block-tridiagonal form, that is,

$$\mathrm{PAP}^\top = \begin{bmatrix} D_1 & L_1^\top & & & \mathcal{O} \\ L_1 & D_2 & L_2^\top & & \\ & \ddots & \ddots & \ddots & \\ & & L_{r-2} & D_{r-1} & L_{r-1}^\top \\ \mathcal{O} & & & L_{r-1} & D_r \end{bmatrix},$$

where the matrices D_i, $i = 1, \ldots, r$ are nonsingular.

For an algorithm which transforms a symmetric matrix to tridiagonal form see [39, p. 335].

Theorem 3.4 (Ortega and Rheinboldt [31, p. 333]). *Suppose that* $F = (f_1, \ldots, f_n) : \mathcal{D} \subset \mathbb{R}^n \to \mathbb{R}^n$ *is continuously differentiable on an open neighborhood* $\mathcal{S}_0 \subset \mathcal{D}$ of a point $x^* \in \mathcal{D}$ *for which* $F(x^*) = \Theta^n$. *Consider the decomposition of the Jacobian matrix* $F'(x)$ *into its diagonal, strictly lower-triangular and strictly upper-triangular parts*

$$F'(x) = D(x) - L(x) - U(x). \tag{3.13}$$

Suppose further that $D(x^*)$ *is nonsingular and* $\rho(\Phi(x^*)) < 1$, *where* $\rho(A)$ *denotes the spectral radius of matrix* A, *and* $\Phi(x)$ *is defined by*

$$\Phi(x) = D(x)^{-1}\,[L(x) + U(x)]. \tag{3.14}$$

Then there exists an open ball $\mathcal{S} = \mathcal{S}(x^*, r)$ *in* $\mathcal{S}_0$, (*where* $\mathcal{S}(x^*, r)$ *denotes the open ball centered at* x^* *with radius* r), *such that, for any* $x^0 \in \mathcal{S}$, *there is a unique sequence,* $\{x^k\}_{k=0}^\infty \subset \mathcal{S}$ *which satisfies the nonlinear Jacobi prescription, such that* $\lim_{k\to\infty} x^k = x^*$.

Theorem 3.5 (Axelsson [4, p. 238]). *If A and D are symmetric and positive definite and A has the property A^{π}, then the eigenvalues of $B = D^{-1}[L + L^{\top}]$, where $A = D - L - L^{\top}$, are real and* $\rho(B) < 1$.

In the following, we present a general convergence result which is applicable to any iterative scheme (3.10) utilizing inexact one-dimensional subminimization.

Theorem 3.6. *Let $f : \mathscr{D} \subset \mathbb{R}^n \to \mathbb{R}$ be twice continuously differentiable on an open neighborhood $\mathscr{S}_0 \subset \mathscr{D}$ of a point $x^* \in \mathscr{D}$ for which $\nabla f(x^*) = \Theta^n$ and the Hessian, $H(x^*)$ is positive definite with the property A^{π}. Then there exists an open ball $\mathscr{S} = \mathscr{S}(x^*, r)$ in $\mathscr{S}_0$ such that any sequence $\{x^k\}_{k=0}^{\infty}$ generated by the iterative scheme* (3.10) *converges to the point x^* which minimizes f.*

Proof. Clearly, the necessary and sufficient conditions for the point x^* to be a local minimizer of the function f are satisfied by the hypothesis $\nabla f(x^*) = \Theta^n$ and the assumption of positive definitiveness of the Hessian at x^*, which means that f curves up from x^* in all directions (see for example [31]). Finding such a point is equivalent to obtaining the corresponding solution $x^* \in \mathscr{D}$ of system (3.7) by applying a nonlinear Jacobi process using any one rootfinding iterative method.

Now, consider the decomposition (3.13) of H. Since, $H(x^*)$ is symmetric and positive definite, $D(x^*)$ is nonsingular [41, p. 80]. By virtue of Theorem 3.5 we obtain $\rho(\Phi(x^*)) < 1$. Thus, by Theorem 3.4, there exists an open ball $\mathscr{S} = \mathscr{S}(x^*, r)$ in $\mathscr{S}_0$, such that x^* is a point of attraction of the iterative scheme (3.10) for any $x^0 \in \mathscr{S}$. Thus the theorem is proved. □

By means of the above theorem the convergence proof of the proposed iterative scheme (3.12) can be easily obtained.

Iteration (3.12) can be reformulated as follows:

$$x^{k+1} = x^k - \omega^k \operatorname{diag}\{1/\Lambda_1^k, \ldots, 1/\Lambda_n^k\} \nabla f(x^k), \quad k = 0, 1, \ldots, \tag{3.15}$$

where ω^k is a relaxation coefficient.

To ensure global convergence of the iterative scheme (3.15) we can either apply relations (2.4) and (2.5), or (2.4) and (2.7), or the corresponding relations (b) and (c) of Algorithm 2 to properly tune ω^k. In this way, we can avoid oscillations of the updates around the minimizer and/or speed up the minimization process when we are far from the minimum.

Thus, by composing the GDAM algorithm with a tuning subprocedure for ω^k, consisting of any of the above pair of relations, we obtain a new algorithm named GDAM-2.

A high-level description of GDAM-2 algorithm is given below. The previous algorithm, GDAM, can be obtained by ignoring Step 5.

Algorithm 4. Gradient descent with adaptive multi-stepsize — GDAM-2

1. Input $\{f; x^0; \omega^0; (\lambda_1^0, \lambda_2^0, \ldots, \lambda_n^0); \mathrm{MIT}; \varepsilon\}$.
2. Set $k = -1$.
3. If $k < \mathrm{MIT}$, replace k by $k + 1$, set $\omega = \omega^0$, and go to the next step; otherwise, go to Step 8.
4. If $k \geqslant 1$ and $\Lambda_i^k = |\partial_i f(x^k) - \partial_i f(x^{k-1})| / |x_i^k - x_i^{k-1}| \neq 0$, for all $i = 1, \ldots, n$, set $\lambda_i^k = 1/\Lambda_i^k$; otherwise set $\lambda_i^k = \lambda_i^0$.

5. Tune ω by means of a tuning subprocedure.
6. Set $x^{k+1} = x^k - \omega \operatorname{diag}\{\lambda_1^k, \lambda_2^k, \dots, \lambda_n^k\} \nabla f(x^k)$.
7. If $\| \nabla f(x^k) \| \leqslant \varepsilon$ go to Step 8; otherwise go to Step 3.
8. Output $\{x^k; f(x^k); \nabla f(x^k)\}$.

Assume now that the tuning subprocedure of Step 5 of Algorithm 4 consists of the pair of relations (2.4) and (2.5). The following theorem states that if f is bounded below, then the sequence $\{x^k\}_{k=0}^{\infty}$ generated by Algorithm 4 converges to a point x^* for which $\nabla f(x^*) = 0$.

Theorem 3.7. *Suppose that the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and bounded below on $\mathbb{R}^n$ and assume that ∇f is Lipschitz continuous on $\mathbb{R}^n$. Then, given any point $x^0 \in \mathbb{R}^n$, for any sequence $\{x^k\}_{k=0}^{\infty}$, generated by Algorithm* 4, *satisfying the Wolfe's conditions* (2.4) *and* (2.5) *implies that* $\lim_{k\to\infty} \nabla f(x^k) = 0$.

Proof. The sequence $\{x^k\}_{k=0}^{\infty}$ follows the direction

$$\varphi^k(x^k) = -\operatorname{diag}\{1/\Lambda_1^k, \dots, 1/\Lambda_n^k\} \nabla f(x^k),$$

which is a descent direction since

$$\langle \nabla f(x^k), \varphi^k(x^k) \rangle < 0.$$

Moreover, the restriction on the angle θ_k is fulfilled since, as it can be easily justified utilizing relation (2.6), $\cos\theta_k > 0$. Thus, by Theorem 2.3 $\lim_{k\to\infty} \nabla f(x^k) = 0$ holds. Thus the theorem is proved. $\square$

A relative convergence result can be proved, using Theorem 2.2, for any sequence $\{x^k\}_{k=0}^{\infty}$ satisfying relations (b) and (c) of Step 4 of Algorithm 2.

4. Numerical applications

The proposed minimization algorithms have been tested on various problems of different dimensions and their performance has been compared with several well known and widely used unconstrained minimization methods. The numerical applications studied here include classical test cases as well as real-life applications such as artificial neural network training.

4.1. Classical test cases

In this subsection we compare the proposed methods with several widely used conjugate gradient and variable metric methods. In Tables 1–3, we present comparative numerical results for the methods: (i) Algorithm 2 (ALS) using $\varphi = -\nabla f$; (ii) the Fletcher–Reeves (FR) method [14,15]; (iii) the Polak–Ribiere (PR) method [15,33,34]; (iv) the Davidon–Fletcher–Powell (DFP) method [34]; (v) the Broyden–Fletcher–Goldfarb–Shanno (BFGS) method [9,34]; (vi) the Steepest Descent with Adaptive Stepsize (SDAS and SDAS-2) and (vii) the gradient descent with adaptive multi-stepsize (GDAM and GDAM-2). In all cases, the line search subprocedure of Algorithm 2 has been used. The initial relaxation coefficient of Algorithm 4 has been taken $\omega^0 = 1$. In Tables 1–3, the reported

Table 1
Results for the variably dimensioned function

Dimension	$n = 4$		$n = 8$		$n = 12$	
Method	IT	FE	IT	FE	IT	FE
ALS	20	170	14	199	23	435
FR	11	98	9	134	15	291
PR	16	130	20	277	21	382
DFP	9	96	9	149	13	330
BFGS	9	96	9	149	13	330
SDAS	34	175	44	405	44	546
SDAS-2	28	148	39	365	41	552
GDAM	17	90	21	198	24	325
GDAM-2	12	77	7	91	18	269

Table 2
Results for the trignometric function

Dimension	$n = 25$		$n = 50$		$n = 100$	
Method	IT	FE	IT	FE	IT	FE
ALS	49	1338	46	2437	20	2219
FR	33	923	27	1484	33	3541
PR	17	493	25	1415	24	2623
DFP	30	828	21	1200	45	4713
BFGS	35	964	34	1829	43	4500
SDAS	33	884	52	2703	63	6464
SDAS-2	33	887	36	1891	53	5471
GDAM	20	546	48	2499	27	1972
GDAM-2	10	290	18	974	18	2007

Table 3
Results for the penalty I function

Dimension	$n = 4$		$n = 8$		$n = 30$	
Method	IT	FE	IT	FE	IT	FE
ALS	D		D		D	
FR	13	81	12	133	12	439
PR	12	69	14	146	18	624
DFP	6	50	9	100	14	1875
BFGS	6	50	8	96	14	1882
SDAS	20	105	26	243	51	1612
SDAS-2	24	137	29	273	38	1223
GDAM	15	80	18	171	27	868
GDAM-2	6	40	9	97	19	635

parameters indicate: n the dimension of the problem, D divergence from the minimizer, IT the total number of iterations required to obtain the local minimizer $x^* = (x_1^*, x_2^*, \dots, x_n^*)$ and FE the total number of function and gradient evaluations. The termination criteria have been

$$|f(x)^{k+1} - f(x)^k| \leqslant 10^{-8} \text{ and } \| \nabla f(x^k) \| \leqslant 10^{-4}.$$

We have tested the above methods for various problems. The results exhibited here are for the following examples:

Example 4.1 (*Variably dimensioned function*, *More et al.* [28]). In this case f is given by

$$f(x) = \sum_{i=1}^{m} f_i^2(x), \tag{4.1}$$

where $m = n + 2$ and n is the number of variables. Also,

$$f_i(x) = x_i - 1, \tag{4.2}$$

$$f_{n+1}(x) = \sum_{j=1}^{n} j(x_j - 1), \tag{4.3}$$

$$f_{n+2}(x) = \left(\sum_{j=1}^{n} j(x_j - 1) \right)^2 . \tag{4.4}$$

This function has a minimizer $x^* = (1, 1, \dots, 1)$ with $f(x^*) = 0$. The initial point has been taken $x^0 = (\xi_j)$, where $\xi_j = 1 - (j/n)$. The results for this example are summarized in Table 1.

Example 4.2 (*Trigonometric function*, *More et al.* [28]). In this case f is given by

$$f(x) = \sum_{i=1}^{n} f_i^2(x), \tag{4.5}$$

where n is the number of variables and

$$f_i(x) = n - \sum_{j=1}^{n} \cos x_j + i(1 - \cos x_i) - \sin x_i. \tag{4.6}$$

Here we have $f(x^*) = 0$ and $x^0 = (1/n, \dots, 1/n)$. The results for this example are exhibited in Table 2.

Example 4.3 (*Penalty function I*, *More et al.* [28]). In this case f is given by

$$f(x) = \sum_{i=1}^{m} f_i^2(x), \tag{4.7}$$

where $m = n + 1$, n is the number of variables and

$$f_i(x) = a^{1/2}(x_i - 1), \quad 1 \leqslant i \leqslant n, \; a = 10^{-5}, \tag{4.8}$$

$$f_{n+1}(x) = \left(\sum_{j=1}^{n} x_j^2 \right) - \frac{1}{4}. \tag{4.9}$$

The initial point has been taken $x^0 = (1, 2, \dots, n)$. The results for this example are summarized in Table 3.

4.2. Artificial neural network training

Artificial neural networks, such as the multilayer feedforward neural network (FNN), are massively parallel, nonlinear systems modeled on the general features of biological networks [37]: collectively, neurons with simple properties, interacting according to relatively simple rules and organized into layers, provide an attractive, alternate algorithmic basis for problem solving in pattern recognition, pattern classification, learning and control.

The goal of neural network training is to iteratively update the network variables in order to globally minimize a measure of the difference between the actual output vector of the network and the desired output vector [22]. It is well known in the neural network field [22,37] that the rapid computation of such a global minimum is rather a difficult task since, in general, the number of network variables is large and the corresponding nonconvex multimodal objective function possesses multitudes of local minima and has broad flat regions adjoined with narrow steep ones.

The selection of initial points is very important in FNN training [45]. Very small initial values lead to very small corrections of the variables so that practically no change takes place for some variables and more iterations are necessary to train the network [37]. In the worst case the learning stops in an undesired local minimum. On the other hand, very large initial values speed up the learning process but in many cases they can lead neurons to saturation and generate very small gradient values. In such cases, learning is considerably slow [25]. A well-known initialization heuristic for FNNs is to select the points with uniform probability from an interval $(x_{\min}, x_{\max})$, where usually $x_{\min} = -x_{\max}$. A common choice is the interval $(-1, +1)$. Thus, 1000 initial starting points have been randomly selected from this interval to test the different methods.

Due to the special characteristics of the training problem, globally convergent methods are required. According to our experience the above-mentioned characteristics of the training problem can be better handled by the SDAS-2 and GDAM-2 versions of the proposed algorithms.

Next, we give quantitative results on three neural network applications applying the following methods: (i) the steepest descent with constant stepsize (SD) [37]; (ii) Algorithm 2 (ALS) using $\varphi = -\nabla f$; (iii) the steepest descent with constant stepsize and momentum (SDM) [37]; (iv) the adaptive steepest descent with heuristics (ASD) [42]; (v) the Fletcher–Reeves (FR) method [15]; (vi) the Polak–Ribiere (PR) method [15]; (vii) the Polak–Ribiere (PR) method constrained by the FR method (PR–FR) [15]. In the implementation of FR, PR, PR–FR, SDAS-2 and GDAM-2 methods, the Armijo line search of Algorithm 2 has been used. The results are exhibited in terms of the average number of iterations (μ_{IT}) required to obtain a local minimum, the average number of gradient and function evaluations (μ_{FE}) and the number of successful runs out of 1000 (Success).

Example 4.4 (*The XOR problem*, *Jacobs* [24], *Magoulas et al.* [26] *and Rumelhart* [37]). The classification of the four XOR patterns into two classes is an interesting problem because it is sensitive to initial points as well as to stepsize variations, and presents a multitude of local minima [5]. The patterns are classified using an FNN with nine variables. This classification problem corresponds

Table 4
Results for the XOR problem ($n = 9$)

Algorithm	μ_{IT}	μ_{FE}	Success
SD	549	1098	810/1000
ALS	64	435	810/1000
SDM	803	1606	810/1000
ASD	157	314	810/1000
FR	84	282	130/1000
PR	21	169	380/1000
PR–FR	22	171	410/1000
SDAS-2	40	162	810/1000
GDAM-2	52	234	810/1000

to the minimization of the following objective function:

$$\begin{aligned} f(x) = {} & \left[1+\exp\left(-\frac{x_7}{1+\exp(-x_1-x_2-x_5)}-\frac{x_8}{1+\exp(-x_3-x_4-x_6)}-x_9\right)\right]^{-2} \\ & + \left[1+\exp\left(-\frac{x_7}{1+\exp(-x_5)}-\frac{x_8}{1+\exp(-x_6)}-x_9\right)\right]^{-2} \\ & + \left[1-\left\{1+\exp\left(-\frac{x_7}{1+\exp(-x_1-x_5)}-\frac{x_8}{1+\exp(-x_3-x_6)}-x_9\right)\right\}^{-1}\right]^2 \\ & + \left[1-\left\{1+\exp\left(-\frac{x_7}{1+\exp(-x_2-x_5)}-\frac{x_8}{1+\exp(-x_4-x_6)}-x_9\right)\right\}^{-1}\right]^2. \qquad (4.10) \end{aligned}$$

The termination condition for all algorithms tested is to find a local minimizer with function value $f \leqslant 0.04$. The results are summarized in Table 4.

In this case the number of successful runs is related to the local minima problem. Thus FR, PR and PR–FR usually converge to an undesired local minimum, i.e. a minimizer with function value $f > 0.04$ which means that some of the patterns are not correctly classified. SDAS-2 and GDAM-2 exhibit better performance than FR, PR and PR–FR as regards the number of successful runs. Concerning training speed, measured by the mean number of function and gradient evaluations needed to successfully classify the patterns, SDAS-2 clearly outperforms all algorithms tested. GDAM-2 outperforms SD, ALS, SDM and FR. Note that PR and PR–FR require less function evaluations than GDAM-2 but they reveal a smaller number of successful runs.

Example 4.5 (*Texture classification problem*, *Magoulas et al.* [26]). A total of 12 Brodatz texture images [8]: 3, 5, 9, 12, 15, 20, 51, 68, 77, 78, 79, 93 (see Fig. 1 in [26]) of size 512×512 is acquired by a scanner at 150 dpi. From each texture image 10 subimages of size 128×128 are randomly selected, and the co-occurrence method, introduced by Haralick [21] is applied. In the co-occurrence method, the relative frequencies of gray-level pairs of pixels at certain relative displacements are computed and stored in a matrix. As suggested by other researchers [30,40], the combination of the nearest-neighbor pairs at orientations 0°, 45°, 90° and 135° are used in the experiment. A set of 10 sixteenth-dimensional training patterns are created from each image. The patterns are

Table 5
Results for the texture classification problem ($n = 244$)

Algorithm	μ_{IT}	μ_{FE}	Success
SD	15 839	31 678	960/1000
ALS	13 256	26 517	965/1000
SDM	12 422	24 844	940/1000
ASD	560	1120	1000/1000
FR	1624	12 674	250/1000
PR	140	810	990/1000
PR–FR	145	1005	996/1000
SDAS-2	383	973	1000/1000
GDAM-2	406	1228	1000/1000

presented in a finite sequence $C = (c_1, c_2, \dots, c_p)$ of input–output pairs $c_p = (u_p, t_p)$ where u_p are the real-valued input vectors in $\mathbb{R}^{16}$ and t_p are binary output vectors in $\mathbb{R}^{12}$, for $p = 1, \dots, 120$, determining the corresponding training pattern. An FNN with 244 variables is trained to classify the patterns to the 12 texture types.

In order to train the network we have to find variable values that minimize the following objective function:

$$f(x) = \sum_{p=1}^{120} \sum_{j=1}^{12} \left[\left\{ 1 + \exp\left(\sum_{i=1}^{8} w_{ij} y_{i,p} + \tau_j \right) \right\}^{-1} - t_{j,p} \right]^2, \tag{4.11}$$

where

$$y_{i,p} = \left\{ 1 + \exp\left(\sum_{k=1}^{16} v_{ki} u_{k,p} + b_i \right) \right\}^{-1}, \tag{4.12}$$

$$x = (w_{11}, \dots, w_{ij}, \dots, w_{8\,12}, \tau_1, \dots, \tau_j, \dots, \tau_{12}, v_{11}, \dots, v_{ki}, \dots, v_{16\,8}, b_1, \dots, b_i, \dots, b_8). \tag{4.13}$$

The parameters w_{ij}, v_{ki}, τ_j and b_i can be arbitrary real numbers. Eq. (4.12) provides an oversimplified description of a biological neuron and is widely used to construct artificial neural networks [22].

Detailed results regarding the training performance of the algorithms are presented in Table 5. The termination condition is a classification error CE $< 3\%$ [26]; that is, the network classifies correctly 117 out of the 120 patterns.

The successfully trained FNNs are tested for their generalization capability, [22,37], using test patterns from 20 subimages of the same size randomly selected from each image. To evaluate the generalization performance of the FNN the *max* rule is used, i.e., a test pattern is considered to be correctly classified if the corresponding output neuron has the greatest value among the output neurons. The average success rate of classification for each algorithm is exhibited in Table 6.

The results of Table 5 suggest that PR exhibits the best performance of all methods tested regarding the average number of gradient and error function evaluations. On the other hand, ASD, SDAS-2 and GDAM-2 are more robust in the sense that they exhibit larger number of successes providing also good generalization capability (see Table 6).

Table 6
The average classification rate for the texture classification problem

Method	SD	ALS	SDM	ASD	FR	PR	PR–FR	SDAS-2	GDAM-2
Generalization	90.0%	90.0%	90.0%	93.5%	92.0%	92.6%	93.5%	94.0%	94.0%

Table 7
Results for the numeric font learning problem ($n = 460$)

Algorithm	μ_{IT}	μ_{FE}	Success
SD	14 489	28 978	660/1000
ALS	12 225	24 454	990/1000
SDM	10 142	20 284	540/1000
ASD	1975	3950	910/1000
FR	620	3121	420/1000
PR	649	2124	960/1000
PR–FR	750	3473	1000/1000
SDAS-2	253	636	1000/1000
GDAM-2	159	739	1000/1000

Example 4.6 *The numeric font learning problem*, *Magoulas et al.* [26] *and Sperduti and Starita* [38]**.** This experiment refers to the training of a multilayer FNN with 460 variables for recognizing 8×8 pixel machine printed numerals from 0 to 9. The network has 64 input neurons and 10 output neurons representing 0–9. Numerals are given in a finite sequence $C = (c_1, c_2, \ldots, c_p)$ of input–output pairs $c_p = (u_p, t_p)$ where u_p are the binary input vectors in $\mathbb{R}^{64}$ determining the 8×8 binary pixel and t_p are binary output vectors in $\mathbb{R}^{10}$, for $p = 1, \ldots, 10$, determining the corresponding numerals. In order to train the network we have to find variable values that minimize the following objective function:

$$f(x) = \sum_{p=1}^{10} \sum_{j=1}^{10} \left[\left\{ 1 + \exp\left(\sum_{i=1}^{6} w_{ij} y_{i,p} + \tau_j \right) \right\}^{-1} - t_{j,p} \right]^2, \tag{4.14}$$

where

$$y_{i,p} = \left\{ 1 + \exp\left(\sum_{k=1}^{64} v_{ki} u_{k,p} + b_i \right) \right\}^{-1}, \tag{4.15}$$

$$x = (w_{11}, \ldots, w_{ij}, \ldots, w_{6\,10}, \tau_1, \ldots, \tau_j, \ldots, \tau_{10}, v_{11}, \ldots, v_{ki}, \ldots, v_{64\,6}, b_1, \ldots, b_i, \ldots, b_6). \tag{4.16}$$

The termination condition is to locate a minimizer with function value less than or equal to 0.001. The results are summarized in Table 7.

Evidently, SDAS-2 exhibits the best performance. It has 100% success and the smallest average of function evaluations. GDAM-2 achieves faster training than all other methods except SDAS-2. In addition, networks trained with GDAM-2 present better generalization capability than networks trained by other methods [27].

5. Concluding remarks

A new effective and efficient class of gradient-based minimization algorithms has been proposed. These algorithms allow the usage of a common adaptive stepsize for all coordinate directions or an individual adaptive stepsize along each coordinate direction. In both cases, stepsize adaptation is based on estimates of the local Lipschitz constant that are obtained without additional function and gradient evaluations. The convergence of the proposed methods is guaranteed under suitable assumptions. The new methods automatically adapt tuning parameters and ensure that the value of the objective function sufficiently decreases with every iteration.

Our experience with the problems tested indicates that the proposed algorithms behave predictably and reliably.

Acknowledgements

The authors wish to thank the anonymous referees for their constructive comments and valuable suggestions which improved the paper.

References

[1] G.S. Androulakis, M.N. Vrahatis, OPTAC: a portable software package for analyzing and comparing optimization methods by visualization, J. Comput. Appl. Math. 72 (1996) 41–62.

[2] G.S. Androulakis, M.N. Vrahatis, T.N. Grapsa, Studying the performance of optimization methods by visualization, Systems Anal. Model. Simulation 25 (1996) 21–42.

[3] L. Armijo, Minimization of function having Lipschitz continuous first partial derivatives, Pacific J. Math. 16 (1966) 1–3.

[4] O. Axelsson, Iterative Solution Methods, Cambridge University Press, New York, 1996.

[5] E.K. Blum, Approximation of Boolean functions by sigmoidal networks: part I: XOR and other two-variable functions, Neural Comput. 1 (1989) 532–540.

[6] R.P. Brent, Algorithms for Minimization Without Derivatives, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1973.

[7] M.E. Brewster, R. Kannan, Nonlinear successive over-relaxation, Numer. Math. 44 (1984) 309–315.

[8] P. Brodatz, Textures — A Photographic Album for Artists and Designers, Dover, New York, 1966.

[9] R.H. Byrd, J. Nocedal, A tool for the analysis of quasi-Newton methods with application to unconstrained minimization, SIAM J. Numer. Anal. (1989) 727–739.

[10] H.B. Curry, The method of steepest descent for non-linear minimization problems, Quart. Appl. Math. 2 (1944) 258–261.

[11] A. Cauchy, Méthode générale pour la résolution des systèmes d'équations simultanées, Comp. Rend. Acad. Sci. Paris 25 (1847) 536–538.

[12] E.K.P. Chong, S.H. Żak, An Introduction to Optimization, Wiley, New York, 1996.

[13] J.E. Dennis Jr., R.B. Schnabel, Numerical Methods for Unconstrained Optimization and Nonlinear Equations, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1983.

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

[15] J.C. Gilbert, J. Nocedal, Global convergence properties of conjugate gradient methods for optimization, SIAM J. Optim. 2 (1992) 21–42.

[16] A.A. Goldstein, Cauchy's method of minimization, Numer. Math. 4 (1962) 146–150.

[17] A.A. Goldstein, On steepest descent, SIAM J. Control 3 (1965) 147–151.

[18] A.A. Goldstein, J.F. Price, An effective algorithm for minimization, Numer. Math. 10 (1967) 184–189.

[19] T.N. Grapsa, M.N. Vrahatis, A dimension-reducing method for unconstrained optimization, J. Comput. Appl. Math. 66 (1996) 239–253.
[20] A.V. Fiacco, G.P. McCormick, Nonlinear Programming: Sequential Unconstrained Minimization Techniques, SIAM, Philadelphia, 1990.
[21] R. Haralick, K. Shanmugan, I. Dinstein, Textural features for image classification, IEEE Trans. System, Man Cybernet. 3 (1973) 610–621.
[22] S. Haykin, Neural Networks: A Comprehensive Foundation, Macmillan College Publishing Company, New York, 1994.
[23] D. Heller, A survey of parallel algorithms in numerical linear algebra, SIAM Rev. 20 (1978) 740–777.
[24] R.A. Jacobs, Increased rates of convergence through learning rate adaptation, Neural Networks 1 (1988) 295–307.
[25] G.D. Magoulas, M.N. Vrahatis, G.S. Androulakis, A new method in neural network supervised training with imprecision, Proceedings of the IEEE Third International Conference on Electronics, Circuits and Systems, 1996, pp. 287–290.
[26] G.D. Magoulas, M.N. Vrahatis, G.S. Androulakis, Effective backpropagation training with variable stepsize, Neural Networks 10 (1997) 69–82.
[27] G.D. Magoulas, M.N. Vrahatis, G.S. Androulakis, Improving the convergence of the backpropagation algorithm using learning rate adaptation methods, Neural Computation 11 (1999) 1769–1796.
[28] B.J. Moré, B.S. Garbow, K.E. Hillstrom, Testing unconstrained optimization, ACM Trans. Math. Software 7 (1981) 17–41.
[29] J. Nocedal, Theory of algorithms for unconstrained optimization, Acta Numerica 1 (1992) 199–242.
[30] P.P. Ohanian, R.C. Dubes, Performance evaluation for four classes of textural features, Pattern Recognition 25 (1992) 819–833.
[31] J.M. Ortega, W.C. Rheinboldt, Iterative Solution of Nonlinear Equations in Several Variables, Academic Press, New York, 1970.
[32] J.M. Ortega, R.G. Voigt, Solution of partial differential equations on vector and parallel computers, SIAM Rev. 27 (1985) 149–270.
[33] E. Polak, Computational Methods in Optimization, Academic Press, New York, 1971.
[34] E. Polak, Optimization: Algorithms and Consistent Approximations, Springer, New York, 1997.
[35] M.J.D. Powell, Direct search algorithms for optimization calculations, Acta Numerica 7 (1998) 287–336.
[36] S.S. Rao, Optimization, Theory and Applications 2nd Edition, 7th reprint, Wiley Eastern Limited, New Delhi, 1992.
[37] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning internal representations by error propagation, in: D.E. Rumelhart, J.L. McClelland (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1, MIT Press, 1986, pp. 318–362.
[38] A. Sperduti, A. Starita, Speed up learning and network optimization with extended back-propagation, Neural Networks 6 (1993) 365–383.
[39] G.W. Stewart, Introduction to Matrix Computations, Academic Press, New York, 1973.
[40] J. Strang, T. Taxt, Local frequency features for texture classification, Pattern Recognition 27 (1994) 1397–1406.
[41] R. Varga, Matrix Iterative Analysis, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1962.
[42] T.P. Vogl, J.K. Mangis, A.K. Rigler, W.T. Zink, D.L. Alkon, Accelerating the convergence of the back-propagation method, Biol. Cybernet. 59 (1988) 257–263.
[43] R.G. Voigt, Rates of convergence for a class of iterative procedures, SIAM J. Numer. Anal. 8 (1971) 127–134.
[44] M.N. Vrahatis, G.S. Androulakis, G.E. Manoussakis, A new unconstrained optimization method for imprecise function and gradient values, J. Math. Anal. Appl. 197 (1996) 586–607.
[45] L.F. Wessel, E. Barnard, Avoiding false local minima by proper initialization of connections, IEEE Trans. Neural Networks 3 (1992) 899–905.
[46] P. Wolfe, Convergence conditions for ascent methods, SIAM Rev. 11 (1969) 226–235.
[47] P. Wolfe, Convergence conditions for ascent methods II: some corrections, SIAM Rev. 13 (1971) 185–188.
[48] D. Young, Iterative methods for solving partial difference equations of elliptic type, Trans. Amer. Math. Soc. 76 (1954) 92–111.
[49] G. Zoutendijk, Nonlinear programming, computational methods, in: J. Abadie (Ed.), Integer and Nonlinear Programming, North-Holland, Amsterdam, 1970, pp. 37–86.