ELSEVIER

Available at WWW.MATHEMATICSWEB.ORG POWERED BY SCIENCE DIRECT®

Applied Numerical Mathematics 45 (2003) 59–77

APPLIED NUMERICAL MATHEMATICS

www.elsevier.com/locate/apnum

From linear to nonlinear iterative methods

M.N. Vrahatis [a,c,*], G.D. Magoulas [b,c], V.P. Plagianakos [a,c]

[a] *Department of Mathematics, University of Patras, GR-26110 Patras, Greece*
[b] *Department of Information Systems and Computing, Brunel University, Uxbridge UB8 3PH, United Kingdom*
[c] *University of Patras Artificial Intelligence Research Center (UPAIRC), GR-26110 Patras, Greece*

Abstract

This paper constitutes an effort towards the generalization of the most common classical iterative methods used for the solution of linear systems (like Gauss–Seidel, SOR, Jacobi, and others) to the solution of systems of nonlinear algebraic and/or transcendental equations, as well as to unconstrained optimization of nonlinear functions. Convergence and experimental results are presented. The proposed algorithms have also been implemented and tested on classical test problems and on real-life artificial neural network applications and the results to date appear to be very promising.

1. Introduction

An *iterative method* to solve the linear system $Ax = b$ starts with an initial approximation x^0 to the solution x and generates a sequence of vectors $\{x^k\}_{k=0}^{\infty}$ that converges to x. Iterative methods involve a process that converts the system $Ax = b$ into an equivalent system of the form $x = Mx + v$, for some fixed matrix M and vector v. After the initial vector, x^0, is selected, the sequence of approximate solutions is generated by computing $x^{k+1} = Mx^k + v$, for each $k = 0, 1, 2, \ldots$. For large systems containing thousands of equations, iterative methods often have decisive advantages over direct methods in terms of speed and demands on computer memory. Sometimes, if the accuracy requirements are not stringent, a modest number of iterations will suffice to produce an acceptable solution. Also, iterative methods are often very efficient for sparse systems problems. In sparse problems, the nonzero elements of A are sometimes stored in a sparse-storage format. In other case, it is not necessary to store A at all; for example, in problems involving the numerical solution of partial differential equations as, in this case,

* Corresponding author.
E-mail address: vrahatis@math.upatras.gr (M.N. Vrahatis).

0168-9274/02/$30.00
doi:10.1016/S0168-9274(02)00235-0

each row of A might be generated as needed but not retained after use. Another important advantage of iterative methods is that they are usually stable, and they will actually dampen errors, due to roundoff or minor blunders, as the process continues.

The best well-known iterative method for solving a linear system of equations $Ax = b$ is the *Gauss–Seidel* method, which can be extended to nonlinear system of equations.

Thus, if

$$F = (f_1, f_2, \ldots, f_n) : \mathcal{D} \subset \mathbb{R}^n \to \mathbb{R}^n,$$

then the basic step of the *nonlinear Gauss–Seidel* iteration is to solve the ith equation:

$$f_i\left(x_1^{k+1}, \ldots, x_{i-1}^{k+1}, x_i, x_{i+1}^{k}, \ldots, x_n^{k}\right) = 0, \tag{1}$$

for x_i, and to set $x_i^{k+1} = x_i$. To obtain x^{k+1} from x^k, we solve successively the n one-dimensional nonlinear equations (1) for $i = 1, 2, \ldots, n$. If relaxation parameters ω_k are introduced, we may set $x_i^{k+1} = x_i^k + \omega_k(x_i - x_i^k)$ and the corresponding method is called the *nonlinear SOR* or *nonlinear successive overrelaxation* method; in the literature this nomenclature is sometimes reserved for the case $\omega > 1$.

In an analogous way, the kth step of the *nonlinear Jacobi iterative scheme* consists of solving the equations:

$$f_i\left(x_1^{k}, \ldots, x_{i-1}^{k}, x_i, x_{i+1}^{k}, \ldots, x_n^{k}\right) = 0, \qquad i = 1, 2, \ldots, n, \tag{2}$$

for x_i and setting $x_i^{k+1} = x_i$, for $i = 1, 2, \ldots, n$.

In contrast to the linear case, in general, the analytic solutions of Eqs. (1) and (2) are not available and an one-dimensional rootfinding method must be applied that terminates after a suitable number of steps. Any type of one-dimensional methods may be used leading to a large variety of combined methods [16]. On the other hand if many steps of these one-dimensional methods are applied the whole procedure becomes cumbersome and thus in practice, in many cases, one step of these methods is applied. In this case, for example, if we apply Newton's method to Eqs. (1) and (2) we obtain respectively the *one-step SOR-Newton* and the *one-step Jacobi Newton* methods [12]. The convergence properties of all the above methods are well studied and analyzed (see, for example, [12]) and to this end there are many theorems available in the literature.

Furthermore, the above iterative linear schemes can be also extended to unconstrained optimization of a nonlinear function [27,28]. Although the nonlinear iterative rootfinding methods have been extensively studied, the unconstrained optimization case has not been thoroughly studied and analyzed. In this paper, we give some recent convergence and experimental results of ours related to the generalization of the iterative linear methods to unconstrained optimization of nonlinear functions.

The paper is organized as follows. In Section 2 we present the theoretical results as well as the proposed algorithms for computing a local minimizer of a function by generalizing various iterative linear methods. A strategy for developing globally convergent algorithms is presented in Section 3. In Section 4, we exhibit numerical results obtained by the proposed method applied to well-known and widely used test function, as well as on real-life artificial neural network applications. The paper ends in Section 5 with some concluding remarks.

2. Unconstrained optimization of nonlinear functions

It is well-known that a minimizer x^* of a continuous differentiable function f should satisfy the necessary conditions:

$$\nabla f\left(x^*\right) = \Theta^n = (0, 0, \ldots, 0). \tag{3}$$

Eq. (3) represents a set of n nonlinear equations which must be solved to obtain x^*. Therefore, one approach to the minimization of the function f is to seek the solutions of the set of Eq. (3) 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:

$$\begin{aligned} &\partial_1 f(x_1, x_2, \ldots, x_n) = 0, \\ &\partial_2 f(x_1, x_2, \ldots, x_n) = 0, \\ &\qquad\vdots \\ &\partial_n f(x_1, x_2, \ldots, x_n) = 0, \end{aligned} \tag{4}$$

where $\partial_i f(x_1, \ldots, x_i, \ldots, x_n)$ denotes the partial derivative of f with respect to the ith parameter.

Next, we consider the classes of *nonlinear Jacobi* and *nonlinear SOR* methods applied to system (4).

2.1. The composite nonlinear Jacobi method and its convergence

The class of *nonlinear Jacobi* methods is widely used for the numerical solution of system (4). The main feature of the nonlinear Jacobi process is that it is a parallel algorithm [12], i.e., it applies a parallel update of the variables. Starting from an arbitrary initial vector $x^0 \in \mathcal{D}$, one can subminimize at the kth iteration the function:

$$f\left(x_1^k, \ldots, x_{i-1}^k, x_i, x_{i+1}^k, \ldots, x_n^k\right), \tag{5}$$

along the ith direction and obtain the corresponding subminimizer $\hat{x}_i$. Obviously for the subminimizer $\hat{x}_i$

$$\partial_i f\left(x_1^k, \ldots, x_{i-1}^k, \hat{x}_i, x_{i+1}^k, \ldots, x_n^k\right) = 0. \tag{6}$$

This is a one-dimensional subminimization because all the components of the vector x^k, except from the ith component, are kept constant. Then the ith component is updated according to the equation:

$$x_i^{k+1} = x_i^k + \tau_k\left(\hat{x}_i - x_i^k\right), \tag{7}$$

for some relaxation factor τ_k. The objective function in (5) is subminimized in parallel for all i.

Various composite nonlinear Jacobi training algorithms can be obtained depending on the one-dimensional minimization method applied. It is worth noticing that the number of the 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 along each variable direction at each iteration. Moreover, this computational effort is increased for problems with a high number of variables, as, for example, when training neural networks with several hundred network parameters (also called weights). On the other hand, it is not certain that this large computational effort speeds up the minimization process for nonconvex functions when far from a minimizer x^*. Thus, we propose to obtain $\hat{x}_i$ by minimizing the

function (5) with one iteration of a subminimization method. Note that this practice is also suggested for the iterative solution of nonlinear equations [8,12,17,33].

By properly tuning the relaxation factor τ_k, we can obtain better parameter iterates because this factor defines the length of the minimization step along the resultant search direction. Thus, we are able to avoid temporary oscillations and/or to enhance the rate of convergence when the current parameter vector is far from a minimizer.

Next, the convergence of the composite nonlinear Jacobi method is discussed. The convergence analysis is developed under appropriate assumptions and provides useful insight into this class of methods. The objective is to show that there is a neighborhood of a minimizer of the objective function for which convergence to the minimizer can be guaranteed.

Theorem 1. *Let $f:\mathcal{D}\subset\mathbb{R}^n\to\mathbb{R}$ be twice continuously differentiable in an open neighborhood $\mathcal{S}_0\subset\mathcal{D}$ of a point $x^*\in\mathcal{D}$ 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 $\mathcal{S}=\mathcal{S}(x^*,r)$ in $\mathcal{S}_0$ (where $\mathcal{S}(x^*,r)$ denotes the open ball centered at x^* with radius r), such that any sequence $\{x^k\}_{k=0}^{\infty}$ generated by the nonlinear Jacobi process converges to x^* which minimizes f.*

Proof. Consider the decomposition of $H(x^*)$ into its diagonal, strictly lower-triangular and strictly upper-triangular parts:

$$H\big(x^*\big)=D\big(x^*\big)-L\big(x^*\big)-L^{\top}\big(x^*\big). \tag{8}$$

Since $H(x^*)$ has the property A^π, the eigenvalues of

$$\Phi\big(x^*\big)=D\big(x^*\big)^{-1}\big[L\big(x^*\big)+L^{\top}\big(x^*\big)\big],$$

are real and $\rho(\Phi(x^*))<1$ [1] (where $\rho(A)$ indicates the spectral radius of the matrix A); then there exists an open ball $\mathcal{S}=\mathcal{S}(x^*,r)$ in $\mathcal{S}_0$, such that, for any initial vector $x^0\in\mathcal{S}$, there is a sequence $\{x^k\}_{k=0}^{\infty}\subset\mathcal{S}$ which satisfies the nonlinear Jacobi process such that $\lim_{k\to\infty}x^k=x^*$ [12]. Thus the theorem is proved. □

Remark 1. *The Property A^π.* Young [34] has discovered a class of matrices described as having *property A* that can be partitioned into block-tridiagonal form, possibly after a suitable permutation [1]. An algorithmic procedure for transforming a symmetric matrix to a tridiagonal form is presented in [21, p. 335].

Below we synthesize three algorithms of this class. These algorithms employ a different stepsize for each parameter based on traditional one-dimensional minimization methods. The first one requires only the sign of the gradient values, while the other two exploit both the function and gradient values.

2.1.1. The multi-step Jacobi-modified bisection method

In order to compute a minimizer approximation $\hat{x}_i$ in the interval $[a_i,b_i]$ we use our modification of the bisection method which is briefly described below.

A solution of the equation $\varphi(x)=0$, where the function $\varphi:[a,b]\subset\mathbb{R}\to\mathbb{R}$ is continuous, is guaranteed to exist in the interval (a,b) if the following criterion is fulfilled:

$$\varphi(a)\,\varphi(b)<0,\quad\text{or}\quad\operatorname{sgn}\varphi(a)\operatorname{sgn}\varphi(b)=-1,$$

where sgn is the well-known three valued sign function. This criterion is known as Bolzano's existence criterion (for a generalization of this criterion to higher dimensions see [26]). Based on this criterion various rootfinding methods, as, for example, the bisection method, were created. Here we shall use the bisection method which has been modified to the following simplified version described in [24,25]. There is reported that, in order to compute a root of $\varphi(x) = 0$ where $\varphi : [a, b] \subset \mathbb{R} \to \mathbb{R}$ is continuous, a simplified version of the bisection method leads to the following iterative formula:

$$r^{p+1} = r^p + c \cdot \operatorname{sgn} \varphi\left(r^p\right) 2^{p+1}, \quad p = 0, 1, \ldots, \left\lceil \log_2 (b-a)\varepsilon^{-1} \right\rceil, \tag{9}$$

with $c = \operatorname{sgn} \varphi(r^0)(b-a)$, $r^0 = a$, where ε is the required accuracy, and $\lceil \cdot \rceil$ defines the ceiling function. Of course the iterations (9) converge to a root $r^* \in (a, b)$ such that $|r^{p+1} - r^*| \leqslant \varepsilon$ if for some r^p, $p = 1, 2, \ldots$, the following holds:

$$\operatorname{sgn} \varphi\left(r^0\right) \operatorname{sgn} \varphi\left(r^p\right) = -1.$$

Furthermore, the number of iterations ν, which are required in obtaining an approximate root r such that $|r - r^*| \leqslant \varepsilon$ for some $\varepsilon \in (0, 1)$ is given by [24,30]:

$$\nu = \left\lceil \log_2 (b-a)\, \varepsilon^{-1} \right\rceil. \tag{10}$$

Instead of the iterative formula (9) we can also use the following one:

$$r^{p+1} = r^p - c \cdot \operatorname{sgn} \varphi\left(r^p\right) 2^{p+1}, \quad p = 0, 1, \ldots, \left\lceil \log_2 (b-a)\varepsilon^{-1} \right\rceil, \tag{11}$$

where $r^0 = b$.

The reason for using the bisection method is that it is a globally convergent method, it can be easily implemented in parallel and it always converges within the given interval. Moreover it has a great advantage since it is optimal, i.e., it possesses asymptotically the best possible rate of convergence [19]. Also, using the relation (10) it is easy to have beforehand the number of iterations that are required for the attainment of an approximate root to a predetermined accuracy. Finally, it requires only the algebraic signs of the function values to be computed, as it is evident from (9) or (11), thus it can be applied to problems with imprecise function values. As a consequence for problems where the function value follows as a result of an infinite series (e.g., Bessel or Airy functions) it can be shown [29,31] that the sign stabilizes after a relatively small number of terms of the series and the calculations can be speed up considerably.

To compute along the ith direction, a minimizer approximation $\hat{x}_i$ within the interval $[a_i, b_i]$, according to a predefined accuracy $\delta \in (0, 1)$, the above modified bisection method (9) assumes the form:

$$x_i^{p+1} = x_i^p + c \cdot \operatorname{sgn} \partial_i f\left(x^p\right) / 2^{p+1}, \quad p = 0, 1, \ldots, \left\lceil \log_2 \left(h_i\, \delta^{-1}\right) \right\rceil, \tag{12}$$

where $c = \operatorname{sgn} \partial_i f(x^0) h_i$, $x_i^0 = a_i$ and $h_i = b_i - a_i$. Of course, the iterations (12) converge to $\hat{x}_i \in (a_i, b_i)$ if for some x_i^p, $p = 1, 2, \ldots$, the following condition holds:

$$\operatorname{sgn} \partial_i f\left(x^0\right) \operatorname{sgn} \partial_i f\left(x^p\right) = -1.$$

To ensure that $\hat{x}_i$ is a subminimizer along the ith variable's direction, we choose the endpoints a_i and b_i in such a way that the ith component of the gradient vector at the left endpoint a_i has a negative value, or, the ith component of the gradient vector at the right endpoint b_i has a positive value. In order this condition to be fulfilled we choose the endpoints by means of the following relation:

$$a_i = x_i^k - \frac{1}{2}\left\{1 + \operatorname{sgn} \partial_i f\left(x^k\right)\right\} h_i - \operatorname{sgn} \partial_i f\left(x^k\right) \beta, \qquad b_i = a_i + h_i, \tag{13}$$

where β is a small positive real number which depends on the relative machine precision (e.g., $\beta = x_i\sqrt{epsmch}$ where *epsmch* denotes the relative machine precision).

It is evident from the sequence (12) that the only computable information required by this method is the algebraic signs of the gradient of the function f. The method does not require the gradient values to be evaluated analytically. It allows to minimize storage requirements regarding gradient and approximate the derivatives using the method of forward-differences, as the sign can be obtained accurately by comparing the relative size of the function value. Hence, the convergence of the method is not affected by rounding and quantization errors that usually cause imprecise function values in simulations, as long as the signs are preserved.

Thus, the value $\hat{x}_i$ in the iterative scheme (7) is the approximation of the subminimizer obtained by (12). For the implementation and the good performance characteristics of (12) in neural networks training with imprecision see [9].

2.1.2. The one-step Jacobi–Newton method and a heuristic scheme

A straightforward implementation of the one-step Jacobi–Newton iteration leads to the following iterative scheme:

$$x_i^{k+1} = x_i^k - \tau_k \frac{\partial_i f(x^k)}{\partial_{ii}^2 f(x^k)}. \tag{14}$$

To avoid the calculation of the second derivative we propose the following scheme which utilizes the notion of the local Lipschitz constant [10]:

$$\Lambda_i^k = \left|\partial_i f\left(x^k\right) - \partial_i f\left(x^{k-1}\right)\right| / \left|x_i^k - x_i^{k-1}\right|, \tag{15}$$

where x^k and x^{k-1} is a pair of consecutive iterates. Thus, the parameters are updated according to the relation:

$$x_i^{k+1} = x_i^k - \tau_k \left\{ \frac{|\partial_i f(x^k) - \partial_i f(x^{k-1})|}{|x_i^k - x_i^{k-1}|} \right\}^{-1} \partial_i f\left(x^k\right). \tag{16}$$

In the steep regions of the objective function's landscape Eq. (16) uses a small value for the stepsize in order to guarantee convergence. On the other hand, when the landscape of the objective function has flat regions, a large stepsize is used to accelerate the convergence. Note that in Eq. (16) the partial derivative with respect to the ith component $\partial_i f(x^{k-1})$ is used. Thus, proper initialization is needed to start the iterative procedure.

2.1.3. The one-step Jacobi with Newton-update method

The problem of minimizing the function f along the ith direction at the kth iteration:

$$\eta_i^* = \min_{\eta_i \geqslant 0} f\left(x^k + \eta_i\, e_i\right), \tag{17}$$

where e_i indicates the ith column of the identity matrix, is equivalent to seeking the value of η_i^* that minimizes the one-dimensional function:

$$\phi_i(\eta_i) = f\left(x^k + \eta_i\, e_i\right). \tag{18}$$

In situations where $f(x) \geqslant 0$, for all $x \in \mathbb{R}^n$, as in the neural network training problem, then the point x^* with $f(x^*) = 0$ minimizes $f(x)$. Therefore the subminimization problem (17) can be handled by applying properly a root finding procedure to the equation:

$$\phi_i(\eta_i) = 0, \tag{19}$$

to obtain an approximation $\hat{\eta}_i$ of η_i^*. To this end, using one step of the Newton's method we obtain:

$$\eta_i^1 = \eta_i^0 - \frac{\phi_i(\eta_i^0)}{\phi_i'(\eta_i^0)}. \tag{20}$$

Since $\eta_i^0 = 0$ we get:

$$\hat{\eta}_i = -\frac{\phi_i(\eta_i^0)}{\phi_i'(\eta_i^0)}. \tag{21}$$

From Eq. (18) we have:

$$\phi_i\left(\eta_i^0\right) = f\left(x^k + \eta_i^0\, e_i\right) = f\left(x^k\right), \tag{22}$$

and

$$\phi_i'\left(\eta_i^0\right) = \nabla f\left(x^k\right)^\top e_i = \partial_i f\left(x^k\right). \tag{23}$$

Thus, Eq. (21) is reformulated as:

$$\hat{\eta}_i = -\frac{f(x^k)}{\partial_i f(x^k)}. \tag{24}$$

The value of $\hat{\eta}_i$ corresponds to the difference $(\hat{x}_i - x_i^k)$ of Eq. (7) and is calculated in parallel for all the components $(i = 1, \ldots, n)$ at each iteration. Consequently, Eq. (7) takes the form:

$$x_i^{k+1} = x_i^k - \tau_k \frac{f(x^k)}{\partial_i f(x^k)}. \tag{25}$$

The iterative scheme (25) takes into consideration information from both the objective function and the magnitude of the gradient components. When the gradient magnitude is small, the local shape of f in this direction is flat, otherwise it is steep. The value of the objective function indicates how close to the minimizer this local shape is. The above pieces of information help the iterative scheme (25) to escape from flat regions with high function values, which are located far from a desired minimizer. Another advantage of this scheme is that there is no need to store information from the previous iterations, e.g., previous values of the objective function and/or of the gradient.

2.2. The composite nonlinear SOR scheme and its convergence

Starting from an arbitrary initial iterate $x^0 \in \mathcal{D}$, the nonlinear SOR scheme subminimizes at the kth iteration the function:

$$f\left(x_1^{k+1}, \ldots, x_{i-1}^{k+1}, x_i, x_{i+1}^k, \ldots, x_n^k\right), \tag{26}$$

along the ith direction and obtain the corresponding subminimizer $\hat{x}_i$. Again in this case, the ith component is updated according to Eq. (7). The main difference from the Jacobi scheme is that the adaptation of the x_i at the kth iteration takes into consideration all the previously updated variables of the same iteration. The convergence result for the nonlinear SOR scheme is as follows:

Theorem 2. *Let $f:\mathcal{D}\subset\mathbb{R}^n\to\mathbb{R}$ be twice continuously differentiable on an open neighborhood $\mathcal{S}_0\subset\mathcal{D}$ of a local minimizer $x^*\in\mathcal{D}$. Then there exists an open ball $\mathcal{S}=\mathcal{S}(x^*,r)$ in $\mathcal{S}_0$ such that the sequence $\{x^k\}$ generated by the nonlinear SOR scheme converges to the point x^*.*

Proof. Suppose that:

$$\Phi_\tau\big(x^*\big)=\big[D\big(x^*\big)-\tau L\big(x^*\big)\big]^{-1}\big[(1-\tau)D\big(w^*\big)+\tau L^\top\big(x^*\big)\big],$$

for $\tau\in(0,2)$ where D and L are defined as in Eq. (8). Now, by virtue of Ostrowski Theorem [22], $\rho(\Phi_\tau(x^*))<1$ for any $\tau\in(0,2)$ and therefore, by the nonlinear SOR theorem [12], there exists an open ball $\mathcal{S}=\mathcal{S}(x^*,r)$ in $\mathcal{S}_0$, such that, for any $x^0\in\mathcal{S}$, $\lim_{k\to\infty}x^k=x^*$. Thus the theorem is proved. □

2.2.1. The multi-step SOR-modified bisection method

The m-step SOR bisection method is similar to m-step Jacobi bisection method. However, to avoid calculating the signs of the gradient values in (12), we propose an alternative scheme:

$$\hat{x}_i^{p+1}=\hat{x}_i^p+c\cdot\operatorname{sgn}\big(f\big(z_i^p\big)-f(z_i)\big)/2^{p+1},\quad p=0,1,2,\ldots \tag{27}$$

where z_i^p, z_i are computed by means of

$$z_i^p=\big(x_1^{k+1},\ldots,x_{i-1}^{k+1},\hat{x}_i^p,x_{i+1}^k,\ldots,x_n^k\big), \tag{28}$$

$$z_i=\big(x_1^{k+1},\ldots,x_{i-1}^{k+1},x_i^k,x_{i+1}^k,\ldots,x_n^k\big), \tag{29}$$

with $c=\operatorname{sgn}(f(z_i^0)-f(z_i))\,h_i$; $h_i=b_i-a_i$, $\hat{x}_i^0=a_i$.

Suppose that the sequence (27) converges to a $\hat{x}_i^\alpha$ then the final approximation to $\hat{x}_i$ is given by:

$$\hat{x}_i=\hat{x}_i^0+\tau_k\big(\hat{x}_i^\alpha-\hat{x}_i^0\big), \tag{30}$$

for some relaxation factor τ_k.

Also, the sign of the gradient in relation (13) can be computed as

$$\operatorname{sgn}\partial_i f\big(x^k\big)=\operatorname{sgn}\big(f\big(x^k+\beta e_i\big)-f\big(x^k\big)\big),$$

where β is a small positive real number which depends on the relative machine precision (e.g., $\beta=x_i^k\sqrt{epsmch}$).

The signs of the objective function values or its gradient values in the iterative scheme (27) can be achieved by solely comparing the relative sizes of the objective function values. Thus, this method seems able to cope with imprecisions or noisy function values.

2.2.2. Another approach based on Powell's method

Here, we briefly describe Powell's method for solving unconstrained minimization problems, without calculating derivatives. We also propose a derivative free minimization method, which is based on Powell's method, and study its termination properties. Powell's method [15] is based on the use of conjugate directions and the main idea of his approach is that the minimum of a positive-definite quadratic form can be found by performing at most n successive exact line searches along mutually conjugate directions, where n is the number of variables. Also, this procedure can be applied to non-quadratic functions by adding a new composite direction at the end of each cycle of n exact line searches. In this case finite termination is no longer expected. One iteration of Powell's method consists of the following steps, where x^0 is the initial point, and u_i, $i=1,2,\ldots,n$ determine the initial set of directions which are equal to the basis vectors e_i, $i=1,2,\ldots,n$:

1. **For** $i = 1, 2, \ldots, n$ **Compute** λ_i to minimize $f(x^{i-1} + \lambda_i u_i)$, and
   Define $x^i = x^{i-1} + \lambda_i u_i$.
2. **For** $i = 1, 2, \ldots, n-1$, **Replace** u_i by u_{i+1}.
3. **Replace** u_n by $(x^n - x^0)$.
4. **Compute** λ to minimize $f(x^n + \lambda u_n)$, and **Set** $x^0 = x^n + \lambda u_n$.

For the subminimization process along the u_i direction, Powell's method uses a subprocedure that calculates the minimum of a quadratic form and it is primarily based on the quadratic defined by three function values. Let p be a point along the search direction u_i and q a steplength. Initially, $f(p)$ and $f(p + qu_i)$ are calculated, and then either $f(p - qu_i)$ or $f(p + 2qu_i)$ is worked out, depending on whether $f(p)$ is less than or greater than $f(p + qu_i)$. These three function values are now used in the general formula which predicts the turning value of the quadratic defined by the points $a, f(p + au_i), b, f(p + bu_i), c$ and $f(p + cu_i)$ to be at $(p + du_i)$, where:

$$d = \frac{1}{2} \times \frac{(b^2 - c^2) f_a + (c^2 - a^2) f_b + (a^2 - b^2) f_c}{(b - c) f_a + (c - a) f_b + (a - b) f_c}. \tag{31}$$

For some technical details needed to find the minimum as well as techniques that reduce the number of function evaluations, see the detailed description of the algorithm in [15].

For a general (non-quadratic) function, the iteration is repeated until some stopping criterion is satisfied. If f is quadratic we minimize along n conjugate directions $u_1, u_2, \ldots, u_n$, and the minimum is reached if the u_i are all nonzero. This is true if $\lambda_1 \neq 0$ at each iteration, since the directions $u_1, u_2, \ldots, u_n$ cannot become linearly dependent. The problem is that Powell's method has a tendency to choose the search directions that are linearly dependent on each other, especially in ill-conditioned problems. There are various procedures to cope with the linear dependence in Powell's algorithm. The simplest way to avoid linear dependence of the search directions is to reset the set of directions u_i, to the basis vectors e_i after n or $(n + 1)$ iterations of the basic procedure. This is the procedure we will follow in our approach and it retains the quadratic termination property, if $\lambda_1 \neq 0$. Below, we propose a modification of Powell's method for the numerical computation of a minimizer of f utilizing only the relative size of the function values. To this end, $f(x + \lambda u)$ is minimized by comparing the relative size of function values.

In our case we have to minimize $f(x^0 + \lambda u)$ along the line u. One way to do this, by applying one-dimensional rootfinding methods, is to compute the value of $\lambda \neq 0$ such that:

$$f\left(x^0 + \lambda u\right) - f\left(x^0\right) = 0. \tag{32}$$

Now, if $\hat{\lambda}$ is the solution of the above equation, then, of course, the point $\hat{x}^0 = x^0 + \hat{\lambda} u$ possesses the same function value as the point x^0, so it belongs to the contour line of x^0. Then, we can choose a point which belongs to the line segment with endpoints x^0 and $\hat{x}^0$ possessing a smaller function value than these endpoints. With this fact in mind we can now choose such a point, say for example:

$$x^1 = x^0 + \gamma\left(\hat{x}^0 - x^0\right), \quad \gamma \in (0, 1).$$

To solve the one-dimensional Eq. (32), we use our modification of the bisection method. Thus, in our case to solve Eq. (32) for λ, along the direction u, the modified bisection method (9) assumes the form:

$$\lambda^{p+1} = \lambda^p + c \cdot \operatorname{sgn}\left[f\left(a + \lambda^p u\right) - f\left(x^0\right)\right] / 2^{p+1}, \quad p = 0, 1, \ldots, \nu, \tag{33}$$

with $\nu = \lceil \log_2 (b - a)\varepsilon^{-1} \rceil$, $c = \operatorname{sgn}[f(a) - f(x^0)](b - a)$, $\lambda^0 = 0$, $h = b - a$ and where h indicates the stepsize. Of course, utilizing our process we are able to obtain also local maxima along the line u. But if

we choose the endpoints a and b in a proper way, then this method deals with a minimum. This can be easily handled by applying the iterative scheme (33) and taking the endpoints a and b from the following relations:

$$a = x^0 - s\beta - \frac{1}{2}(1+s)h \quad \text{and} \quad b = a + h, \tag{34}$$

where $s = \operatorname{sgn}[f(x^0 + \beta u) - f(x^0)]$ and β is a small positive real number which depends on the relative machine precision (e.g., $\beta = |a|\sqrt{epsmch}$).

To study the termination properties of our approach we give the following result that states that any search method involving minimization along a set of linearly independent conjugate directions has a quadratically termination property.

Theorem 3 [15,37]. *If a quadratic function $f(x)$ of dimension n is minimized sequentially, once along each direction of a set of n linearly independent, conjugate directions, the global minimum of f will be located in n or less cycles independent of the starting point as well as the order in which the minimization directions are used.*

Remark 2. A method that minimizes f according to the requirements of the above theorem has the property known as *quadratic termination*. Also, the order in which the directions are used is immaterial.

Theorem 4 [15,37]. *The directions generated in Powell's method are conjugate.*

Theorem 5. *The proposed method locates the minimum of an n-dimensional quadratic function $f(x)$, in n or less iterations, utilizing only the relative size of the function values of f, independent of the starting point as well as the order in which the minimization directions are used.*

Proof. Since the proposed method uses the direction set u_i of Powell's method, by Theorem 4 these directions are conjugate. Also, the starting search directions are coordinate directions and hence they are linearly independent. Furthermore, the proposed method avoids linear dependence of the search directions by resetting the set of directions u_i, to the coordinate directions after n or $(n+1)$ iterations of the basic procedure. Thus the assumptions of Theorem 3 are fulfilled and the result follows. □

As it is well-known the quadratic termination property is very powerful, because most of the general (non-quadratic) functions can be approximated very closely (near their minima) by a quadratic one. Thus, this property is expected to speed up the convergence even for general functions. On the other hand Powell's method, as well as the proposed one, require generally more than, the theoretically estimated number of n cycles for quadratic functions. The proof of the quadratic termination property has been established with the assumption that the exact minimum is found in each of the one-dimensional subminimizations. In practice the actual subminimizer is approximated and hence the subsequent directions will not be conjugate. Thus, the methods require more iterations for achieving the overall convergence. For a proof of convergence, when the directions of the basis vectors are used and inexact one-dimensional subminimization procedures are applied, see [27, p. 377].

3. A strategy for developing globally convergent algorithms

In this section we present a strategy for developing globally convergent algorithms, i.e., algorithms with the property that starting from almost any starting point the sequence of the iterates will converge to a local minimizer of the objective function. This strategy is similar to the nonlinear Jacobi approach, since it utilizes approximations of the subminimizers in each coordinate direction, and is a parallel algorithm. The theoretical result presented below, allows us to equip the algorithms with a strategy for adapting the direction of search to a descent one. In this way, a decrease of the function values at each iteration is ensured, and convergence to a local minimizer of the objective function is obtained from remote initial points.

Theorem 6. *Suppose that*: (a) $f:\mathbb{R}^n \to \mathbb{R}$ *is bounded below in* $\mathbb{R}^n$, (b) *the gradient is Lipschitz continuous, i.e., there exists a constant* $L > 0$ *such that*

$$\left\|\nabla f(x) - \nabla f(y)\right\| \leqslant L\|x - y\|, \quad \forall x, y \in \mathcal{N},$$

and (c) *that* f *is continuously differentiable in a neighborhood* $\mathcal{N}$ *of the level set* $\mathcal{L} = \{x\colon\ f(x) \leqslant f(x^0)\}$, *where* x^0 *is the starting point of the iterative scheme*:

$$x^{k+1} = x^k + \alpha^k d^k, \tag{35}$$

where $d^k = -\operatorname{diag}\{\tau_1^k, \ldots, \tau_i^k, \ldots, \tau_n^k\}\nabla f(x^k)$ *denotes the search direction, and* $\tau_m^k,\ m = 1, 2, \ldots, i-1, i+1, \ldots, n$ *are arbitrarily chosen small positive stepsizes,*

$$\tau_i^k = -\frac{\zeta}{\partial_i f(x^k)} - \frac{1}{\partial_i f(x^k)} \sum_{\substack{j=1 \\ j\neq i}}^{n} \tau_j^k \partial_j f\left(x^k\right), \quad 0 < \zeta \ll \infty, \quad \partial_i f\left(x^k\right) \neq 0, \tag{36}$$

and $\alpha^k > 0$ *satisfies the Wolfe's conditions*:

$$f\left(x^k + \alpha^k d^k\right) - f\left(x^k\right) \leqslant \sigma_1 \alpha^k \nabla f\left(x^k\right)^\top d^k, \tag{37}$$

$$\nabla f\left(x^k + \alpha^k d^k\right)^\top d^k \geqslant \sigma_2 \nabla f\left(x^k\right)^\top d^k, \tag{38}$$

where $0 < \sigma_1 < \sigma_2 < 1$. *Then the sequence* $\{x^k\}_{k=0}^{\infty}$, *generated by the iterative scheme* (35) *is globally convergent to a local minimizer of the objective function* f.

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

$$d^k = -\operatorname{diag}\left\{\tau_1^k, \ldots, \tau_i^k, \ldots, \tau_n^k\right\}\nabla f\left(x^k\right),$$

which is a descent direction if $\tau_m^k,\ m = 1, 2, \ldots, i-1, i+1, \ldots, n$ are arbitrarily chosen real positive step lengths and τ_i^k is given by relation (36), since

$$\nabla f\left(x^k\right)^\top d^k < 0.$$

Moreover, the Zoutendijk condition [38]

$$\sum_{k\geqslant 1} \cos^2\theta_k \left\|\nabla f\left(x^k\right)\right\|^2 < \infty, \tag{39}$$

where

$$\cos\theta_k = \frac{-\nabla f(x^k)^\top d^k}{\|\nabla f(x^k)\|\,\|d^k\|}, \tag{40}$$

is fulfilled [35,36,38]. In our case relation (40) becomes

$$\cos\theta_k = \frac{-\nabla f(x^k)^\top d^k}{\|\nabla f(x^k)\|\,\|d^k\|} > 0, \tag{41}$$

thus $\lim_{k\to\infty}\|\nabla f(x^k)\| = 0$, which means that the sequence of gradients converges to zero. From the previous it is evident that the sequence $\{x^k\}_{k=0}^{\infty}$ is globally convergent to a local minimizer. Thus, the theorem is proved. □

Remark 3. Theorem 6 guarantees convergence to a local minimizer for any minimization algorithm that adopt the following strategy: (a) define $(n-1)$, say $\{1,2,\ldots,i-1,i+1,\ldots,n\}$, out of the n step lengths, of the set $\{1,2,\ldots,n\}$, as computed directly by the algorithm, and (b) analytically calculate the remaining one (the ith step length) using the values of the others, $\{1,2,\ldots,i-1,i+1,\ldots,n\}$. Note that no additional objective function or gradient evaluations are required; the proposed strategy uses pieces of information that are already computed.

Note that in many applications such as neural network training and nonlinear least square problems, the objective function f is always bounded below, thus condition (a) of Theorem 6 is always fulfilled.

4. Numerical results

The proposed 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 problems

The procedures described in Section 2.2.2, have been implemented and tested in two test functions. Our modified version of Powell's method (SIGNOPT) has been compared with two other well-known minimization methods, namely Powell's and Rosenbrock's methods. To study the influence of imprecise information (regarding the values of the objective function), we simulate imprecisions with the following approach: information about $f(x)$ is obtained in the form of $f^\sigma(x)$, where $f^\sigma(x)$ is an approximation to the true function value $f(x)$, contaminated by a small amount of noise. For the test problems, the reported parameters are: n, the dimension of the objective function; σ, the value of the standard deviation of the simulated noise; and $x^0 = (x_1, x_2, \ldots, x_n)$, the classical starting point for each function.

4.1.1. Broyden banded function [11]

In this example f is given by:

$$f(x) = \sum_{i=1}^{n} {f_i}^2(x), \quad \text{with } f_i(x) = x_i\left(2+5x_1^2\right) + 1 - \sum_{j\in J_i} x_j(1+x_j),$$

Table 1
Function evaluations (Broyden banded function)

n	σ	Powell	Rosenbrock	SIGNOPT
2	0	246	599	536
	0.01	1549	12437	964
	0.10	9369	20524	1289
	0.20	–	15819	1180
	0.30	–	30756	1824
3	0	2182	1426	1002
	0.01	–	43654	3429
	0.10	–	42231	2574
	0.20	–	90159	4000
	0.30	–	–	9289

where $J_i = \{j\colon\ j \neq i,\ \max(1, i - m_l) \leqslant j \leqslant \min(n, i + m_u)\}$ and $m_l = 5,\ m_u = 1$. For $n = 2$, we have used the starting values $x^0 = (1, 1)$, while for $n = 3$ we have started the methods from the point $x^0 = (1, 1, 1)$. In Table 1 we exhibit the obtained results. SIGNOPT converged in all cases and had predictable performance, while Powell's method diverged as we increased σ and Rosenbrock's method exhibited slow convergence.

4.1.2. Hilbert function [2]

In this example f is given by:

$$f(x) = x^\top A x, \quad a_{ij} = \frac{1}{i + j - 1}, \quad \text{for } 1 \leqslant i, j \leqslant n,$$

where A is an n by n Hilbert matrix. This is a positive definite quadratic function, but its condition number increases rapidly with n. In our tests, the starting point was $x^0 = (1, 1)$ for $n = 2$, $x^0 = (1, 1, 1)$ for $n = 3$ and $x^0 = (1, 1, 1, 1)$ for $n = 4$. In Table 2 we exhibit the corresponding results obtained by the methods. Powell's method for $n = 4$ has shown a quite random behavior and Rosenbrock's method had a slow convergence. Once again, SIGNOPT outperformed the other two methods; was faster and more predictable.

4.2. Neural networks training

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

$$f(x) = \sum_{p=1}^{P} \sum_{j=1}^{J} \left[\left\{ 1 + \exp\left(\sum_{i=1}^{I} w_{ij} y_{i,p} + \tau_j \right) \right\}^{-1} - t_{j,p} \right]^2, \tag{42}$$

where

$$y_{i,p} = \left\{ 1 + \exp\left(\sum_{k=1}^{K} v_{ki} u_{k,p} + b_i \right) \right\}^{-1}, \quad \text{and}$$
$$x = (v_{11}, \ldots, v_{ki}, \ldots, v_{KI}, b_1, \ldots, b_i, \ldots, b_I, w_{11}, \ldots, w_{ij}, \ldots, w_{IJ}, \tau_1, \ldots, \tau_j, \ldots, \tau_J). \tag{43}$$

Table 2
Function evaluations (Hilbert function)

n	σ	Powell	Rosenbrock	SIGNOPT
2	0	106	205	322
	0.01	1151	1213	432
	0.10	–	2411	538
	0.20	–	38882	324
	0.30	–	65861	1825
3	0	439	507	428
	0.01	2523	3553	716
	0.02	3370	3814	716
	0.03	11370	3514	716
	0.04	15221	3623	858
	0.05	15598	4343	858
	0.10	24533	5040	4225
4	0	882	717	533
	0.01	6160	3810	359
	0.02	7174	4426	359
	0.03	4297	7136	359
	0.04	9757	6659	359
	0.05	1624	9373	538
	0.10	4505	11332	4824

The parameters v_{ki}, b_i, w_{ij} and τ_j can be arbitrary real numbers. Eq. (43) provides an oversimplified description of a biological neuron and it is widely used to construct artificial neural networks [18, 6]. Examples used for training the network are presented 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 can either be real or binary valued input vectors in $\mathbb{R}^K$ and t_p are real or binary output vectors in $\mathbb{R}^J$, for $p = 1, \ldots, P$, determining the corresponding training pattern.

Next, we give quantitative results applying various methods in four neural network applications:

(i) a variant of the Steepest Descent with constant stepsize (SD) [18];
(ii) the Steepest Descent with Line Search (SDLS) [14, p. 30];
(iii) a modification of the Steepest Descent with constant stepsize and Momentum (SDM) [18];
(iv) an Adaptive Steepest Descent with heuristics for tuning the stepsize (ASD) [32];
(v) the Fletcher–Reeves (FR) method [4];
(vi) the Polak–Ribiere (PR) method [4];
(vii) the Polak–Ribiere (PR) method constrained by the FR method (PR–FR) [4];
(viii) the Heuristic Jacobi–Newton method (HJN) of Eq. (16);
(ix) the one-step Jacobi with Netwon–Update method (JNU) of Eq. (25); and
(x) the Multi-step SOR-modified bisection method (m-SOR) of Eq. (30).

Note that in the implementation of FR, PR, PR–FR, the line search of [14] has been used. Methods testing has been conducted using a set of 1000 randomly chosen initial points.

Table 3
Results for the XOR problem ($n = 9$)

Algorithm	μ_{GRD}	μ_{FE}	μ_{ASE}	Success
SD	549	549	n/a	810/1000
SDLS	64	371	n/a	810/1000
SDM	803	803	n/a	810/1000
ASD	157	157	n/a	810/1000
FR	84	198	n/a	130/1000
PR	21	148	n/a	380/1000
PR–FR	22	149	n/a	410/1000
HJN	52	182	n/a	810/1000
m-SOR	n/a	n/a	193	440/1000

4.2.1. The XOR problem [18]

The classification of the four XOR patterns in two classes is an interesting problem because it is sensitive to initial conditions as well as to stepsize variations, and presents a multitude of local minima. The binary patterns are presented to the network 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 $\mathbb{R}^2$ determining the binary input pattern and t_p are binary output vectors in $\mathbb{R}^1$, for $p = 1, \ldots, 4$, determining the corresponding number of patterns. A neural network with 9 variables is used for this classification task.

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 3, where μ_{GRD} denotes the mean number of gradient evaluations, μ_{FE} denotes the mean number of objective function evaluations required to obtain convergence, *Success* shows the number of successful simulations out of 1000 runs, i.e., in the successful runs the iterates converge to a minimizer with function value less than or equal to 0.04, and μ_{ASE} is the mean number of algebraic sing evaluations required by the m-SOR.

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. HJN exhibits better performance than FR, PR and PR–FR with regards to the number of successful runs. HJN also outperforms SD, SDLS, SDM and FR in training speed, measured by the mean number of function and gradient evaluations needed to successfully classify the patterns. Note that PR and PR–FR require less function evaluations than HJN but they reveal a smaller number of successful runs. It is worth noticing that the m-SOR compares favorably to the conjugate gradient methods in terms of successes. In addition, m-SOR does not require gradient evaluations.

4.2.2. Function approximation problem [23]

This experiment concerns the approximation of the function $f(x) = \sin(x)\cos(2x)$ with domain $0 \leqslant x \leqslant 2\pi$ using 20 input–output points, i.e., u_p are real-valued input vectors in $\mathbb{R}^1$ determining the input point and t_p are real-valued output points in $\mathbb{R}^1$, for $p = 1, \ldots, 20$, determining the corresponding number of points. A neural network with 31 variables that is based on hidden neurons of hyperbolic tangent activations and on a linear output neuron is used [23]. Training is considered successful when $E \leqslant 0.0125$. Comparative results are shown in Table 4, where the abbreviations are as in Table 3.

Table 4
Comparative results for the function approximation problem ($n = 31$)

Algorithm	μ_{GRD}	μ_{FE}	μ_{ASE}	Success
SD	1588720	1588720	n/a	1000/1000
SDM	578848	578848	n/a	1000/1000
ASD	388457	388457	n/a	1000/1000
SDLS	886364	1522890	n/a	1000/1000
HJN	198172	311773	n/a	1000/1000
m-SOR	n/a	n/a	46995	1000/1000

Table 5
Results for the numeric font learning problem ($n = 460$)

Algorithm	μ_{GRD}	μ_{FE}	μ_{ASE}	Success
SD	14489	14489	n/a	660/1000
SDLS	12225	12229	n/a	990/1000
SDM	10142	10142	n/a	540/1000
ASD	1975	1975	n/a	910/1000
FR	620	2501	n/a	420/1000
PR	649	1475	n/a	960/1000
PR–FR	750	2723	n/a	1000/1000
HJN	159	581	n/a	1000/1000
JNU	1361	3708	n/a	1000/1000
m-SOR	n/a	n/a	3696	1000/1000

4.2.3. The numeric font learning problem [20]

This experiment refers to the training of a multilayer neural network 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 through 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. The termination condition is to locate a minimizer with function value less than or equal to 0.001. The results are summarized in Table 5 using the same notation as in Table 3. Evidently, HJN exhibits the best performance. It has 100% success and the smallest average of function evaluations. HJN achieves faster training than all other methods.

4.2.4. Texture classification problem

In this experiment a set of 12 Brodatz texture images [3]: 3, 5, 9, 12, 15, 20, 51, 68, 77, 78, 79, 93 of size 512×512 is acquired by a scanner at 150dpi. From each texture image 10 subimages of size 128×128 are randomly selected, and the co-occurrence method, introduced by Haralick et al. [5] 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. The combination of the nearest neighbor pairs at orientations 0°, 45°, 90° and 135° are used in the experiment. 10 sixteenth-dimensional training patterns are created from each image and the network is trained to classify the patterns to 12 texture types. Patterns 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 real-valued input vectors in $\mathbb{R}^{16}$ and t_p are binary output vectors in $\mathbb{R}^{12}$, for $p = 1, \ldots, 120$, determining the

Table 6
Comparative results for the texture classification problem ($n = 244$)

Algorithm	μ_{GRD}	μ_{FE}	Success	μ_{CS}
SD	15893	15893	960/1000	90%
SDLS	13256	13261	965/1000	90%
SDM	12422	12422	940/1000	90%
ASD	560	560	1000/1000	93%
FR	1624	11050	250/1000	92%
PR	140	670	990/1000	92%
PR–FR	145	860	996/1000	93%
HJN	382	591	1000/1000	94%
JNU	791	2185	1000/1000	93%

number of patterns in the training set. The termination condition is a classification error $CE < 3\%$. The successfully trained networks are tested for their classification ability using patterns from 20 subimages of the same size randomly selected from each image, i.e., the test set consists of 240 patterns. To evaluate the classification performance of a network 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.

Detailed results regarding the training performance of the algorithms are presented in Table 6, where μ_{GRD} denotes the mean number of gradient evaluations, μ_{FE} denotes the mean number of objective function evaluations required to obtain convergence, *Success* shows the number of successful simulations out of 1000 runs, i.e., in the successful runs the iterates converge to a minimizer that provides a suitable classification error, $CE < 3\%$, and μ_{CS} is the percentage of classification success in testing. The results of Tables 6 suggest that HJN significantly outperforms other methods in terms of overall performance, i.e., number of gradient and error function evaluations as well as in the percentage of successful simulations. Almost similar performance is exhibited by ASD, however ASD requires fine tuning five heuristic parameters as well as the stepsize. Both ASD and JNU provide good classification success, which is considered as a very critical factor when choosing training algorithms.

5. Conclusions and further research

In this paper, an investigation on the generalization of the most common classical iterative methods used for the solution of linear systems (like Gauss–Seidel, SOR, Jacobi, and others) to the unconstrained optimization of nonlinear functions has been conducted. Although the nonlinear iterative rootfinding methods have been extensively studied, the unconstrained optimization case has not been thoroughly analyzed. Thus, in this work unconstrained optimization algorithms for nonlinear functions based on generalizations of iterative linear methods were introduced. Theoretical convergence results for the proposed algorithms have been derived for computing a local minimizer of a function. A strategy for developing globally convergent modifications of these algorithms has also been proposed. The new algorithms have been implemented and tested on classical test problems and on real-life artificial neural network applications and the results to date appear to be very promising. In a subsequent communication we intend to implement in parallel the methods of the Jacobi class, using the Parallel Virtual Machine (PVM) [7]. Preliminary results indicate that utilizing PVM, the speed up achieved is analogous to the number of the processors used [13], thus considerably shorten the minimization process time.

References

[1] O. Axelsson, Iterative Solution Methods, Cambridge University Press, New York, 1996.
[2] R.P. Brent, Algorithms for Minimization Without Derivatives, Prentice-Hall, Englewood Cliffs, NJ, 1973.
[3] P. Brodatz, Textures—A Photographic Album for Artists and Designer, Dover, New York, 1966.
[4] J.C. Gilbert, J. Nocedal, Global convergence properties of conjugate gradient methods for optimization, SIAM J. Optimization 2 (1992) 21–42.
[5] R. Haralick, K. Shanmugan, I. Dinstein, Textural features for image classification, IEEE Trans. System Man. Cybernetics 3 (1973) 610–621.
[6] S. Haykin, Neural Networks: A Comprehensive Foundation, Macmillan, New York, 1994.
[7] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, V. Sunderam, PVM: Parallel Virtual Machine. A User's Guide and Tutorial for Networked Parallel Computing, MIT Press, Cambridge, MA, 1994.
[8] C.T. Kelley, Iterative Methods for Linear and Nonlinear Equations, SIAM, Philadelphia, PA, 1995.
[9] G.D. Magoulas, M.N. Vrahatis, G.S. Androulakis, A new method in neural network supervised training with imprecision, in: Proceedings of the IEEE 3rd International Conference on Electronics, Circuits and Systems, IEEE Press, Piscataway, NJ, 1996, pp. 287–290.
[10] G.D. Magoulas, M.N. Vrahatis, G.S. Androulakis, Improving the convergence of the back-propagation algorithm using learning rate adaptation methods, Neural Comput. 11 (1999) 1769–1796.
[11] B.J. Moré, B.S. Garbow, K.E. Hillstrom, Testing unconstrained optimization, ACM Trans. Math. Software 7 (1981) 17–41.
[12] J.M. Ortega, W.C. Rheinboldt, Iterative Solution of Nonlinear Equations in Several Variables, Academic Press, New York, 1970.
[13] V.P. Plagianakos, N.K. Nousis, M.N. Vrahatis, Locating and computing in parallel all the simple roots of special functions using PVM, J. Comput. Appl. Math. 133 (2001) 545–554.
[14] E. Polak, Optimization: Algorithms and Consistent Approximations, Springer, New York, 1997.
[15] M.J.D. Powell, An efficient method for finding the minimum of a function of several variables without calculating derivatives, Computer J. 7 (1964) 155–162.
[16] A. Ralston, P. Rabinowitz, A First Course in Numerical Analysis, McGraw-Hill, New York, 1978.
[17] W.C. Reinboldt, Methods for Solving Systems of Nonlinear Equations, SIAM, Philadelphia, PA, 1974.
[18] 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, Cambridge, MA, 1986, pp. 318–362.
[19] K. Sikorski, Bisection is optimal, Numer. Math. 40 (1982) 111–117.
[20] A. Sperduti, A. Starita, Speed up learning and network optimization with extended back-propagation, Neural Networks 6 (1993) 365–383.
[21] G.W. Stewart, Introduction to Matrix Computations, Academic Press, New York, 1973.
[22] R. Varga, Matrix Iterative Analysis, 2nd Edition, Springer, Berlin, 2000.
[23] P.P. Van der Smagt, Minimization methods for training feedforward neural networks, Neural Networks 7 (1994) 1–11.
[24] M.N. Vrahatis, Solving systems of nonlinear equations using the nonzero value of the topological degree, ACM Trans. Math. Software 14 (1988) 312–329.
[25] M.N. Vrahatis, CHABIS: A mathematical software package for locating and evaluating roots of systems of nonlinear equations, ACM Trans. Math. Software 14 (1988) 330–336.
[26] M.N. Vrahatis, A short proof and a generalization of Miranda's existence theorem, Proc. Amer. Math. Soc. 107 (1989) 701–703.
[27] M.N. Vrahatis, G.S. Androulakis, J.N. Lambrinos, G.D. Magoulas, A class of gradient unconstrained minimization algorithms with adaptive stepsize, J. Comput. Appl. Math. 114 (2000) 367–386.
[28] M.N. Vrahatis, G.S. Androulakis, G.E. Manousakis, A new unconstrained optimization method for imprecise function and gradient values, J. Math. Anal. Appl. 197 (1996) 586–607.
[29] M.N. Vrahatis, T.N. Grapsa, O. Ragos, F.A. Zafiropoulos, On the localization and computation of zeros of Bessel functions, Z. Angew. Math. Mech. 77 (1997) 467–475.
[30] M.N. Vrahatis, K.I. Iordanidis, A rapid generalized method of bisection for solving systems of nonlinear equations, Numer. Math. 49 (1986) 123–138.

[31] M.N. Vrahatis, O. Ragos, F.A. Zafiropoulos, T.N. Grapsa, Locating and computing zeros of Airy functions, Z. Angew. Math. Mech. 76 (1996) 419–422.
[32] T.P. Vogl, J.K. Mangis, A.K. Rigler, W.T. Zink, D.L. Alkon, Accelerating the convergence of the back-propagation method, Biol. Cybern. 59 (1988) 257–263.
[33] R.G. Voigt, Rates of convergence for a class of iterative procedures, SIAM J. Numer. Anal. 8 (1971) 127–134.
[34] D. Young, Iterative methods for solving partial difference equations of elliptic type, Trans. Amer. Math. Soc. 76 (1954) 92–111.
[35] P. Wolfe, Convergence conditions for ascent methods, SIAM Rev. 11 (1969) 226–235.
[36] P. Wolfe, Convergence conditions for ascent methods. II: Some corrections, SIAM Rev. 13 (1971) 185–188.
[37] W.I. Zangwill, Minimizing a function without calculating derivatives, Computer J. 10 (1967) 293–296.
[38] G. Zoutendijk, Nonlinear programming, computational methods, in: J. Abadie (Ed.), Integer and Nonlinear Programming, North-Holland, Amsterdam, 1970, pp. 37–86.