

# Optimal Algorithms for Well-Conditioned Nonlinear Systems of Equations

Monica Bianchini, Stefano Fanelli, and Marco Gori, *Fellow, IEEE*

**Abstract**—We propose solving nonlinear systems of equations by function optimization and we give an optimal algorithm which relies on a special canonical form of gradient descent. The algorithm can be applied under certain assumptions on the function to be optimized, that is, an upper bound must exist for the norm of the Hessian, whereas the norm of the gradient must be lower bounded. Due to its intrinsic structure, the algorithm looks particularly appealing for a parallel implementation. As a particular case, more specific results are given for linear systems. We prove that reaching a solution with a degree of precision  $\varepsilon$  takes  $\Theta(n^2 k^2 \log \frac{1}{\varepsilon})$ ,  $k$  being the condition number of  $A$  and  $n$  the problem dimension. Related results hold for systems of quadratic equations for which an estimation for the requested bounds can be devised. Finally, we report numerical results in order to establish the actual computational burden of the proposed method and to assess its performances with respect to classical algorithms for solving linear and quadratic equations.

**Index Terms**—Computational complexity, nonlinear systems of equations, parallel processing, terminal attraction dynamics.

## 1 INTRODUCTION

LINEAR and nonlinear systems of equations are the basis of many models in science and engineering and their efficient numerical solution is critical to progress in these areas. In particular, the efficient and robust solution of a system of nonlinear equations can be a rather challenging problem—especially in cases where only little a priori information on the solution is available.

A well-known method for solving nonlinear equations is the Newton method, an iterative scheme which is known to converge quadratically, but only if the initial guess is sufficiently close to the solution. Now, in order to extend the convergence domain of Newton's method, some globalizations are in common use, e.g., damped Newton methods, Levenberg-Marquardt, and steepest descent methods [1], [2], [3], [4], [5]. Based on the latter techniques, some packages have been developed, e.g., the codes from IMSL, NAG, and MINPACK [6].

There are also algorithms for finding zeros or fixed points of nonlinear systems of equations that are globally convergent for almost all starting points, i.e., with probability one. The essence of all such algorithms is the construction of an appropriate homotopy map and then the tracking of some smooth curve in the zero set of this homotopy map—HOMPACK [7], for instance, provides three qualitatively different algorithms for tracking the homotopy zero curve: ordinary differential equation-based, normal flow, and augmented Jacobian matrix. A similar approach was introduced in [8] where the solutions of a system of nonlinear algebraic equations are determined as asymptotic values of trajectories of systems of ordinary

differential equations. More precisely, the numerical integration of the classical Cauchy problem associated to the given system by means of Euler's equations allows the construction of very efficient algorithms. A-stable methods for the Cauchy problem lead, in fact, to the implementation of quadratic or superlinear algorithms for the original algebraic system. The idea of connecting the zeros of the systems of algebraic equations to the solutions of differential equations was used also in [9]. Another classical approach is based on the extension to nonlinear systems of the algorithms of the so-called ABS class [10], [11]. The close relationship between the ABS class and the well-known Brent-Brown methods [12], [13], as well as the excellent computational properties of both classes, lead to the construction of an efficient technique based on the ideas given in [14] (Section 7.4). More recently, a nonlinear version of the Generalized Conjugate Gradient (NGCG) and the corresponding global convergence results were introduced in [15] under suitable assumptions. It is worth mentioning that this method can be efficiently implemented in conjunction with nonlinear preconditioning. Moreover, by combining an approximate version of Newton's method and NGCG, global convergence can be guaranteed under rather general conditions. Finally, Newton type methods are also used for the approximate solution of nonlinear ill-posed operator equations.

Classical iterative methods for linear systems, such as Jacobi iteration, can be accelerated considerably by Krylov subspace<sup>1</sup> methods like GMRES [16]. Inexact Newton methods for nonlinear problems can be accelerated in a similar way [17], leading to a general framework that includes many well-known techniques for solving linear and nonlinear systems. Inexact Newton methods are frequently used in practice to avoid the expensive exact solution of the large linear system arising in the (possibly inexact) linearization step of Newton's process. Moreover,

• M. Bianchini and M. Gori are with the Dipartimento di Ingegneria dell'Informazione, Università di Siena, Via Roma, 56, 53100 Siena, Italy. E-mail: marco@ing.unisi.it.

• S. Fanelli is with the Dipartimento di Matematica, Università di Roma "Tor Vergata," Viale della Ricerca Scientifica, 1, 00133 Rome, Italy.

Manuscript received 6 Nov. 1998; revised 9 Oct. 2000; accepted 9 Mar. 2001. For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 108192.

1. The Krylov subspace is the linear space  $\text{span}[\mathbf{r}_0, A\mathbf{r}_0, A^2\mathbf{r}_0, \dots, A^i\mathbf{r}_0]$ , with  $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$  and  $\mathbf{x}_0$  initial estimation for the linear system solution.

the solution of the nonlinear system can be approached by solving a stream of linear systems, with slowly varying righthand side and coefficient matrix. In [18], a method is described which is also well-suited for parallel implementation.

A typical way of solving nonlinear systems is that of framing the problem in an optimization scheme. Basically, a cost function is defined which collects the residual-errors. This approach is quite general and can also be applied to the resolution of linear systems (see, e.g., [19], [20]). Nevertheless, apart from the convex case, the fundamental drawback of descending the cost surface by gradient-based algorithms is their susceptibility to local minima. Recently, some authors have independently introduced new optimization algorithms in the area of neural networks that are based on the properties of terminal attractors and repellers [21], [22], [23]. Although there is no theoretical assurance that the global solution can be reached via this kind of algorithms, apart from the convex case or when starting in the domain of attraction of the global minimum, nevertheless the terminal attraction dynamics allows us to reach the (eventually local) solution in finite time.<sup>2</sup> In this paper, we propose two algorithms, referred to as CGD (*Canonical Gradient Descent*)<sup>3</sup> and CGD-BP (*Canonical Gradient Descent-Boosting Precision*), respectively, which are based on terminal attraction. In the case of local minima free error functions or when starting the optimization in the basin of attraction of the desired minimum, we prove that the optimal solution is reached in a number of steps which is independent of the cost function. Most importantly, these algorithms allow us to determine the time required for finding the optimal solution. We prove that reaching the solution in the linear case takes  $\Theta(n^2 k^2 \log \frac{k}{\epsilon})$ , where  $n$  is the system dimension,  $k$  is the condition number, and  $\epsilon$  is the required accuracy. In practice, those problems for which the condition number is independent of the matrix dimension can be solved optimally with a given degree of precision. Related, but more significant, results can also be devised in the quadratic case just by directly extending the concept of system conditioning for nonlinear problems. Finally, it is worth mentioning that the proposed methodology is well-suited for parallel implementations since the proposed algorithms are based on gradient computation.

This paper is organized as follows: In the next section, we define a canonical form of gradient descent and give the notation used throughout the paper. In Section 3, we describe how the canonical gradient descent can be used for solving linear systems by quadratic optimization of the residual-error function. Section 4 is devoted to the more general problem of solving systems of quadratic equations. In Section 5, some results are given concerning linear system solving and quadratic optimization. In the quadratic case, for generalized Rosenbrock and tridiagonal Broyden

2. Moreover, terminal attractor algorithms may escape from the basins of attraction of local minima because of numerical errors which produce random jumps in the unknown space, thereby providing a restart in the trajectory to the solution. This instability behavior turns out to be a positive feature of the algorithms which makes them useful for practical applications.

3. The terminal attractor dynamics allows us to end up with a *canonical* differential equation which makes it possible to find the solution independently of the function.

functions [24], CGD-BP shows a noticeable increase of performance with respect to the Powell hybrid and Levenberg-Marquardt methods. Finally, in Section 6, we draw some conclusions. In order to improve readability, the proofs of the main theoretical results are collected in the Appendix.

## 2 CANONICAL GRADIENT DESCENT

Let us consider the nonlinear system of equations

$$\mathbf{F}(x) = \mathbf{0},$$

where  $x \in \Omega \subset \mathbb{R}^n$ . Of course, this equation can be solved by optimizing

$$E = \|\mathbf{F}(x)\|^2.$$

The optimization of function  $E$  can be carried out by means of the differential scheme

$$\frac{dx}{dt} = f(t, x) = -\frac{\eta}{\|\nabla_x E\|^2} \nabla_x E, \quad (1)$$

where  $x \in \Omega$  is the unknown vector. Based on this scheme, the dynamics of the error function becomes

$$\frac{dE}{dt} = (\nabla_x E)^T \frac{dx}{dt} = (\nabla_x E)^T \left( -\frac{\eta}{\|\nabla_x E\|^2} \nabla_x E \right) = -\eta, \quad (2)$$

where  $\eta$  is the rate of convergence. The function  $E$  decreases to zero by means of a *terminal attractor* dynamics, that is, in the finite time  $t_e$ , the transient beginning from  $E(0) = E_0$  reaches the equilibrium point  $E = 0$ , which is a *terminal attractor*. Finally, if we choose  $\eta = \frac{E_0}{\sigma}$ , then the solution is reached for  $t_e = \sigma$ , independently of the function at hand, being  $E(t) = E_0 - \frac{E_0}{\sigma} t$ .

This analysis holds for continuous computational models and one might wonder whether this nice property of forcing the dynamics still holds when using the discrete counterpart. The continuous canonical gradient descent model is indeed ideal, as no problem either due to accuracy or limited energy has been taken into account. This kind of problem may occur particularly in very flat and very abrupt zones of the error function. The use of a quantization step that is not small enough may generate errors due to undersampling, as well as problems of numerical representation. Nevertheless, there are a couple of reasons for choosing the special dynamics of (2). First, its dynamics is that of a terminal attractor and, in particular, it allows us to determine, in a very simple mathematical form, the time required for approaching the solution. Second, its simple form is also desirable in order to evaluate the errors arisen from the discretization process.

Let us now consider the following discrete version of (1), which represents Euler's approximation to the unknown dynamics:

$$x_{i+1} = x_i - \tau \eta \frac{\nabla E_i}{\|\nabla E_i\|^2}, \quad (3)$$

where the iteration index  $i$  is related to the continuous time  $t$  and to the quantization step  $\tau$  by  $t = i\tau$ .

**Definition 2.1.** *The Eulerian approximation (3) is consistent with the continuous equation (1) provided that  $\forall \varepsilon_a > 0, \exists \tau > 0$  such that  $\forall t = i\tau, |E(t) - E_i| \leq \varepsilon_a$ .*

The following theorem gives a suggestion on the choice of the quantization step  $\tau$  which guarantees the desired approximation  $\varepsilon_a$  and, consequently, allows us to estimate the number of steps  $i^*$  required for the optimization.

**Theorem 2.1.** *Let  $\tau \leq \frac{\varepsilon_a}{\eta}$ . Then,  $\forall \varepsilon_a \in \mathbb{R}^+$ , the Eulerian approximation is consistent with the continuous evolution of the error dynamics in the domain*

$$\mathcal{D}_{\varepsilon_a} \doteq \{\mathbf{x}_i \in \mathbb{R}^n : E_i > \varepsilon_a\},$$

when choosing quantization steps lower than or equal to

$$\tau^* = 2\sigma \frac{\varepsilon_s^2 \varepsilon_a}{HE_o^2}, \quad (4)$$

where  $\sigma = \frac{E_o}{\eta}$ ,  $\|\nabla E_i\| \geq \varepsilon_s$  in  $\mathcal{D}_{\varepsilon_a}$  (i.e.,  $\varepsilon_s$  is a lower bound for the norm of the gradient in  $\mathcal{D}_{\varepsilon_a}$ ) and the Hessian matrix  $\mathcal{H}(\mathbf{x})$  is such that,  $\forall \mathbf{x}, \|\mathcal{H}(\mathbf{x})\| \leq H$  (i.e.,  $H$  is an upper bound for the norm of the Hessian).

**Proof.** See the Appendix. □

**Corollary 2.1 (Number of steps for terminal solutions).** *If we adopt the terminal condition  $E_{i^*} \leq \varepsilon_a$ , then a residual error  $E(\mathbf{x}_{i^*})$  bounded by  $\varepsilon_e \doteq 2\varepsilon_a$  is reached after no more than*

$$i^* = \left\lceil \frac{HE_o^2}{\varepsilon_s^2 \varepsilon_e} \right\rceil \quad (5)$$

steps of (3).

**Proof.** See the Appendix. □

It can promptly be seen that the number of steps explodes when forcing the terminal condition to yield arbitrarily high precision ( $\varepsilon_e \rightarrow 0$ ).

**Definition 2.2.** *The degree of approximation carried out in (3) is defined by*

$$\rho \doteq \frac{\varepsilon_e}{E_o},$$

which is referred to as the approximation ratio.

**Remark 2.1.** Note that the number of steps of (5) depends on the required residual tolerance  $\varepsilon_e$ . In particular, it can be conveniently expressed by means of

$$i^* = \left\lceil \frac{HE_o}{\rho \varepsilon_s^2} \right\rceil.$$

The following algorithm can be used to determine the required solution.

**Algorithm 1:** CGD

**Input:**

Function  $\mathbf{F}(\mathbf{x})$ ;

Approximation ratio  $\rho$ .

**Output:**

A parameter vector  $\mathbf{x}_{i^*}$  such that  $E(\mathbf{x}_{i^*}) \leq \varepsilon_e$ .

**begin**

Choose a random initial point  $\mathbf{x}_o$ ;

$$\begin{aligned} E_o &\leftarrow E(\mathbf{x}_o); \\ H &\leftarrow \max_{\mathbf{x}} \|\mathcal{H}(\mathbf{x})\|; \end{aligned}$$

$$\varepsilon_s = \min_{\mathbf{x} \in \mathcal{D}_{\varepsilon_a}} \|\nabla E(\mathbf{x})\|;$$

$$i^* \leftarrow \left\lceil \frac{HE_o}{\rho \varepsilon_s^2} \right\rceil;$$

**for**  $i \leftarrow 1, \dots, i^*$  **do**

$$\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i - \frac{E_o}{i^*} \frac{\nabla E_i}{\|\nabla E_i\|^2};$$

**return**  $\mathbf{x}_{i^*}$ ;

**end**

**Remark 2.2.** The integration step  $\tau$  is independent of the special function  $E(\cdot)$  involved and is only related to  $H, E_o, \rho, \varepsilon_s$ .

### 3 THE SPECIAL CASE OF LINEAR SYSTEMS

The CGD algorithm proposed in the previous section can be nicely specialized to the case of linear systems in which  $\mathbf{F}(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b}$ . We consider the following error index:

$$\begin{cases} E(\mathbf{x}) \doteq \frac{1}{2} \lambda \frac{\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2}{\|\mathbf{b}\|^2}, & \|\mathbf{b}\| \neq 0, \lambda > 0, \\ E(\mathbf{x}_o) = \frac{1}{2} \lambda, & \text{for } \mathbf{x}_o = \mathbf{0}. \end{cases} \quad (6)$$

The following lemma allows us to determine a lower bound for the norm of the gradient when the discrete residual-error is greater than the threshold  $\varepsilon_a$ .

**Lemma 3.1** *Let us consider the error function (6). Then,  $\forall \rho > 0$ , we can choose*

$$\varepsilon_s \doteq \sqrt{\frac{\rho}{2}} \frac{\lambda}{\|\mathbf{A}^{-1}\| \|\mathbf{b}\|}$$

such that  $\|\nabla E_i\| \geq \varepsilon_s$  holds  $\forall \mathbf{x}_i \in \mathcal{D}_{\varepsilon_a}$ .

**Proof.** See the Appendix. □

We can now establish how the number of steps  $i^*$  given by Corollary 2.1 depends on the system parameters.

**Theorem 3.1.** *The number of steps  $i^*$  required by Algorithm 1 to reach the condition  $E(\mathbf{x}^*) \leq \varepsilon_e$  is given by*

$$i^* \leq \left\lceil \frac{k^2(\mathbf{A})}{\rho^2} \right\rceil,$$

where  $k(\mathbf{A}) \doteq \|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\|$  is the condition number of  $\mathbf{A}$ .

**Proof.** See the Appendix. □

So far, we have considered a stopping criterion involving solely the residual-error. However, the relative-error of the solution could be more informative, especially for ill-conditioned systems.

**Lemma 3.2.** *Let  $E(\mathbf{x})$  be defined as in (6) and consider the value  $\mathbf{x}_{i^*}$  such that  $E_{i^*} \leq \varepsilon_a$ . Then, the relative error of the solution is bounded by*

$$\varepsilon \doteq \frac{\|\mathbf{x}_{i^*} - \hat{\mathbf{x}}\|}{\|\hat{\mathbf{x}}\|} \leq \sqrt{\frac{\rho}{2}} k(\mathbf{A}), \quad (7)$$

where  $\hat{\mathbf{x}}$  is the exact solution.

**Proof.** See the Appendix.  $\square$

**Theorem 3.2.** *The number of steps  $i^*$  required by Algorithm 1 to reach relative accuracy  $\varepsilon$  is given by*

$$i^* \leq \left\lceil \frac{k^6(A)}{4\varepsilon^4} \right\rceil.$$

**Proof.** See the Appendix.  $\square$

Either Theorem 3.1 or Theorem 3.2 clearly indicate that Algorithm 1 allows a computational complexity of

$$\Theta\left(\frac{k^6(A)}{\varepsilon^4} n^2\right), \quad (8)$$

which is optimal when  $k(A)$  is independent of the matrix dimension.<sup>4</sup> Although theoretically interesting, the dependence on  $\varepsilon$  and  $k(A)$  given in (8) suggests that Algorithm 1 can hardly meet typical precision requirements of practical applications in reasonable time.

**Remark 3.1 (Sparse matrices).** Equation (8) gives the computational complexity of the CGD algorithm in the general case of dense matrices. If  $A$  is sparse, the method can be adapted in order to “fit” the matrix structure. Therefore, the computational burden due to gradient calculation remains proportional to the number of nonzero entries of  $A$ .

**Remark 3.2 (Parallel implementation).** Thanks to its intrinsic modular structure, the CGD algorithm can be easily rewritten following a parallel scheme. In fact, CGD can significantly benefit from a parallel computation of the gradient. In particular, when either the condition number  $k(A)$  or the accuracy  $\varepsilon$  are given, the complexity of a parallel implementation of CGD is the same as that of computing the gradient. Since  $\nabla E = \lambda \frac{A^T(Ax-b)}{\|b\|^2}$ , the parallel computation just consists of the parallelization of matrix-vector multiplications, which has been the subject of massive research (see, e.g., [25], pp. 160-168). The matrix-vector product heavily depends on the density of the matrix and, eventually, on its particular sparsity pattern. Some interesting results on the parallelization of iterative methods for large and sparse linear systems are found in [26], [27], [28], [29].

### 3.1 Boosting Precision

A careful analysis of Algorithm 1 shows that most of the computational resources are dissipated whenever we need a high degree of accuracy. One can, however, circumvent this problem by simply reducing the error to a certain fraction of the initial value (e.g., to its half), restoring the initial value, and then repeating the gradient descent for  $s$  steps until the desired degree of accuracy is reached.

First of all, let us calculate the number of steps required for reducing the initial value of the error function to its half.

**Corollary 3.1.** *If we adopt the terminal condition  $E(x_{i_{1/2}^*}) \leq E_o/2 - \varepsilon_a$ , then  $E_{i_{1/2}^*} \leq E_o/2$  after no more than*

4. In fact, in this case, the computational complexity of Algorithm 1 equals the cost of loading the data, i.e., the matrix  $b^T A$ .

$$i_{1/2}^* = i^* \left\lceil \frac{1+\rho}{2} \right\rceil$$

steps of (3).

**Proof.** See the Appendix.  $\square$

Note that, unlike Corollary 2.1, in the case of halved-solutions we need not choose a small value for  $\varepsilon_a$  since we are only interested in reaching an error less than  $E_o/2$ . For instance, we can choose  $\varepsilon_a = E_o/4$  ( $\rho = 1/2$ ) so that the number of steps which guarantees  $E(x_{i_{1/2}^*}) \leq E_o/2$  becomes

$$i_{1/2}^* \leq \frac{3HE_o}{2\varepsilon_s^2}.$$

In particular, in the case of linear equations, from Theorem 3.1, we derive that

$$i_{1/2}^* \leq 3k^2(A). \quad (9)$$

Note that  $i_{1/2}^*$  can be established once we know in advance that our equation has a bounded value of  $k(A)$ . Alternatively, when no such information is given,  $k(A)$  can optimally be estimated with an  $\mathcal{O}(n^2)$  computational burden.<sup>5</sup>

A formal statement of the proposed idea is given by the following algorithm for linear systems<sup>6</sup>:

#### Algorithm 2: CGD-BP (CGD-Boosting Precision)

**Input:**

Matrices  $A$  and  $b$ ;

Approximation ratio  $\rho$ .

**Output:**

A solution  $\tilde{x}$  such that  $E(\tilde{x}) = \frac{1}{2} \lambda \frac{\|A\tilde{x}-b\|^2}{\|b\|^2} \leq \varepsilon_e$ .

**begin**

$x_o \leftarrow 0$ ; {set the starting guess}

$s \leftarrow 0$ ;

$E_o^s \leftarrow E(x_o)$ ;

Compute  $k(A)$ ;

$i_{1/2}^* \leftarrow 3k^2(A)$ ; {see (9)}

$E^s \leftarrow E_o^s$ ;

**while**  $\frac{E^s}{E_o^s} > \rho$  **do**

**begin**

**for**  $i \leftarrow 1, \dots, i_{1/2}^*$  **do**

$$x_{i+1} \leftarrow x_i - \frac{E_i^s}{i_{1/2}^*} \frac{\nabla E_i^s}{\|\nabla E_i^s\|};$$

$s \leftarrow s + 1$ ;

$$E^s \leftarrow \frac{1}{2} \lambda \frac{\|Ax_{i_{1/2}^*} - b\|^2}{\|b\|^2};$$

5. The problem of having an economical method for estimating the condition number of  $A$ , which is obviously analogous to that of approximating  $\|A^{-1}\|$ , has been widely debated. Since 1979, when the LINPACK package was released, the same authors have proposed an approximating algorithm involving  $\mathcal{O}(n^2)$  floating-point operations and assuring a reliable indication for the order of magnitude of  $k(A)$  [30]. After such a proposal, several researchers attempted to enhance the previous result [31], [32] until, in 1984, Hager suggested a new technique for estimating the  $l_1$  condition number with an error under  $5^o/_{oo}$  and a probability higher than 0.97 [33]. Finally, it is worth mentioning that, because of the very nature of the approximation method, all such algorithms calculate an underestimation of the actual condition number.

6. The algorithm can easily be extended to the general case of nonlinear equations.

$$E_o^s \leftarrow 4E_{i_{1/2}^{s-1}}; \quad \{\varepsilon_a = E_o/4\}$$

**end**  
**return**  $\tilde{x}$ ;  
**end**

Now, let us estimate the number of steps  $s$  required to attain the desired accuracy. The following theorem gives an answer to this problem:

**Theorem 3.3.** *The number of steps  $s$  required by Algorithm 2 to reach relative accuracy  $\varepsilon$  is bounded by*

$$s \leq 2 \left\lceil \log_2 \frac{k(\mathbf{A})}{\sqrt{2}\varepsilon} \right\rceil.$$

**Proof.** See the Appendix.  $\square$

From this theorem and (9), we can immediately see that Algorithm 2 provides a solution in

$$\Theta\left(k^2(\mathbf{A}) \log_2 \frac{k(\mathbf{A})}{\varepsilon} n^2\right) \quad (10)$$

steps.

#### 4 SYSTEMS OF QUADRATIC EQUATIONS

Let us consider the quadratic system

$$\mathbf{F}(\mathbf{x}) = \mathbf{0}, \text{ where } \begin{cases} F_j(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T \mathbf{A}_j \mathbf{x} + \mathbf{x}^T \mathbf{b}_j - c_j = 0, \\ \mathbf{A}_j \text{ symmetric,} \end{cases}$$

$$j = 1, \dots, n,$$

to which the following error index can be attached:

$$E(\mathbf{x}) = \frac{1}{2} \|\mathbf{F}(\mathbf{x})\|^2 = \frac{1}{2} \sum_{j=1}^n [F_j(\mathbf{x})]^2. \quad (11)$$

As for linear systems, we need to find an upper bound of the Hessian matrix and a lower bound of the gradient.

**Lemma 4.1.** *Let us consider the error function (11). Let*

$$\tilde{\mathbf{A}}(\mathbf{x}) = [\mathbf{a}_1(\mathbf{x}), \mathbf{a}_2(\mathbf{x}), \dots, \mathbf{a}_n(\mathbf{x})] = \mathbf{J}^T(\mathbf{x}) \in \mathbb{R}^{n,n},$$

with  $\mathbf{a}_j(\mathbf{x}) \doteq \mathbf{A}_j \mathbf{x} + \mathbf{b}_j$ . If  $[\tilde{\mathbf{A}}(\mathbf{x}_i)]^{-1}$  exists, then

$$\|\nabla E_i\| \geq \frac{\sqrt{2\varepsilon_a}}{\|[\tilde{\mathbf{A}}(\mathbf{x}_i)]^{-1}\|},$$

holds  $\forall \mathbf{x}_i \in \mathcal{D}_{\varepsilon_a}$ .

**Proof.** See the Appendix.  $\square$

**Remark 4.1.** The bound on the norm of the gradient depends on the point actually reached in the descending procedure. When a good estimation of the solution is available, the above inverse matrix can be roughly estimated using the starting guess  $\mathbf{x}_o$ :

$$\varepsilon_s \approx \frac{\sqrt{2\varepsilon_a}}{\|[\tilde{\mathbf{A}}(\mathbf{x}_o)]^{-1}\|}. \quad (12)$$

If  $\mathbf{x}_o$  is not a good approximation of the solution, we can use the value of  $\varepsilon_s$  for a limited number of steps and then recalculate  $[\tilde{\mathbf{A}}(\mathbf{x}_o)]^{-1}$  again.

An estimation for  $H$  can be obtained as follows:

$$\begin{aligned} \|\mathcal{H}(\mathbf{x}_i)\| &= \left\| \sum_{j=1}^n [\mathbf{F}'_j(\mathbf{x}_i) \mathbf{F}'_j(\mathbf{x}_i)]^T + \mathbf{F}_j(\mathbf{x}_i) \mathbf{F}''_j(\mathbf{x}_i) \right\| \\ &\leq \left\| \sum_{j=1}^n \mathbf{F}'_j(\mathbf{x}_i) [\mathbf{F}'_j(\mathbf{x}_i)]^T \right\| + \left\| \sum_{j=1}^n \mathbf{F}_j(\mathbf{x}_i) \mathbf{A}_j \right\| \\ &\leq \left\| \tilde{\mathbf{A}}(\mathbf{x}_i) [\tilde{\mathbf{A}}(\mathbf{x}_i)]^T \right\| + \|\mathbf{F}(\mathbf{x}_o)\| \max_{1 \leq j \leq n} \|\mathbf{A}_j\|. \end{aligned}$$

If  $\mathbf{x}_o$  is close to  $\hat{\mathbf{x}}$ ,

$$H \approx C \|\tilde{\mathbf{A}}(\mathbf{x}_o)\|^2, \quad (13)$$

with  $C = 1 + \frac{\|\mathbf{F}(\mathbf{x}_o)\|}{\|\tilde{\mathbf{A}}(\mathbf{x}_o)\|^2} \max_{1 \leq j \leq n} \|\mathbf{A}_j\|$ .

As for Theorem 3.1, we can now estimate the number of steps  $i^*$  for attaining the desired precision. In fact, based on (5),

$$\begin{aligned} i^* &\approx \left\lceil \frac{C \|\tilde{\mathbf{A}}(\mathbf{x}_o)\|^2}{\rho^2} \right\rceil \\ &= \left\lceil \frac{C \|\tilde{\mathbf{A}}(\mathbf{x}_o)\|^2 \|\tilde{\mathbf{A}}(\mathbf{x}_o)^{-1}\|^2}{\rho^2} \right\rceil \\ &= \left\lceil \frac{C k^2(\tilde{\mathbf{A}}(\mathbf{x}_o))}{\rho^2} \right\rceil, \end{aligned}$$

where  $k(\tilde{\mathbf{A}}(\mathbf{x}_o)) \doteq \|\tilde{\mathbf{A}}(\mathbf{x}_o)\| \cdot \|\tilde{\mathbf{A}}(\mathbf{x}_o)^{-1}\|$  is the condition number of  $\tilde{\mathbf{A}}(\mathbf{x}_o)$ , provided that (12)-(13) allow valid estimations for  $\varepsilon_s$  and  $H$ , respectively.

In particular, adopting the terminal condition,  $E(\mathbf{x}_{i_{1/2}^*}) \leq E_o/2 - \varepsilon_a$ , then  $E_{i_{1/2}^*} \leq E_o/2$  approximately after

$$i_{1/2}^* \approx \lceil 3C k^2(\tilde{\mathbf{A}}(\mathbf{x}_o)) \rceil$$

steps of (3).

Therefore, CGD-BP is expected to approach the solution in

$$\Theta\left(n^2 \sum_{s=1}^{\lceil \log_2 1/\rho \rceil} k^2(\tilde{\mathbf{A}}(\mathbf{x}_o^s))\right)$$

steps.

**Remark 4.2.** It is worth mentioning that, as for the linear case, CGD-BP behaves optimally in the sense that its computational complexity reaches the  $\Theta(n^2)$  lower bound.

## 5 EXPERIMENTAL RESULTS

### 5.1 Example 1: Linear Systems

In this section, we report some numerical results aimed at comparing CGD-BP with classical Gauss algorithm. The comparison with Gauss algorithm makes sense since CGD-BP can actually deal with random dense matrices.

TABLE 1  
Experimental Results for Symmetric Matrices with  $k_2(A) = 5$  and  $\rho = 0.0625$

$n$	CGD-BP/Gauss	$err_{res}$ (CGD-BP)	$err_{rel}$ (CGD-BP)	$t$ (CGD-BP)	$t$ (Gauss)
100	17.00	$2.11E - 3$	0.164	00'03.6''	00'00.5''
500	3.60	$2.10E - 3$	0.153	01'31.7''	01'06.4''
1000	1.81	$2.10E - 3$	0.146	06'22.9''	09'06.8''
1800	1.01	$2.10E - 3$	0.155	22'01.4''	57'25.1''

The total number of steps for CGD-BP is  $i_{tot}^* \leq 300$ .

TABLE 2  
Experimental Results for Symmetric Matrices with  $k_2(A) = \sqrt[3]{n}$  and  $\rho = 0.125$

$n$	$i_{tot}^*$	CGD-BP/Gauss	$err_{res}$ (CGD-BP)	$err_{rel}$ (CGD-BP)	$t$ (CGD-BP)	$t$ (Gauss)
100	90	5.22	$8.89E - 3$	0.277	00'01.1''	00'00.5''
500	204	2.45	$7.39E - 3$	0.280	01'03.6''	01'05.5''
1000	285	1.71	$8.11E - 3$	0.332	06'02.3''	08'59.3''
1500	348	1.39	$8.26E - 3$	0.350	17'24.0''	32'23.6''

For these experiments, we consider  $A$  matrices,

$$A = P^T D P, \quad P = I - 2 \frac{v v^T}{v^T v},$$

which are constructed by applying orthogonal *Householder reflections* to a diagonal matrix with fixed condition number. We choose matrix  $D$  in such a way that its minimum eigenvalue is  $\lambda_{min} = 1$  and its maximum eigenvalue is  $\lambda_{max} = k_2(A)$ . In so doing,  $\|A\|_2 = \lambda_{max}$  and  $\|A^{-1}\|_2 = \lambda_{min}$ . Hence,  $k_2(A)$ , the 2-norm condition number, has the prescribed value. All the entries  $b_j$  of the righthand side are randomly generated in the interval  $(0, 1)$ .

In Table 1, the experiments carried out on matrices with constant condition number are summarized. In the first column, the matrix dimension is reported, whereas, in the second one, the ratio between CGB-BP and Gauss significant operations is given. Multiplications, divisions, and comparisons between floating-point numbers are taken into account. The third and fourth columns contain the values of the final residual-error and of the relative-error gained by CGD-BP. Finally, the runtime measurements, obtained on a Spark 10 Sun Station,<sup>7</sup> for both the methods are shown in the last two columns.

Our numerical results clearly assess the quadratic computational cost of CGD-BP, which directly follows from counting the dominant operations. Moreover, the elapsed-time measurements show that the practical implementation on the computer yields even better scaling with respect to less significant operations, not considered in the CGD-BP/Gauss ratio.

Note that the total number of steps for the iterative procedure is

7. The measurements are obtained using the Unix command *time* which times the execution of a script.

$$i_{tot}^* \leq i_{1/2}^* \cdot s = 3k_2(A) \left[ \log_2 \frac{1}{\rho} \right].$$

When assuming  $k_2(A) = 5$ ,  $\rho = 0.0625$ , we find that  $i_{tot}^* \leq 300$ .

It is worth noting that the Gaussian elimination (in single precision) obviously guarantees a higher precision w.r.t. CGD-BP. Nevertheless, when the time factor is critical, especially in solving very large systems, CGD-BP yields a *good* solution in a shorter time w.r.t. Gaussian elimination, also being less sensitive to round-off errors. Finally, if we stress precision requirements with respect to the experiments reported in Table 1, choosing  $\rho = 0.015625$ , the CGD-BP method becomes more expensive. For instance, for  $n = 1,800$ , the ratio CGD-BP/Gauss = 1.51. On the other hand,  $t(\text{CGD-BP})/t(\text{Gauss}) \approx 0.5777$ , which indicates that CGD-BP is already practically more efficient.

Table 2 summarizes some results obtained for matrices in which the condition number grows according to  $k_2(A) = \sqrt[3]{n}$ . The meaning of the columns is the same as in Table 1, apart from column two, where the total number of iterations is reported.

When choosing  $\rho = 0.031250$ , for  $n = 1,500$ , we obtain CGD-BP/Gauss = 2.33, while

$$t(\text{CGD-B})/t(\text{Gauss}) \approx 0.88672.$$

The CGD-BP algorithm has been tested on two well-known benchmarks [24], namely the *extended Rosenbrock function* and the *Broyden tridiagonal function*. For the extended Rosenbrock function, in the first test, a starting guess is selected which closely approaches the solution. Then, for both types of functions, a starting guess which does not represent a "good" estimation of the solution is chosen so that the number of steps  $i_{1/2}^*$  is estimated. The

TABLE 3

Experimental Results for the Extended Rosenbrock Function

$n$	$i^*$	$\varepsilon$	$t(\text{CGD-BP})$	$t(\text{PH})$	$t(\text{LM})$
2	9639	$6.272E-3$	$0'00.4''$	$0'00.0''$	$0'00.0''$
10	9925	$6.272E-3$	$0'00.8''$	$0'00.0''$	$0'00.0''$
50	10565	$6.272E-3$	$0'03.1''$	$0'00.5''$	$0'00.6''$
100	11043	$6.271E-3$	$0'06.1''$	$0'03.7''$	$0'04.3''$
150	11411	$6.271E-3$	$0'09.8''$	$0'12.2''$	$0'14.3''$
200	11721	$6.271E-3$	$0'12.9''$	$0'28.7''$	$0'33.8''$

$\xi_1[2j-1] = 0.98$ ,  $\xi_1[2j] = 1.02$ ,  $j = 1, \dots, n/2$ ;  $\rho_1 = 0.5$  (case in which  $x_o$  is close to the solution).

computation of  $i_{1/2}^*$  requires estimating the condition number of the matrix  $\tilde{\mathbf{A}}(x)$ , which takes  $\mathcal{O}(n^2)$ . Nevertheless, as will be shown later on, in both cases  $k(\tilde{\mathbf{A}}(x))$  can be computed even more efficiently.

### 5.2 Example 2: Extended Rosenbrock Function

The benchmark we consider is:

$$\begin{cases} F_{2j-1}(x) = 10(x[2j] - x[2j-1])^2, \\ F_{2j}(x) = 1 - x[2j-1], \\ x_o = \xi_i, i = 1, 2, \\ \text{where } \begin{cases} \xi_1[2j-1] = 0.98, & \xi_1[2j] = 1.02, \\ \xi_2[2j-1] = -1.2, & \xi_2[2j] = 1.0, \end{cases} j = 1, \dots, n/2, \\ \mathbf{F}(x) = \mathbf{0} \text{ at } (1, \dots, 1). \end{cases}$$

In this case,  $\tilde{\mathbf{A}}(x)$  is a block-diagonal matrix of the form:

$$\tilde{\mathbf{A}}(x) = \begin{pmatrix} \tilde{\mathbf{A}}_1(x) & \dots & \dots \\ \dots & \tilde{\mathbf{A}}_2(x) & \dots \\ \dots & \dots & \dots \\ \dots & \dots & \tilde{\mathbf{A}}_{n/2}(x) \end{pmatrix},$$

$$\tilde{\mathbf{A}}_j(x) = \begin{pmatrix} -20x[2j-1] & -1 \\ 10 & 0 \end{pmatrix},$$

where  $k(\tilde{\mathbf{A}}(x))$  is constant in the matrix dimension.<sup>8</sup> Therefore, every time the condition number must be evaluated, this will be done for matrices of dimension  $n = 2$ .

The experiments for the extended Rosenbrock function were carried out on systems of dimension 2 (*Rosenbrock function*), 10, 50, 150, 100, and 200, imposing  $\rho_1 = 0.5$  (case in which  $x_o$  is close to the solution) and  $\rho_2 = 4.8828E-4$  (case in which  $x_o$  is far away from the solution). In Tables 3 and 4, the total number of iterations and the relative-error gained by CGD-BP procedure are listed, together with the elapsed-time for the Powell hybrid (PH), the Levenberg-Marquardt

<sup>8</sup>  $k(\tilde{\mathbf{A}}(x))$  is constant in the matrix dimension because of the choice of the starting guess and of the block structure of  $\tilde{\mathbf{A}}(x)$ .

TABLE 4

Experimental Results for the Extended Rosenbrock Function

$n$	$i_{tot}^*$	$\varepsilon$	$t(\text{CGD-BP})$	$t(\text{PH})$	$t(\text{LM})$
2	51965	$3.785E-3$	$0'01.7''$	$0'00.0''$	$0'00.0''$
10	54864	$3.782E-3$	$0'04.5''$	$0'00.0''$	$0'00.1''$
50	61325	$3.787E-3$	$0'18.0''$	$0'02.0''$	$0'05.4''$
100	66177	$3.785E-3$	$0'37.4''$	$0'13.1''$	$0'39.3''$
150	69892	$3.788E-3$	$0'57.5''$	$0'41.1''$	$2'08.6''$
200	73020	$3.792E-3$	$1'21.5''$	$1'34.2''$	$4'59.7''$

$\xi_2[2j-1] = -1.2$ ,  $\xi_2[2j] = 1.0$ ,  $j = 1, \dots, n/2$ ;  $\rho_2 = 4.8828E-4$  (case in which  $x_o$  is far away from the solution).

(LM), and the canonical gradient descent algorithms, using Fortran code<sup>9</sup> and obtained on a Spark 10 Sun Station.

Starting from  $\xi_2$ , the stopping criterion  $E(x) \leq E(x_o) \cdot 4.8828E-4$  is verified after 11 evaluations of the condition number  $k(\tilde{\mathbf{A}}(x))$ , with  $n = 2$ .

Notice that, in the nonlinear case, a relationship between  $\rho$  and  $\varepsilon$ , which is independent of  $x$ , can hardly be devised. Nevertheless, augmenting the precision requirements on the residual-error obviously guarantees a decrease of the relative-error.

**Remark 5.1.** Following elapsed-time measurements in Tables 3 and 4, as a matter of fact, the canonical gradient descent algorithm shows a linear computational cost. This is due to the fact that, because of the special block structure of matrix  $\tilde{\mathbf{A}}(x)$ , the gradient can be optimally calculated with  $\mathcal{O}(n)$  operations. In the general framework previously described, this means that only the entries of matrix  $\tilde{\mathbf{A}}(x)$  different from zero are taken into account. It can easily be shown that such entries are exactly  $\frac{3}{2}n$ .

### 5.3 Example 3: Broyden Tridiagonal Function

The benchmark we consider is:

$$\begin{cases} F_j(x) = (3 - 2x[j])x[j] - x[j-1] - 2x[j+1] + 1, \\ j = 1, \dots, n, \\ \text{where } x[0] = x[n+1] = 0, \\ x_o = (-1, \dots, -1). \end{cases}$$

$\tilde{\mathbf{A}}(x)$  is a tridiagonal matrix of the form:

$$\tilde{\mathbf{A}}(x) = \begin{pmatrix} 3 - 4x[1] & -1 & \dots & \dots & \dots \\ -2 & 3 - 4x[2] & -1 & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & -2 & 3 - 4x[n] \end{pmatrix},$$

<sup>9</sup> Subroutines *hybrj1* and *lmdr1* are extracted from MINPACK [1], [34] at the GAMS (Guide to Available Mathematical Software) web site <http://math.nist.gov/>. In particular, *hybrj1* finds a zero of a system of  $n$  nonlinear functions in  $n$  variables by a modification of the Powell hybrid method, whereas *lmdr1* calculates a minimum of the sum of the squares of  $m$  nonlinear functions in  $n$  variables by a modification of the Levenberg-Marquardt algorithm. In both cases, the user must provide a subroutine which calculates the functions and the Jacobian.

TABLE 5  
Experimental Results  
for the Broyden Tridiagonal Function;  $\rho = 1.529E - 5$

$n$	$i_{tot}^*$	$\varepsilon$	$t(\text{CGD-BP})$	$t(\text{PH})$	$t(\text{LM})$
3	217	$1.305E - 5$	0'04.8''	0'00.0''	0'00.0''
10	328	$9.914E - 6$	0'04.8''	0'00.0''	0'00.0''
50	369	$5.594E - 6$	0'17.7''	0'00.8''	0'02.6''
100	374	$5.265E - 6$	0'17.9''	0'04.6''	0'19.6''
150	378	$5.154E - 6$	0'18.1''	0'14.5''	1'04.8''
200	378	$5.097E - 6$	0'18.2''	0'34.0''	2'34.2''

having a condition number which increases very slowly with the dimension. We also conjecture that the condition number  $k(\tilde{\mathbf{A}}(\mathbf{x}))$  is asymptotically bounded for all  $\mathbf{x}$ . For instance, with the suggested starting guess,  $k(\tilde{\mathbf{A}}(\mathbf{x}_o)) = 1.8648, 2.3934, 2.4948, 2.4987, 2.4994, 2.4997, 2.4998, 2.4999, 2.4999$ , respectively, for  $n = 3, 10, 50, 100, 150, 200, 300, 400, 500$ . Therefore, even in this case, using an estimation for the condition number on a "small"  $\tilde{\mathbf{A}}(\mathbf{x})$  (e.g.,  $n = 50$ ) does not produce significant errors.

The experiments for the Broyden tridiagonal function were carried out on systems of dimension 3 (*Broyden function*), 10, 50, 100, 150, and 200, respectively, by imposing  $\rho = 1.5259E - 5$ . The stopping criterion was verified after 16 evaluations of the condition number  $k(\tilde{\mathbf{A}}(\mathbf{x}))$ , with  $n = 50$ . In Table 5, the total number of iterations and the relative-error gained by the CGD-BP procedure are listed, together with the elapsed-time for the Powell hybrid, the Levenberg-Marquardt, and the canonical gradient descent algorithms.

**Remark 5.2.** For the Broyden tridiagonal function, evaluating the condition number is just the most expensive calculation; about 17'' of elapsed-time for  $n \geq 50$ . Also, for  $n = 3, 10$ , the measured elapsed-time (4.8'') is exclusively due to the condition number estimation.

## 6 CONCLUSIONS

This paper presents a new approach for solving nonlinear systems of equations by function optimization which is based on terminal attractors. The algorithm can be applied under certain assumptions on the function to be optimized, that is, an upper bound must exist for the norm of the Hessian, whereas the norm of the gradient must be lower bounded.

As a particular case, more specific results are given for linear systems. We prove that reaching a solution with degree of precision  $\varepsilon$  takes  $\Theta(n^2 k^2 \log \frac{k}{\varepsilon})$ ,  $k$  being the condition number of  $A$  and  $n$  the problem dimension. Since computing  $k$  takes  $O(n^2)$ , one can actually exhibit an optimal algorithm which returns the solution with any degree of precision  $\varepsilon$  fixed in advance. The quadratic case is also treated in detail. We give results which are very similar

to the case of linear systems, but, unfortunately, we can only provide an estimation of the required bounds on the Hessian matrix and the gradient. The theoretical results given in the paper are supported by numerical experiments which definitely confirm the optimal scaling-up with the system dimension.

Moreover, it is also pointed out that the proposed approach is very well-suited to parallel implementations. Finally, it is worth mentioning that the given methodology is likely to provide optimal algorithms in a number of different domains whenever a given problem is naturally put in the framework of function optimization. Possible candidates are linear and quadratic programming and computational geometry problems [35].

## APPENDIX

### Proof of Theorem 2.1

When updating the parameters from  $\mathbf{x}_i$  to  $\mathbf{x}_{i+1}$  according to (3), the error function changes from  $E_i$  to  $E_{i+1}$  and the variation can be calculated by using Taylor's expansion in  $\mathbf{x}_i$ . Hence,

$$E_{i+1} = E_i + (\nabla E_i)^T (\mathbf{x}_{i+1} - \mathbf{x}_i) + \frac{1}{2} (\mathbf{x}_{i+1} - \mathbf{x}_i)^T \mathcal{H}(\xi_i) (\mathbf{x}_{i+1} - \mathbf{x}_i),$$

with  $\xi_i \in ]\mathbf{x}_i, \mathbf{x}_{i+1}[$ . When updating  $\mathbf{x}_i$  according to (3), we get

$$E_{i+1} = E_i - \tau\eta + \frac{1}{2} (\mathbf{x}_{i+1} - \mathbf{x}_i)^T \mathcal{H}(\xi_i) (\mathbf{x}_{i+1} - \mathbf{x}_i). \quad (14)$$

Since  $E(t) = E_o - \eta t$ , the chosen approximation gives rise to an error in (14) that involves the second-order term only.<sup>10</sup> Once  $\mathbf{x}(t)$  is restricted to  $\mathcal{D}_{\varepsilon_a}$ , the correspondent function  $f(t, \mathbf{x})$  in (1) has a bounded partial derivative with respect to its second argument  $\mathbf{x}$ , whereas  $\mathbf{x}(t)$  has a bounded second derivative. Under this condition, the Eulerian approximation (3) of (1) converges to the exact solution as  $\tau \rightarrow 0$  (see [36], p. 26). therefore, the algorithm is stopped after a number of steps  $i = i^*$  such that  $E_{i^*-1} > \varepsilon_a$  and  $E_{i^*} \leq \varepsilon_a$  and, by the terminal attraction hypothesis,  $i^* \leq \frac{t_e}{\tau}$ . Hence,

$$\begin{aligned} E_{i^*} &= E_o - i^* \tau \eta \\ &+ \frac{1}{2} (\mathbf{x}_{i^*} - \mathbf{x}_{i^*-1})^T \mathcal{H}(\xi_{i^*-1}) (\mathbf{x}_{i^*} - \mathbf{x}_{i^*-1}) + \dots \\ &+ \frac{1}{2} (\mathbf{x}_1 - \mathbf{x}_o)^T \mathcal{H}(\xi_o) (\mathbf{x}_1 - \mathbf{x}_o) \\ &\leq E_o - i^* \tau \eta + \frac{1}{2} i^* \max_{0 \leq i \leq i^*-1} \left| (\mathbf{x}_{i+1} - \mathbf{x}_i)^T \mathcal{H}(\xi_i) (\mathbf{x}_{i+1} - \mathbf{x}_i) \right|. \end{aligned}$$

The maximum error is bounded in  $\mathcal{D}_{\varepsilon_a}$  provided that

$$\frac{1}{2} \frac{t_e}{\tau} \max_{0 \leq i \leq i^*-1} \left| (\mathbf{x}_{i+1} - \mathbf{x}_i)^T \mathcal{H}(\xi_i) (\mathbf{x}_{i+1} - \mathbf{x}_i) \right| \leq \varepsilon_a.$$

Since, during the gradient descent,  $\|\nabla E_i\| \geq \varepsilon_s$ , we have

10. Note that this property holds because of the special choice of the error dynamics. In general, a discretization error is introduced also concerning the first-order term.

$$\begin{aligned}
 & \frac{1}{2} \frac{t_e}{\tau} \max_{0 \leq i \leq i^* - 1} \left| (\mathbf{x}_{i+1} - \mathbf{x}_i)^T \mathcal{H}(\xi_i) (\mathbf{x}_{i+1} - \mathbf{x}_i) \right| \\
 & \leq \frac{1}{2} \frac{t_e}{\tau} \max_{0 \leq i \leq i^* - 1} \left[ \|\mathcal{H}(\xi_i)\| \|\mathbf{x}_{i+1} - \mathbf{x}_i\|^2 \right] \\
 & \leq \frac{1}{2} \frac{t_e}{\tau} H \left( \frac{\eta\tau}{\varepsilon_s} \right)^2 \leq \varepsilon_a.
 \end{aligned}$$

Finally, we can easily see that, since  $t_e = \frac{E_0}{\eta}$ , the quantization step  $\tau^*$  guarantees the required approximation according to (4).  $\square$

### Proof of Corollary 2.1

If we adopt the stopping criterion  $E_{i^*} \leq \varepsilon_a$ , then  $E(\mathbf{x}_{i^*}) \leq \varepsilon_e \doteq 2\varepsilon_a$  and this holds after no more than  $\lceil \sigma/\tau^* \rceil$  steps, which gives rise to the value  $i^*$  of (5).  $\square$

### Proof of Lemma 3.1

Let us consider  $\varepsilon_a > 0$  and let  $E_i \geq \varepsilon_a$ . If we calculate the gradient  $\nabla E_i$  and consider its 2-norm,<sup>11</sup> the thesis follows from

$$\begin{aligned}
 \|\nabla E_i\| &= \lambda \frac{\|A^T(\mathbf{Ax}_i - \mathbf{b})\|}{\|\mathbf{b}\|^2} = \lambda \frac{\|(A^T)^{-1}\| \|A^T(\mathbf{Ax}_i - \mathbf{b})\|}{\|A^{-1}\| \|\mathbf{b}\|^2} \\
 &\geq \frac{\sqrt{2\lambda\varepsilon_a}}{\|A^{-1}\| \cdot \|\mathbf{b}\|} = \sqrt{\frac{\rho}{2}} \frac{\lambda}{\|A^{-1}\| \|\mathbf{b}\|}.
 \end{aligned}$$

$\square$

### Proof of Theorem 3.1

From Corollary 2.1 and, particularly, from (5), we derive that  $i^*$  only depends on  $H$  and  $\varepsilon_s$ . The Hessian can be computed precisely, whereas, for the gradient, we can use the bound of Lemma 3.1. Hence:

$$i^* \leq \left\lceil \frac{\frac{1}{2} \lambda^2 \frac{\|A^T A\|}{\|\mathbf{b}\|^2}}{\frac{1}{2} \frac{\rho^2 \lambda^2}{\|A^{-1}\|^2 \|\mathbf{b}\|^2}} \right\rceil \leq \left\lceil \frac{\|A\|^2 \|A^{-1}\|^2}{\rho^2} \right\rceil = \left\lceil \frac{k^2(A)}{\rho^2} \right\rceil.$$

$\square$

### Proof of Lemma 3.2

Algorithm 1 guarantees that  $E_{i^*} \leq \varepsilon_a$  upon return and, consequently, the following inequalities hold

$$\sqrt{\frac{2\varepsilon_a}{\lambda}} \geq \frac{\|A\mathbf{x}_{i^*} - \mathbf{b}\|}{\|\mathbf{b}\|} = \frac{\|A\mathbf{x}_{i^*} - A\hat{\mathbf{x}}\|}{\|\mathbf{b}\|} \geq \frac{\|\mathbf{x}_{i^*} - \hat{\mathbf{x}}\|}{\|A^{-1}\| \|\mathbf{b}\|},$$

which implies

$$\begin{aligned}
 \frac{\|\mathbf{x}_{i^*} - \hat{\mathbf{x}}\|}{\|\hat{\mathbf{x}}\|} &\leq \sqrt{\frac{2\varepsilon_a}{\lambda}} \frac{\|A^{-1}\| \|\mathbf{b}\|}{\|\hat{\mathbf{x}}\|} = \sqrt{\frac{2\varepsilon_a}{\lambda}} \frac{\|A^{-1}\| \|A\hat{\mathbf{x}}\|}{\|\hat{\mathbf{x}}\|} \\
 &\leq \sqrt{\frac{2\varepsilon_a}{\lambda}} \frac{\|A^{-1}\| \|A\| \|\hat{\mathbf{x}}\|}{\|\hat{\mathbf{x}}\|} = \sqrt{\frac{2\varepsilon_a}{\lambda}} k(A),
 \end{aligned}$$

and, finally,

$$\varepsilon \leq \sqrt{\frac{2\varepsilon_a}{\lambda}} k(A) = \sqrt{\frac{\rho}{2}} k(A).$$

11. Using the 2-norm,  $\|A\| = \|A^T\|$ .

### Proof of Theorem 3.2

The thesis follows straightforwardly from Theorem 3.1 and Lemma 3.2.  $\square$

### Proof of Corollary 3.1

The time required by continuous gradient descent to go below  $E_0/2 - \varepsilon_a$  is

$$t_{1/2} = \sigma \frac{E_0/2 + \varepsilon_a}{E_0}.$$

Hence, the number of steps required to meet the desired stopping criterion is bounded by:

$$i_{1/2}^* \leq \frac{\sigma}{\tau^*} \frac{1 + \rho}{2} = i^* \frac{1 + \rho}{2},$$

which yields the thesis.  $\square$

### Proof of Theorem 3.3

In order to reduce the error from  $E_0$  to  $\rho E_0$ , the number of steps  $s$  required by Algorithm 2 is bounded by

$$s = \left\lceil \log_2 \frac{1}{\rho} \right\rceil.$$

Therefore, from (7),

$$\rho \geq \frac{2\varepsilon^2}{k^2(A)},$$

which, in turn, yields the thesis.  $\square$

### Proof of Lemma 4.1

Let us consider  $\varepsilon_a > 0$  and let  $E_i \geq \varepsilon_a$ . Then,

$$\begin{aligned}
 \|\nabla E_i\| &= \left\| \sum_{j=1}^n F_j(\mathbf{x}_i) \mathbf{F}'_j(\mathbf{x}_i) \right\| = \|\tilde{\mathbf{A}}(\mathbf{x}_i) \mathbf{F}(\mathbf{x}_i)\| \\
 &\geq \frac{\sqrt{2\varepsilon_a}}{\|\tilde{\mathbf{A}}(\mathbf{x}_i)^{-1}\|},
 \end{aligned}$$

being  $\mathbf{F}'_j(\mathbf{x}_i) = \mathbf{a}_j(\mathbf{x}_i)$ .  $\square$

### ACKNOWLEDGMENTS

The authors gratefully thank Dr. Franco Scarselli (University of Siena) for careful reading of the paper and very useful suggestions. This paper is dedicated to the memory of Marco Protasi, Professor of Computer Science at the University Tor Vergata (Rome), who contributed strongly to activate this research.

### REFERENCES

- [1] J.J. Moré and M.Y. Cosnard, "Numerical Solution of Nonlinear Equations," *ACM Trans. Math. Software*, vol. 5, no. 1, pp. 64-85, 1979.
- [2] J.E. Dennis and R.B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. SIAM, 1996.
- [3] A. Eiger, K. Sikorski, and F. Stenger, "A Bisection Method for Systems of Nonlinear Equations," *ACM Trans. Math. Software*, vol. 10, no. 4, pp. 367-377, 1984.
- [4] A.A.M. Cuyt, "Computational Implementation of the Multivariate Halley Method for Solving Nonlinear Systems of Equations," *ACM Trans. Math. Software*, vol. 11, no. 1, pp. 20-36, 1985.

$\square$

- [5] P. Deuffhard, *Newton Techniques for Highly Nonlinear Problems—Theory and Algorithms*. Academic Press, 1996.
- [6] J.J. Moré, B.S. Garbow, and K.E. Hillstom, "User's Guide for MINPACK-1," Technical Report ANL-80-74, Argonne Nat'l Laboratory, Argonne, Ill., 1980.
- [7] L.T. Watson, S.C. Billups, and A.P. Morgan, "ALGORITHM 652: HOMPAC: A Suite of Codes for Globally Convergent Homotopy Algorithms," *ACM Trans. Math. Software*, vol. 13, no. 3, pp. 281-310, 1987.
- [8] F. Zirilli, "The Solutions of Nonlinear Systems of Equations by Second Order Systems of Ordinary Differential Equations and Linearly Implicit A-Stable Techniques," *SIAM J. Numerical Analysis*, vol. 19, no. 4, pp. 800-815, 1982.
- [9] H. Jarausch and W. Mackens, "Solving Large Nonlinear Systems of Equations by an Adaptive Condensation Process," *Numerical Math.*, vol. 50, no. 6, pp. 633-653, 1987.
- [10] J. Abaffy, C.G. Broyden, and E. Spedicato, "A Class of Direct Methods for Linear Systems," *Numerical Math.*, vol. 45, no. 3, pp. 361-376, 1984.
- [11] J. Abaffy, A. Galántai, and E. Spedicato, "The Local Convergence of ABS Methods for Nonlinear Algebraic Equations," *Numerical Math.*, vol. 51, no. 4, pp. 429-439, 1987.
- [12] R.P. Brent, "Some Efficient Algorithms for Solving Systems of Nonlinear Equations," *SIAM J. Numerical Analysis*, vol. 10, no. 2, pp. 327-344, 1973.
- [13] K.M. Brown, "A Quadratically Convergent Newton-Like Method Based upon Gaussian-Elimination," *SIAM J. Numerical Analysis*, vol. 6, no. 4, pp. 560-569, 1969.
- [14] J.M. Ortega, W.C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*. New York: Academic Press, 1970.
- [15] O. Axelsson and A.T. Chronopoulos, "On Nonlinear Generalized Conjugate Gradient Methods," *Numerical Math.*, vol. 69, no. 1, pp. 1-15, 1994.
- [16] Y. Saad and M. H. Schultz, "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems," *SIAM J. Scientific and Statistical Computing*, vol. 7, pp. 856-869, 1986.
- [17] D.R. Fokkema, G.L.G. Sleijpen, and H.A. Van der Vorst, "Accelerated Inexact Newton Schemes for Large Systems of Nonlinear Equations," *SIAM J. Scientific Computing*, vol. 19, no. 2, pp. 657-674, 1998.
- [18] S.Y. Berkovich, "An Overlaying Technique for Solving Linear Equations in Real-Time Computing," *IEEE Trans. Computers*, vol. 42, no. 5, pp. 513-517, 1993.
- [19] R.W. Freund and N.M. Nachtigal, "QMR: A Quasi-Minimal Residual Method for Non-Hermitian Linear Systems," *Numerical Math.*, vol. 60, pp. 315-339, 1991.
- [20] R.W. Freund, "A Transpose-Free Quasi-Minimal Residual Algorithm for Non-Hermitian Linear Systems," *SIAM J. Scientific Computing*, vol. 14, no. 2, pp. 470-482, 1993.
- [21] M. Zak, "Terminal Attractors for Addressable Memory in Neural Networks," *Physics Letters A*, vol. 133, no. 1.2, pp. 18-22, 1988.
- [22] M. Zak, "Terminal Attractors in Neural Networks," *Neural Networks*, vol. 2, no. 4, pp. 259-274, 1989.
- [23] S. Wang and C.H. Hsu, "Terminal Attractor Learning Algorithms for Backpropagation Neural Networks," *Proc. Int'l Joint Conf. Neural Networks*, pp. 183-189, Nov. 1991.
- [24] J.J. Moré, B.S. Garbow, and K.E. Hillstom, "Testing Unconstrained Optimization Software," *ACM Trans. Math. Software*, vol. 7, no. 1, pp. 17-41, 1981.
- [25] V. Kumar, A. Grama, A. Gupta, and G. Karypis, *Introduction to Parallel Computing*. Benjamin Cummings, 1994.
- [26] C. Aykanat, F. Ozguner, F. Ercal, and P. Sadayappan, "Iterative Algorithms for Solution of Large Sparse Systems of Linear Equations on Hypercubes," *IEEE Trans. Computers*, vol. 37, no. 12, pp. 1554-1567, Dec. 1988.
- [27] D.H. Lawrie and A.H. Sameh, "The Computation and Communication Complexity of a Parallel Banded System Solver," *ACM Trans. Math. Software*, vol. 10, pp. 185-195, 1984.
- [28] E. Dekker and L. Dekker, "Parallel Minimal Norm Method for Tridiagonal Linear Systems," *IEEE Trans. Computers*, vol. 44, no. 7, pp. 942-946, July 1995.
- [29] A.R. Chowdhury, N. Bellas, and P. Banerjee, "Algorithm-Based Error-Detection Schemes for Iterative Solution of Partial Differential Equations," *IEEE Trans. Computers*, vol. 45, no. 4, pp. 394-407, Apr. 1996.

- [30] A.K. Cline, C.B. Moler, G.W. Stewart, and J.H. Wilkinson, "An Estimate for the Condition Number of a Matrix," *SIAM J. Numerical Analysis*, vol. 16, no. 2, pp. 368-375, 1979.
- [31] D.P. O'Leary, "Estimating Matrix Condition Numbers," *SIAM J. Scientific and Statistical Computing*, vol. 1, pp. 205-209, 1980.
- [32] A.K. Cline and R.K. Rew, "A Set of Counter-Examples to Three Condition Number Estimators," *SIAM J. Scientific and Statistical Computing*, vol. 4, no. 4, pp. 602-611, 1983.
- [33] W.W. Hager, "Condition Estimates," *SIAM J. Scientific and Statistical Computing*, vol. 5, no. 2, pp. 311-316, 1984.
- [34] "MINPACK Documentation," Applied Math. Division, Argonne Nat'l Laboratory, Argonne Ill., <http://www.nctlib.org/minpack/readme>, 2001.
- [35] F. Preparata and M.I. Shamos, *Computational Geometry: An Introduction*. Springer-Verlag, 1985.
- [36] G.H. Golub and J.M. Ortega, *Scientific Computing and Differential Equations: An Introduction to Numerical Methods*. Academic Press, 1992.



neural networks, optimization, artificial intelligence, and numerical methods for linear systems and ODEs.



optimal learning, numerical approaches to number theory problems.



professor. His main research interests are in neural networks, parallel processing, and applications of machine learning to information retrieval on the Internet. He has led a number of research projects on these themes with either national or international partners and has been involved in the organization of many scientific events, including the IEEE-INNS International Joint Conference on Neural Networks, for which he acted as the program chair. Dr. Gori serves (served) as an associate editor of a number of technical journals related to his areas of expertise, including *Pattern Recognition*, the *IEEE Transactions on Neural Networks*, *Neurocomputing*, *Neural Computing Survey*, *Pattern Analysis and Application*, the *International Journal on Computer Research*, and the *International Journal on Pattern Recognition and Artificial Intelligence*. He is the Italian chairman of the IEEE Neural Network Council (R.I.G.) and is acting as the cochair of the TC3 technical committee of the IAPR (International Association for Pattern Recognition) on Neural Networks and Machine Learning. Dr. Gori is a fellow of the IEEE.

► For further information on this or any computing topic, please visit our Digital Library at <http://computer.org/publications/dlib>.