Notes on Newton's Method After 1960

José Mario Martínez

Abstract Some Newtonian ideas will be reported with respect to research areas that emerged in numerical Mathematics after, approximately, 1960. For the problems of solving nonlinear equations and unconstrained optimization, Quasi-Newton methods, which stayed in the mainstream of numerical optimization for more than 30 years, will be motivated and discussed. The topic of complexity in unconstrained optimization will be introduced and some fundamental results will be rigorously proved. Newtonian algorithmic schemes in Linear Programming, which emerged after 1984 and presently represent competitive alternatives for large-scale problems, will be commented. Finally, surprising negative results concerning the capacity of Newton's method to detect approximate solutions of constrained optimization problems will be reported.

1 Introduction

Every undergraduate student of Mathematics, Physics, or Engineering learns that Newton's method is a powerful tool for solving equations and that this method converges very fast if the initial approximation is reasonably close to the solution. Moreover, in most situations, fast convergence occurs even if the initial approximation is poor. Fast convergence means, in general, "quadratic" convergence, a property that guarantees that the number of correct digits at some iteration approximately doubles the corresponding number at the previous one. Later, students learn that there are many methods in numerical mathematics that are called "Newtonian" or, simply, "Newton" for solving different practical problems. Popular knowledge about these methods guarantee that they all are "good," in the sense that they are very fast close to the solutions and that enjoy other theoretical and practical excellent properties. More recently, it became accepted the idea that, not only "every Newton

J. M. Martínez (✉)
Institute of Mathematics, Statistics and Scientific Computing, University of Campinas, Campinas, SP, Brazil
e-mail: martinez@ime.unicamp.br

C. Lavor, F. A. M. Gomes (eds.), *Advances in Mathematics and Applications*,
https://doi.org/10.1007/978-3-319-94015-1_9

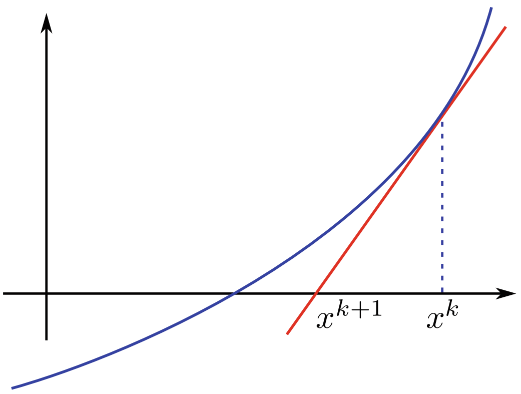

Fig. 1 Newton's iteration for solving a scalar equation

is good" but, also, "every good is Newton," because several methods that were well known as being effective and having nice convergence properties were shown to be, in one sense or another, versions of Newton's method [22].

Newton's idea is the following: Given a complicated problem and some approximation to its solution, one builds a simpler and solvable problem and we postulate that its solution is a better approximation to the solution of the original problem than the previously computed approximation. The simpler problem is built using information available at the current approximation.

The Newtonian paradigm can be applied to many problems, even nonmathematical ones. The most simple case consists of finding a solution of a scalar equation $g(x) = 0$. If x^k is an approximate solution, the presumably better approximation x^{k+1} is obtained by solving (when possible) the linear equation $g(x^k) + g'(x^k)(x - x^k) = 0$. See Fig. 1.

In the same way, we define the Newtonian iteration when the problem is to solve a nonlinear system of equations $g(x) = 0$, where $g : \mathbb{R}^n \to \mathbb{R}^n$. In this case, the Jacobian $g'(x^k)$ is an $n \times n$ matrix and finding the new iterate involves the solution of an $n \times n$ linear system of equations.

One of the most popular applications of solving nonlinear systems comes from the unconstrained minimization of scalar functions. If $f : \mathbb{R}^n \to \mathbb{R}$, $g = \nabla f$ is its gradient, and $H = g' = \nabla^2 f$ is its Hessian, the iterate defined by the solution of $g(x^k) + g'(x^k)(x - x^k) = 0$ defines a stationary point (perhaps a minimizer) of the quadratic approximation $f(x^k) + \langle g(x^k), x - x^k\rangle + \frac{1}{2}(x - x^k)^T H(x^k)(x - x^k)$.

Thus, the simple problem that corresponds to the minimization of an n-dimensional scalar function f consists of minimizing a quadratic function, a problem that, in turn, is roughly equivalent to solving a linear system of equations.

We use to say that the simple problem associated with every iteration of a Newtonian method is a Model of the original problem. In unconstrained optimization, Newton deals with quadratic models, although different interpretations are possible, as we will see later. Before 1960, it was believed that minimizing functions with more than 10 variables employing Newton's method was very hard because of complications solving "big" linear systems and the computation of second derivatives. These complications motivated the upraise of the quasi-Newton age, as we will see in Sect. 2.

2 Quasi-Newton Age

The quasi-Newton age arose around 1960 associated to the unconstrained minimization problem:

$$\text{Minimize } f(x), \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$. The Steepest Descent method (or Cauchy's method, or Gradient method [10]) for solving (1) proceeds, at each iteration, computing the gradient $g(x^k)$ and performing a "line search" along that direction with the aim of obtaining a better approximate solution:

$$x^{k+1} = x^k - t_k g(x^k), \tag{2}$$

where $t_k > 0$ is such that, at least, $f(x^{k+1}) < f(x^k)$. If $g(x^k)$ does not vanish, this condition is always verified if t_k is small enough because the direction $-g(x^k)$ is a "descent direction." Many alternatives exist for deciding the most convenient value of t_k. Cauchy's method is easy to implement and relatively cheap since performing one iteration only needs computation of function values and a gradient (no Hessians), while linear algebra calculations associated with (2) are trivial. Moreover, memory requirements for the implementation of (2) are minimal.

However, the sequences generated by the Cauchy method usually converge to stationary points of (1) (points where $g(x) = 0$) very slowly. This is because Cauchy's method reflects a "greedy" way of taking decisions. According to the steepest descent point of view, the decision maker stays at x^k, verifies the characteristics of its problem in a very small neighborhood of the present approximation, and takes a decision based only on such "myopic" observation. Of course, problems do not behave far from the actual approximation in the same way as they do close to it. For this reason, the number of iterations needed to achieve good solutions could be unacceptably large.

On the other hand, Newton's method seems to work in a very different way. Instead of taking a quick decision based on local considerations, Newton "stops to think" about the choice of a good model that, perhaps, should reflect problem features in a smarter way. This model will correspond to the minimization of the quadratic that coincides with the objective function f up to its second derivatives (not only the first ones). The consequence is that, in general, the goal of obtaining very good approximate solutions in a small number of iterations is achieved but, on the other hand, the computational effort to perform an iteration is considerably bigger than the one required by Cauchy.

The dream of the pioneers of the quasi-Newton age [13, 18] was to devise algorithms for solving (1) that, at the first iterations, behave as Cauchy and, at the end, behave as Newton. The rationale behind this idea is that, at the beginning, when we are probably far from the solution, there is no reason to lose a lot of time building or solving a good model, whereas, close to the solution (at the end), the Newtonian model reflects very accurately the original problem and, therefore, Newtonian iterations produce very fast approximations. (The local convergence properties of Newton's methods for solving nonlinear systems were known several decades ago.) On the other hand, quasi-Newton iterations should involve considerably less computational effort than Newton steps.

By (2), the gradient method with line searches takes the form

$$x^{k+1} = x^k - t_k H_k g(x^k) \tag{3}$$

with $H_k = I$ (the Identity matrix) for all $k \in \mathbb{N}$. Moreover, a line-search version of Newton's method also has the form (3) with $H_k = \nabla^2 f(x^k)^{-1}$. Should it be possible to devise a method of the form (3) in which $H_0 = I$ and $H_k \approx \nabla^2 f(x^k)^{-1}$ for k large with moderate computational cost per iteration? Methods with such purpose were ultimately called "quasi-Newton methods" and their development and analysis dominated mainstream research in computational optimization for more than three decades.

By the Mean Value Theorem, we have that:

$$\left[\int_0^1 \nabla^2 f(x^k + ts^k)dt\right] s^k = y^k, \tag{4}$$

where

$$s^k = x^{k+1} - x^k \text{ and } y^k = g(x^{k+1}) - g(x^k).$$

Then, since the matrix $[\int_0^1 \nabla^2 f(x^k + ts^k)dt]$ is an average of the Hessians of f in the segment $[x^k, x^{k+1}]$, it turns out that the Hessian $\nabla^2 f(x^{k+1})$ approximately satisfies the "secant equation"

$$Bs^k = y^k. \tag{5}$$

The secant system has n equations and n^2 unknowns (the entries of B). The number of unknowns can be reduced to $(n + 1)n/2$ considering that Hessians are symmetric matrices and (5) defines an affine subspace in the space of matrices. If B_k is an approximation to $\nabla^2 f(x^k)$, it is natural to define B_{k+1}, the approximation to $\nabla^2 f(x^{k+1})$, as some kind of projection of B_k on the affine subspace defined by the secant equation. For example, the BFGS method, which is the most popular quasi-Newton method for unconstrained minimization, is defined by:

$$x^{k+1} = x^k - t_k B_k^{-1} g(x^k)$$

and

$$B_{k+1} = B_k + \frac{y^k (y^k)^T}{(y^k)^T s^k} - \frac{B_k s^k (s^k)^T B_k}{(s^k)^T B_k s^k}. \tag{6}$$

The interpretation of (6) as a (variable with respect to x^k) projection on the set of solutions of (5) may be found in the classical book by Dennis and Schnabel [15].

The BFGS method may be defined without explicitly mentioning the inverse of any matrix, since the inverse of B_{k+1} in (6) can be computed in terms of the inverse of B_k by means of a judicious application of the Sherman–Morrison formula [19].

The line-search parameter t_k is used to guarantee sufficient descent of $f(x^{k+1})$ with respect to $f(x^k)$. Algorithms for choosing t_k differ in degrees of sophistication and cause different numerical behaviors of the methods so far implemented.

Quasi-Newton methods were generalized to solving arbitrary nonlinear systems of equations by Broyden [5] and many followers. Generalizations include taking advantage of specific structures (for example, nonlinear least squares problems), using sparsity patterns of Hessians or Jacobians, direct updates of factorizations [25], nonlinear systems coming from constrained optimization, and many others.

Roughly speaking, as one can expect quadratic local convergence from Newton's method, superlinear convergence is usually observed, and many times proved, in quasi-Newton algorithms. The pioneers' project of devising methods that smoothly evolve from Cauchy behavior to Newtonian behavior was only partially successful. Most practitioners believe that, when it is affordable to use Newton in unconstrained optimization or nonlinear systems, the Newton alternative is more efficient than quasi-Newton ones. The motivation for quasi-Newton methods decreased with the development of algorithmic differentiation [21], sparse matrix techniques [16], and the use of iterative methods for solving the Newtonian linear equation [14]. However, quasi-Newton ideas emerge frequently in modern optimization in combination with new techniques for multiobjective problems, equilibrium problems, constrained and nonsmooth optimization, and many others.

3 Linear Programming

Linear Programming is the problem of minimizing a linear function subject to linear inequalities and equalities. Every Linear Programming problem can be reduced to the Standard Form:

$$\text{Minimize } c^T x \text{ subject to } Ax = b \text{ and } x \geq 0. \tag{7}$$

A point $x \in \mathbb{R}^n$ is a solution of (7) if and only if it satisfies the KKT conditions:

$$c + A^T y - z = 0, x_j z_j = 0 \text{ for all } j = 1, \ldots, n, \tag{8}$$

for some $y \in \mathbb{R}^m$ and $z \geq 0$, together with the feasibility conditions

$$Ax = b, x \geq 0. \tag{9}$$

Moreover, if the Linear Programming problem has a solution, then one of its solutions is a vertex of the polytope defined by (9). See [28] and many other textbooks.

The latter property motivates the best known method for solving Linear Programming problems: The Simplex Method, invented by George Dantzig in 1949 [12], proceeds visiting vertices of the polytope (9) always reducing the objective function value. Since the number of vertices is finite, the Simplex Method finds a solution of (7), when such a solution exists, in a finite number of steps.

The Simplex Method was the standard procedure for solving Linear Programming problems until 1984 and, perhaps, still is. However, at least from the theoretical point of view, this method has a drawback: In the worst case, it may need to visit all the vertices of a polytope for finding a solution and, since the number of vertices grows exponentially with the number of variables, the computer time needed to solve a large problem may be, in the worst case, unaffordable. This drawback motivated, in 1979, the introduction of a new method by Khachiyan [24] who showed that a solution with arbitrary chosen precision can be found in polynomial time. However, Khachiyan's method was shown very soon to be ineffective in practical computations.

In 1984, Karmarkar [23] introduced a new method for Linear Programming, enjoying similar convergence properties as Khachiyan's method, for which he claimed that, especially for large problems, the performance was orders of magnitude better than the performance of the Simplex algorithm. His results and claims attracted the attention of the whole optimization community. Karmarkar's method, whose practical performance could not be reproduced by independent experiments, introduced new ideas, as projective transformations, approximation by means of interior points, and potential functions, that seemed to be in the kernel of polynomiality proofs and practical performance. Later, it was verified that the only new idea that was crucial both for proofs and for practical behavior was the interiority of the sequence of iterates generated by the method. See [20].

Independently of the eventual discard of the original Karmarkar's method, his work had the merit of motivating a lot of fruitful research that showed that challenging alternatives to the Simplex method may exist. Ultimately, the challenge of the so-called Interior Point methods motivated an enormous improvement in Simplex implementations.

Modern descriptions of Interior Point methods are closely related to the Newton paradigm. In fact, from (8) and (9) we may extract the nonlinear system of equations:

$$c + A^T y - z = 0, Ax = b, x_j z_j = 0 \text{ for all } j = 1, \ldots, n. \tag{10}$$

If (x, y, z) is a solution of (10) such that $x \geq 0$ and $z \geq 0$, we have that x is a solution of (7).

But, (10) is a nonlinear system of equations with $2n+m$ equations and unknowns, then using Newton's method is an interesting alternative for its solution. On the other hand, we are interested only in solutions such that $x \geq 0$, $z \geq 0$, which justifies the decision of starting with $x^0 > 0$, $z^0 > 0$ and to maintain $x^k > 0$, $z^k > 0$ throughout the calculations. It is not recommendable to admit $x_j^k = 0$ or $z_j^k = 0$ because, in this case, Newton's method would maintain $x_j^{k+1} = 0$ (or $z_j^{k+1} = 0$) for all k. Therefore, the pure Newtonian iterations must be modified in order to prefer the positivity (interiority) of x^k and z^k. This is usually done by means of the introduction of (close to 1) damping parameters. Namely, if (x^k, y^k, z^k) (with $x^k > 0$ and $z^k > 0$) is the current iteration and (d_x, d_y, d_z) is the increment computed by one iteration of Newton's method for solving the nonlinear system (10), we will compute:

$$(x^{k+1}, y^{k+1}, z^{k+1}) = (x^k + \theta_k d_x, y^k + \theta_k d_y, z^k + \theta_k d_z),$$

in such a way that the new iterate remains interior and the difference with respect to the pure Newton iterate is cautiously small.

The procedure described above is called Primal-Dual Affine-Scaling method. In Newtonian terms, this is a damped Newton method that preserves interiority. This method behaves well except when some variable x_j (or z_j) becomes close to zero when it must be positive at the solution. In order to understand the best succeeded procedures for improving the robustness of the primal-dual affine-scaling method, let us assume first that (x^k, y^k, z^k) is such that

$$c + A^T y^k - z^k = 0,\ Ax^k = b,\ x^k > 0,\ \text{and } z^k > 0.$$

Clearly, (x^k, y^k, z^k) is a solution of the nonlinear system

$$c + A^T y - z = 0,\ Ax = b,\ x_j z_j = x_j^k z_j^k,\ j = 1, \ldots, n.$$

This means that we already know a solution of the system

$$c + A^T y - z = 0,\ Ax = b,\ x_j z_j = t x_j^k z_j^k,\ j = 1, \ldots, n \tag{11}$$

with $x > 0$, $z > 0$, for $t = 1$, whereas we wish a solution for $t = 0$. The Primal-Dual Affine-Scaling (Newton) step is an aggressive attempt of achieving the solution for $t = 0$. If this attempt is considered to be unsuccessful (for some more or less theoretical justified criterion), the natural procedure is to try a less ambitious value of $t > 0$. For approximating the solution of (11) for the new value of t, starting from an iterate (x^k, y^k, z^k), a Newton-like iteration is also employed that may use the same matrix factorization as the one employed for finding the Primal-Dual Affine-Scaling step. Variations of this idea define the best succeeded modern Interior Point methods for Linear Programming. It is remarkable that a problem traditionally solved by means of a combinatorial procedure as Simplex, later challenged by the nonstandard ideas of Khachiyan and Karmarkar, eventually found in the Newton paradigm one of the most promising solution tools for many difficult, especially large-scale, situations.

4 Convergence and Complexity in Unconstrained Optimization

Numerical methods for solving general continuous optimization problems are iterative. Since finding global minimizers without employing the specific structure of the problems is very difficult, we generally rely on methods that guarantee convergence to points that satisfy necessary optimality conditions (hopefully, local minimizers). Classical convergence theories analyze the sequences generated by optimization methods and prove that the sequence of gradients tend to zero or that the gradient vanishes at limit points. These "global" theories say nothing about the speed of convergence. Many times they are complemented with "capture theorems" that say that, when an iterate is close enough to a local minimizer with good properties, convergence to the local minimizer takes place with satisfactory convergence rate.

Only recently, it has been considered to be relevant to compute bounds for the computer effort that is necessary to achieve a predetermined precision ε. For example, if one assumes that "precision ε" means that the norm of the gradient is smaller than ε, the question is about the number of iterations and function-gradient evaluations that are necessary to achieve such precision, as a function of ε, the functional value at the initial point, characteristics of the problem, and parameters of the method.

Being a bit more formal than in the previous sections, we will assume here that $f : \mathbb{R}^n \to \mathbb{R}$ has continuous first derivatives $g(x) = \nabla f(x)$ and that a Lipschitz inequality for the gradient holds. As a consequence, by Elementary Calculus, there exists $\gamma > 0$ such that

$$f(x+s) \leq f(x) + g(x)^T s + \gamma \|s\|^2. \tag{12}$$

This assumption is "slightly" weaker than saying that f has bounded second derivatives on $\mathbb{R}^n$. We are going to analyze the worst-case complexity of a version of Cauchy's method, with the aim of relating this analysis with analogous analyses concerning Newton's method.

There is a reason for considering that Cauchy's method is also a Newton-like method: In Newton, for minimizing functions, we use to say that the objective function is approximated, locally, by a quadratic model. Analogously, in Cauchy we may think that we approximate the objective function, locally, by a linear model.

Equivalently, in Newton we approximate the gradient by a linear model whereas in Cauchy we approximate the gradient by a constant vector, namely, the gradient at the current iterate.

The difficulty in this point of view about Cauchy is that, in general, linear functions do not admit minimizers. Therefore, the "Newtonian subproblem" cannot be solved. We will fix this inconvenience observing that, although the linear model $g(x^k)^T(x - x^k)$ does not have a minimizer, the "regularized" version of this model: $g(x^k)^T(x - x^k) + \frac{\rho}{2}\|x - x^k\|^2$ has a unique solution independently of the value of the regularizing parameter $\rho > 0$. Moreover, if we choose $\|\cdot\|$ as the Euclidian norm $\|\cdot\|_2$, the minimizer of $g(x^k)^T(x - x^k) + \frac{\rho}{2}\|x - x^k\|^2$ is given by $x = x^k - \frac{1}{\rho}g(x^k)$. This idea is formalized in the following algorithm.

Algorithm 4.1
Let $x^0 \in \mathbb{R}^n$ and $\alpha > 0$ be given. Initialize $k \leftarrow 0$.

Step 1 Set $\rho \leftarrow 1/2$.
Step 2 Solve the subproblem

$$\text{Minimize } g(x^k)^T s + \rho\|s\|^2,$$

obtaining the solution s^{trial}. (Note that $s^{trial} = -\frac{1}{2\rho}g(x^k)$ if $\|\cdot\| = \|\cdot\|_2$.)
Step 3 (Test the sufficient descent condition)

If

$$f(x^k + s^{trial}) \le f(x^k) - \alpha\|s^{trial}\|^2, \tag{13}$$

set $s^k = s^{trial}$, $x^{k+1} = x^k + s^k$, $k \leftarrow k + 1$, and go to Step 1.
Otherwise, set $\rho \leftarrow 2\rho$ and go to Step 2.

When $\|\cdot\|$ is the Euclidian norm, Algorithm 4.1 is Cauchy's method with the most simple line-search procedure (backtracking dividing the trial step by 2) and the clothes of regularization.

Lemma 4.1 *If $\rho \ge \gamma + \alpha$, the sufficient descent condition (13) is fulfilled.*

Proof By (12), the hypothesis of this lemma, and Step 2 of the algorithm,

$$\begin{aligned} f(x^k + s) &\le f(x^k) + g(x^k)^T s + \gamma\|s\|^2 \\ &= f(x^k) + g(x^k)^T s + (\gamma + \alpha)\|s\|^2 - \alpha\|s\|^2 \\ &\le f(x^k) + g(x^k)^T s + \rho\|s\|^2 - \alpha\|s\|^2 \le f(x^k) - \alpha\|s\|^2. \end{aligned}$$

This completes the proof. □

By Lemma 4.1, the first term in the sequence $\{1/2, 1, 2, 4, 8\ldots\}$ bigger than $\gamma + \alpha$ necessarily defines a value of ρ for which (13) holds. As a consequence, the following corollary holds.

Corollary 4.1 *At each iteration of Algorithm 4.1, after a maximum of* $1 + \log_2(\gamma + \alpha)$ *tests (backtrackings, functional evaluations) we necessarily obtain the descent condition and the final* ρ *for which (13) holds and satisfies:*

$$\rho < 2(\gamma + \alpha).$$

For the sake of simplicity, assume now that $\|\cdot\| = \|\cdot\|_2$. Then, $s^{trial} = -\frac{1}{2\rho} g(x^k)$ and, so, by Corollary 4.1, $\|s^k\| \geq \frac{1}{4(\gamma+\alpha)} \|g(x^k)\|$. Then, by the sufficient descent condition,

$$f(x^{k+1}) \leq f(x^k) - \frac{\alpha}{16(\gamma + \alpha)^2} \|g(x^k)\|^2.$$

Then, if $\|g(x^k)\| \geq \varepsilon$,

$$f(x^{k+1}) \leq f(x^k) - \frac{\alpha}{16(\gamma + \alpha)^2} \varepsilon^2. \tag{14}$$

Assume that $f_{target} < f(x^0)$ is arbitrary. Then, (14) implies that the number of iterations at which $\|g(x^k)\| \geq \varepsilon$ and $f(x^k) > f_{target}$ is bounded by:

$$[f(x^0) - f_{target}] \frac{16(\gamma + \alpha)^2}{\alpha} \varepsilon^{-2}. \tag{15}$$

Thus, by Corollary 4.1, the number of evaluations is bounded by:

$$[f(x^0) - f_{target}][1 + \log_2(\gamma + \alpha)] \frac{16(\gamma + \alpha)^2}{\alpha} \varepsilon^{-2}. \tag{16}$$

Both expressions (15) and (16) have the form $c\varepsilon^{-2}$, where c is a constant that only depends on characteristics of the problem (γ), parameters of the algorithm (α), the initial point x^0, and, of course, the target with respect to which we desired to estimate the computational effort. The dependence on the precision required is represented by ε^{-2}. For this reason, we generally say that the complexity of the algorithm is $O(\varepsilon^{-2})$.

"Gradient-related" methods for unconstrained optimization are characterized by the generation of directions d^k that are related to $g(x^k)$ by means of angle and relative-size conditions as:

$$g(x^k)^T d^k \leq -\theta \|g(x^k)\|_2 \|d^k\|_2 \text{ and } \|d^k\| \geq \beta \|g(x^k)\|, \tag{17}$$

where $\theta \in (0, 1)$ and $\beta > 0$ are algorithmic parameters. These conditions are sufficient to show that gradient-related methods have complexity $O(\varepsilon^{-2})$.

Quasi-Newton methods (as BFGS) also enjoy the worst-case complexity $O(\varepsilon^{-2})$ when conveniently safeguarded in order to satisfy gradient-related conditions. The standard BFGS method (without safeguards) does not satisfy such property. In fact, there exist counterexamples that show that all the limit points generated by this popular method may be such that the norm of the gradient does not vanish at all [11, 26]. Of course, this prevents the possibility of satisfactory complexity results.

Newton's method with line searches may also generate sequences with associated gradients that are bounded away from zero [27]. This cannot happen when Newton's method is coupled with a "trust-region" strategy, which guarantees that every limit point is stationary. In spite of this, it has been shown that even Newton's method with the robust trust-region strategy has worst-case complexity not better than $O(\varepsilon^{-2})$ [6].

It is disappointing that, with Newton's method plus traditional globalization procedures, a complexity better than $O(\varepsilon^{-2})$ cannot be obtained. Fortunately, the reason is that the "traditional globalization procedures" *are not* the natural globalization procedures that should be used for Newton. Mimicking the complexity proof given for Cauchy, we will show that a better complexity result may be obtained for Newton, if one replaces quadratic regularization with cubic regularization and quadratic sufficient descent with cubic convergence descent with respect to $\|s^{trial}\|$. Analogous results with variations with respect to the sufficient descent criterion were given in [3, 7, 29].

In order to define Algorithm 4.2, assume that the Hessian $\nabla^2 f(x)$ exists for all $x \in \mathbb{R}^n$.

Algorithm 4.2
Let $x^0 \in \mathbb{R}^n$ and $\alpha > 0$ be given. Initialize $k \leftarrow 0$.

Step 1 Set $\rho \leftarrow 0$.
Step 2 Solve the subproblem

$$\text{Minimize } g(x^k)^T s + \frac{1}{2} s^T \nabla^2 f(x^k) s + \rho \|s\|^3,$$

obtaining the solution s^{trial}. If the subproblem has no solution (which may occur only if $\rho = 0$), reset $\rho \leftarrow 1$ and repeat Step 2.
Step 3 (Test the sufficient descent condition)

If

$$f(x^k + s^{trial}) \le f(x^k) - \alpha \|s^{trial}\|^3, \tag{18}$$

set $s^k = s^{trial}$, $x^{k+1} = x^k + s^k$, $k \leftarrow k+1$, and go to Step 1.
Otherwise, set $\rho \leftarrow 2\rho$ and go to Step 2.

The complexity proof for Algorithm 4.2 needs to assume that the Hessian $\nabla^2 f$ is Lipschitz continuous for all $x \in \mathbb{R}^n$. This implies, as in the case of (12), that there exists $\gamma_2 > 0$ such that, for all $x, s \in \mathbb{R}^n$,

$$f(x+s) \le f(x) + g(x)^T s + \frac{1}{2} s^T \nabla^2 f(x) s + \gamma_2 \|s\|^3 \tag{19}$$

and

$$\|g(x+s)\| \leq \|g(x) + \nabla^2 f(x)s\| + \gamma_2 \|s\|^2. \tag{20}$$

Lemma 4.2 below is entirely analogous to Lemma 4.1.

Lemma 4.2 *If $\rho \geq \gamma_2 + \alpha$, the sufficient descent condition (18) is fulfilled.*

Proof By (19), the hypothesis of this lemma, and Step 2 of the algorithm,

$$\begin{aligned} f(x^k + s) &\leq f(x^k) + g(x^k)^T s + \frac{1}{2} s^T \nabla^2 f(x^k) s + \gamma_2 \|s\|^3 \\ &= f(x^k) + g(x^k)^T s + \frac{1}{2} s^T \nabla^2 f(x^k) s + (\gamma_2 + \alpha)\|s\|^3 - \alpha \|s\|^3 \\ &\leq f(x^k) + g(x^k)^T s + \frac{1}{2} s^T \nabla^2 f(x^k) s + \rho \|s\|^3 - \alpha \|s\|^3 \\ &\leq f(x^k) - \alpha \|s\|^3. \end{aligned}$$

This completes the proof. □

Moreover, as in Corollary 4.1, we have:

Corollary 4.2 *At each iteration of Algorithm 4.2, after a maximum of $1 + \log_2(\gamma_2 + \alpha)$ tests we necessarily obtain the descent condition (18) with*

$$\rho < 2(\gamma_2 + \alpha).$$

By Corollary 4.2, after computer time that only depends on γ_2 (characteristic of the problem) and α (characteristic of the algorithm), we obtain a decrease at least $\alpha \|s^{trial}\|^3$. Recall that in Algorithm 4.1 the corresponding decrease was $\alpha \|s^{trial}\|^2$. In Algorithm 4.1, our proof finished showing that $\|s^k\|$ was bigger than a multiple of $\|g(x^k)\|$. Here, we will show that $\|s^k\|$ is bigger than a multiple of $\|g(x^k + s^k)\|^{\frac{1}{2}}$. In other words, we will prove that $\|g(x^k + s^k)\|$ is smaller than a multiple of $\|s^k\|^2$. In fact, by (20),

$$\|g(x^k + s^k)\| \leq \|g(x^k) + \nabla^2 f(x^k) s^k\| + \gamma_2 \|s^k\|^2.$$

So, assuming, for simplicity, that $\|\cdot\| = \|\cdot\|_2$, using that $\nabla(\|s\|^3) = 3s\|s\|$, and the fact that gradient of the objective function of the subproblem must vanish at s^k, we have that:

$$\begin{aligned} \|g(x^k + s^k)\| &\leq \left\| g(x^k) + \nabla^2 f(x^k) s^k + 3\rho s^k \|s^k\| \right\| + 3\rho \|s^k\|^2 + \gamma_2 \|s^k\|^2 \\ &= (3\rho + \gamma_2)\|s^k\|^2. \end{aligned}$$

Then, since the final ρ accepted at (18) is smaller than $2(\gamma + \alpha)$,

$$\|g(x^k + s^k)\| \leq [6(\gamma_2 + \alpha) + \gamma_2]\|s^k\|^2.$$

Thus,

$$\|s^k\| \geq \frac{\|g(x^{k+1})\|^{\frac{1}{2}}}{\sqrt{[6(\gamma_2 + \alpha) + \gamma_2]}}.$$

By (18), this implies that, at each iteration of Algorithm 4.2,

$$f(x^{k+1}) \leq f(x^k) - \alpha \frac{\|g(x^{k+1})\|^{3/2}}{[6(\gamma_2 + \alpha) + \gamma_2]^{3/2}}.$$

Therefore, the number of iterations for which $\|g(x^{k+1})\| \geq \varepsilon$ and $f(x^{k+1}) \geq f_{target}$ is bounded above by $c[f(x^0) - f_{target}]\varepsilon^{-3/2}$, where c is a constant that only depends on γ_2 and α. As in the case of Algorithm 4.1, by Corollary 4.2, this implies that the worst-case complexity of the Newtonian Algorithm 4.2 is $O(\varepsilon^{-3/2})$.

This rather simple result, as several analogous ones [3, 7, 17, 29], confirms the intuition that some version of Newton's method should have better worst-case complexity than every gradient-related method.

A straightforward generalization of Algorithm 4.2 consists of replacing the subproblem with the minimization of the q-th Taylor polynomial plus a regularization of the form $\rho\|s\|^{q+1}$ and replacing (18) with

$$f(x^k + s^{trial}) \leq f(x^k) - \alpha\|s^{trial}\|^{q+1}.$$

Assuming Lipschitz conditions on the derivatives of order q and following, mutatis mutandi, the proof for the case $q = 2$, we obtain an algorithm with complexity $O(\varepsilon^{(q+1)/q})$. Slight variations of this algorithm have been given in [3].

5 Newton in Constrained Optimization

The smooth constrained optimization problem consists of minimizing a smooth function $f(x)$ subject to $h(x) = 0$ and $g(x) \leq 0$, where $h : \mathbb{R}^n \to \mathbb{R}^m$ and $g : \mathbb{R}^n \to \mathbb{R}^p$ are, also, sufficiently smooth. For simplicity, this section will be restricted to the case in which there are not inequality constraints, although all the arguments apply straightforwardly to the case $p > 0$. Then, the problem considered here is

$$\text{Minimize } f(x) \text{ subject to } h(x) = 0. \qquad (21)$$

In unconstrained optimization, it is quite natural to require, as stopping criterion for computer algorithms, the condition $\|\nabla f(x^k)\| \le \varepsilon$ because $\nabla f(x) = 0$ is a necessary condition for every local minimizer. However, in constrained optimization we have the additional requirement on the feasibility of the approximate solution and, moreover, a computable necessary condition based on the gradients of f and the constraints does not exist. In fact, the Lagrange conditions (called KKT in the presence of inequalities) establish that

$$\nabla f(x) + \sum_{i=1}^{m} \lambda_i \nabla h_i(x) = 0 \tag{22}$$

should hold for suitable multipliers $\lambda \in \mathbb{R}^m$, but these conditions are guaranteed to hold at a minimizer only if such point satisfies a "constraint qualification." For example, the problem of minimizing x subject to $x^2 = 0$ has an obvious global minimizer at $x = 0$, but (22) does not hold.

This inconvenience raises the question about the practical convergence test that should be used in numerical algorithms designed to solve (21). Some authors employ stopping criteria based on "scaled KKT conditions." Instead of requiring that

$$\left\| \nabla f(x) + \sum_{i=1}^{m} \lambda_i^k \nabla h_i(x) \right\| \le \varepsilon \tag{23}$$

they stop their algorithms when

$$\frac{1}{\max\{1, \|\lambda^k\|_\infty\}} \left\| \nabla f(x) + \sum_{i=1}^{m} \lambda_i^k \nabla h_i(x) \right\| \le \varepsilon, \tag{24}$$

a weaker condition than (23) that may hold close to a minimizer at which constraint qualifications are not fulfilled [8, 9]. However, (24) may hold in simple problems at points that are arbitrarily far from the solution. For example, consider the problem of minimizing x^2 subject to $0x = 0$ and take $x^k = 10^{20}$. Clearly, (24) holds with $\varepsilon = 10^{-10}$ and $\lambda^k = 2 \times 10^{30}$. Thus, the criterion (23) may be useful to save computer work when convergence of an algorithm is in fact occurring to a correct minimizer but may also lead to incorrect decisions when the iterate is far from a solution.

Fortunately, an interesting result concerning the approximate fulfillment of (22) at local minimizers exists. Although local minimizers may not satisfy (22), they do satisfy the approximate version of this system of equations. By this we mean that, if x^* is a local minimizer, given $\varepsilon > 0$ arbitrary small, there exist $x \in \mathbb{R}^n$ and $\lambda \in \mathbb{R}^m$ such that $\|x - x^*\| \le \varepsilon$, $\|h(x)\| \le \varepsilon$, and $\|\nabla f(x) + \sum_{i=1}^{m} \lambda_i \nabla h_i(x)\| \le \varepsilon$. See [1, 4] and other papers that study Sequential Optimality Conditions.

As a consequence, the following is a well-justified stopping criteria for algorithms that aim to solve (21):

$$\|h(x^k)\| \leq \varepsilon, \quad \left\| \nabla f(x^k) + \sum_{i=1}^{m} \lambda_i^k \nabla h_i(x^k) \right\| \leq \varepsilon. \tag{25}$$

The natural question that arises is: Given a particular algorithm for solving (21) that converges to a minimizer x^*, is it possible to prove that, for all $\varepsilon > 0$, an iterate x^k, associated with suitable multipliers λ^k, exists? If the answer is positive, the algorithm will eventually stop satisfying (25). It has been proved that this is the case of penalty and Augmented Lagrangian algorithms [4]. Surprisingly, it can be shown in very simple examples that, when Newton's method is applied to the nonlinear system that includes $h(x) = 0$ and (22), the resulting sequence x^k may converge to a minimizer of (21) but the KKT-residual $\|\nabla f(x^k) + \sum_{i=1}^m \tilde{\lambda}_i^k \nabla h_i(x^k)\|$ is bounded away from zero independently of the value of $\tilde{\lambda}^k$. This means that, even converging to the solution, Newton's method would never detect that such convergence occurs [2]. Therefore, no complexity results associated with the condition (25) is possible for Newton's method. The example given in [2] consists of minimizing x_1 subject to $\|x\|_2^2 = 0$, with $n \geq 2$. Starting with $\lambda^1 > 0$, the sequence generated by Newton's method converges to the solution $x = 0$ but the norm of the KKT-residual is bounded away from zero (bigger than $(\sqrt{5}-1)/4$ if $n = 2$) for most initial choices of x^0. The simplicity of this example is amazing and suggests that this "failure" of Newton's method might occur frequently in practical problems in which optimality cannot be easily detected by other means.

Acknowledgements I am indebted to Lúcio Tunes dos Santos and Luiz Rafael Santos for revising the first version of this manuscript.

This work was supported by Cepid-Cemeai-Fapesp, PRONEX-CNPq/FAPERJ E-26/111.449/2010-APQ1, FAPESP (grants 2010/10133-0, 2013/03447-6, 2013/05475-7, and 2013/07375-0), and CNPq.

References

1. R. Andreani, G. Haeser, and J. M. Martínez, On sequential optimality conditions for smooth constrained optimization, *Optimization* **60**, pp. 627–641 (2011).
2. R. Andreani, J. M. Martínez, and L. T. Santos, Newton's method may fail to recognize proximity to optimal points in constrained optimization, To appear in *Mathematical Programming*.
3. E. G. Birgin, J. L. Gardenghi, J. M. Martínez, S. A. Santos, and Ph. L. Toint, Worst-case evaluation complexity for unconstrained nonlinear optimization using high-order regularized models, to appear in *Mathematical Programming*.
4. E. G. Birgin and J. M. Martínez, *Practical Augmented Lagrangian Methods for Constrained Optimization*, SIAM Publications, Series: Fundamentals of Algorithms, Philadelphia, 2014.
5. C. G. Broyden, A class of methods for solving nonlinear simultaneous equations, *Mathematics of Computation* **19**, pp. 577–593 (1965).

6. C. Cartis, N. I. M. Gould, and Ph. L. Toint, On the complexity of steepest descent, Newton's and regularized Newton's methods for nonconvex unconstrained optimization, *SIAM Journal on Optimization* **20**, pp. 2833–2852 (2010).
7. C. Cartis, N. I. M. Gould, and Ph. L. Toint, Adaptive cubic regularization methods for unconstrained optimization, *Mathematical Programming* **127**, pp. 245–295 (2011).
8. C. Cartis, N. I. M. Gould, and Ph. L. Toint, On the complexity of finding first-order critical points in constrained nonlinear optimization, *Mathematical Programming* **144**, pp. 93–106 (2014).
9. C. Cartis, N. I. M. Gould, and Ph. L. Toint, Corrigendum: On the complexity of finding first-order critical points in constrained nonlinear optimization, Preprint RAL-P-2014-013 (2014), Rutherford Appleton Laboratory, Chilton, England.
10. A. Cauchy, Méthode générale pour la résolution des systèmes d'équations simultaneés, *Computes Rendus de l'Académie des Sciences* **27**, pp. 536–538 (1847).
11. Y.-H. Dai, Convergence properties for the BFGS algorithm, *SIAM Journal on Optimization* **13**, pp. 693–701 (2002).
12. G. B. Dantzig, *Linear Programming and Extensions*, Princeton University Press, Princeton, N. J., 1963.
13. W. C. Davidon, Variable metric method for minimization, AEC Research and Development Report, Argonne National Laboratory ANL-5990, 1959.
14. R. S. Dembo, S. C. Eisenstat, and T. Steihaug, Inexact Newton Methods, *SIAM Journal on Numerical Analysis* **19**, pp. 400–408 (1982).
15. J. E. Dennis, Jr. and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, volume 16 of *Classics in Applied Mathematics*, SIAM, Philadelphia, 1996.
16. I. S. Duff, A. M. Erisman, and J. K. Reid, *Direct Methods for Sparse Matrices*, Oxford University Press, New York and Oxford, 1986.
17. J. P. Dussault, Simple unified convergence proofs for the trust-region and a new ARC variant, Technical Report, University of Sherbrooke, Sherbrooke, Canada, 2015.
18. R. Fletcher and M. J. D. Powell, A rapidly convergent descent method for minimization, *Computer Journal* **6**, pp. 163–168 (1963).
19. G. H. Golub and Ch. F. Van Loan, *Matrix computations*, Johns Hopkins University Press, Baltimore, 2012.
20. C. C. Gonzaga, Path-Following Methods for Linear Programming, *SIAM Review* **34**, pp. 167–224 (1992).
21. A. Griewank, *Automatic Differentiation*, Princeton Companion to Applied Mathematics, Nicolas Higham Ed., Princeton University Press, 2014.
22. A. F. Izmailov and M. V. Solodov, *Newton-Type Methods for Optimization and Variational Problems*, Springer Series in Operations Research and Financial Engineering, New York, 2014.
23. N. Karmarkar, A new polynomial-time algorithm for linear programming, *Combinatorics* **4**, pp. 373–395 (1984).
24. L. G. Khachiyan, A polynomial algorithm in linear programming, *Doklady Akad. Nauk. USSR* **244**, pp. 1093–1096 (1979).
25. J. M. Martínez, A family of quasi-Newton methods with direct secant updates of matrix factorizations, *SIAM Journal on Numerical Analysis* **27**, pp. 1034–1049 (1990).
26. W. F. Mascarenhas, The BFGS method with exact line searches fails for non-convex objective functions, *Mathematical Programming* **99**, pp. 49–61 (2004).
27. W. F. Mascarenhas, On the divergence of line search methods, *Computational and Applied Mathematics* **26**, pp. 129–169 (2007).
28. J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer Series in Operations Research and Financial Engineering, New York, Second Edition, 2006.
29. Y. Nesterov and B. T. Polyak, Cubic regularization method and its global performance, *Mathematical Programming* **108**, pp. 177–205 (2006).