Optimization Letters
https://doi.org/10.1007/s11590-019-01395-z

ORIGINAL PAPER

On the use of third-order models with fourth-order regularization for unconstrained optimization

E. G. Birgin[1] · **J. L. Gardenghi**[1] · **J. M. Martínez**[2] · **S. A. Santos**[2]

Received: 7 November 2017 / Accepted: 20 January 2019

Abstract
In a recent paper (Birgin et al. in Math Program 163(1):359–368, 2017), it was shown that, for the smooth unconstrained optimization problem, worst-case evaluation complexity $O(\epsilon^{-(p+1)/p})$ may be obtained by means of algorithms that employ sequential approximate minimizations of p-th order Taylor models plus $(p+1)$-th order regularization terms. The aforementioned result, which assumes Lipschitz continuity of the p-th partial derivatives, generalizes the case $p = 2$, known since 2006, which has already motivated efficient implementations. The present paper addresses the issue of defining a reliable algorithm for the case $p = 3$. With that purpose, we propose a specific algorithm and we show numerical experiments.

Keywords Unconstrained minimization · Third-order models · Regularization · Complexity

In memory of Chris Floudas, who is being missed a lot because of his friendship, enthusiasm, and daring initiative.

✉ E. G. Birgin
egbirgin@ime.usp.br

J. L. Gardenghi
john@ime.usp.br

J. M. Martínez
martinez@ime.unicamp.br

S. A. Santos
sandra@ime.unicamp.br

[1] Department of Computer Science, Institute of Mathematics and Statistics, University of São Paulo, Rua do Matão, 1010, Cidade Universitária, 05508-090 São Paulo, SP, Brazil

[2] Department of Applied Mathematics, Institute of Mathematics, Statistics, and Scientific Computing, University of Campinas, Campinas, SP, Brazil

Published online: 31 January 2019

1 Introduction

In 2006, Nesterov and Polyak [33] introduced a version of Newton's method for unconstrained optimization with worst-case evaluation complexity $O(\varepsilon^{-3/2})$. This means that the number of functional, gradient, and Hessian evaluations necessary to obtain a gradient norm smaller than ε is at most $c(f(x^0) - f_{\text{low}})\varepsilon^{-3/2}$, where x^0 is the initial guess, f_{low} is a lower bound for f, and c is a constant that only depends on parameters of the algorithm and characteristics of the problem. For obtaining this result one needs to assume Lipschitz-continuity of second-order derivatives and the main tool of the algorithm is the employment of cubic regularization, formerly addressed by Griewank [28]. Later, Cartis, Gould, and Toint [20–22] and other authors [17,24,26, 27,29] introduced practical algorithms with the same theoretical property employing cubic regularization [21,22], non-naive trust regions [24], or safeguarded quadratic regularization [17] techniques.

In [9], it was proved that, by means of a generalization of the ARC (acronym for Adaptive Regularization by Cubics) technique [21,22] and using subproblems based on p-th Taylor approximations to the objective function with a $(p+1)$-order regularization, it is possible to obtain complexity $O(\varepsilon^{-(p+1)/p})$, if one assumes Lipschitz-continuity of the derivatives of order p. The present paper investigates the practicality of an algorithm with this property, using $p = 3$. In other words, in the present work it is introduced, implemented, and evaluated an algorithm that possesses worst-case evaluation complexity $O(\varepsilon^{-4/3})$. At each iteration, the proposed algorithm computes a step that approximately minimizes the third-order Taylor approximation to the objective function with a fourth-order regularization, and the step satisfies a suitable sufficient descent criterion. We provide a simple proof showing that the worst-case complexity results of [9] are preserved and we describe a practical implementation for the case $p = 3$, accompanied by numerical experiments.

This work is organized as follows. Section 2 introduces the proposed algorithm. Well-definiteness and complexity results are given in Sect. 3. A variant of the proposed algorithm that uses a different approach to update the regularization parameter is introduced in Sect. 4. Numerical experiments are presented and analyzed in Sect. 5. The last section gives some conclusions and lines for future research.

Notation $\|\cdot\|$ represents an arbitrary norm on $\mathbb{R}^n$.

2 Model algorithm

Consider the unconstrained minimization problem given by

$$\underset{x\in\mathbb{R}^n}{\text{Minimize}}\ f(x), \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$. Given $s \in \mathbb{R}^n$, we denote by $T_p(x, s)$ the p-th Taylor polynomial of $f(x + s)$ around x, given by

$$T_p(x, s) \overset{\text{def}}{=} f(x) + \sum_{j=1}^{p} \frac{1}{j!} P_j(x, s),$$

where $P_j(x,s)$ is the homogeneous polynomial of degree j defined by

$$P_j(x,s) \stackrel{\text{def}}{=} \left(s_1 \frac{\partial}{\partial x_1} + \cdots + s_n \frac{\partial}{\partial x_n} \right)^j f(x).$$

We also define the regularized model

$$m_p(x,s,\sigma) \stackrel{\text{def}}{=} T_p(x,s) + \frac{\sigma}{p+1}\|s\|^{p+1}, \tag{2}$$

where the nonnegative scalar σ plays the role of the regularization parameter. Algorithm 2.1 below is a high-order adaptive-regularization algorithm for tackling problem (1) that, at each iteration, in the same spirit of [9], computes an approximate minimizer of the regularized model (2). Meaningful variations with respect to [9] are that (a) the algorithm proposed here accepts zero as a value for the regularization parameter σ; (b) it employs a step control strategy that, in practice, prevents the algorithm of evaluating the objective function at points not prone to cause a significant decrease in the objective function; and (c) the acceptance of the step is based on a $(p+1)$-th order sufficient descent condition instead of the actual-versus-predicted reduction condition considered in [9]. In the following description, we present the main model algorithm without stopping criterion, so that, in principle, the algorithm may perform infinitely many iterations.

Algorithm 2.1: AR$_{p+1}$—Adaptive regularization of order $p+1$

Input. Let $x^0 \in \mathbb{R}^n$, $p \in \{1,2,\ldots\}$, $\alpha > 0$, $\eta_1, \eta_2 > 0$, $\sigma_{\text{low}} > 0$, $\theta > 0$, $J \in \mathbb{N}$, and $\gamma > 1$ be given. Initialize $k \leftarrow 0$.

Step 1. Define $\sigma_{k,0} = 0$ and initialize $j \leftarrow 0$.

Step 2. Compute, if possible, $s^{k,j}$ such that

$$m_p(x^k, s^{k,j}, \sigma_{k,j}) \le m_p(x^k, 0, \sigma_{k,j}) \tag{3}$$

and

$$\|\nabla_s m_p(x^k, s^{k,j}, \sigma_{k,j})\| \le \theta \|s^{k,j}\|^p. \tag{4}$$

If $j = 0$ and finding $s^{k,j}$ satisfying (3) and (4) was not possible, define $\sigma_{k,1} = \sigma_{\text{low}}$, set $j \leftarrow 1$, and repeat Step 2.

Step 3. If

$$j \ge J \quad \text{or} \quad \left(\frac{T_p(x^k,0) - T_p(x^k, s^{k,j})}{\max\left\{1, |T_p(x^k,0)|\right\}} \le \eta_1 \quad \text{and} \quad \frac{\|s^{k,j}\|_\infty}{\max\left\{1, \|x^k\|_\infty\right\}} \le \eta_2 \right), \tag{5}$$

go to Step 4. Otherwise, define $\sigma_{k,j+1} = \max\{\sigma_{\text{low}}, \gamma\sigma_{k,j}\}$, set $j \leftarrow j+1$, and go to Step 2.

Step 4. If

$$f(x^k + s^{k,j}) \leq f(x^k) - \alpha \|s^{k,j}\|^{p+1}, \tag{6}$$

go to Step 5. Otherwise, define $\sigma_{k,j+1} = \max\{\sigma_{\text{low}}, \gamma\sigma_{k,j}\}$, set $j \leftarrow j + 1$, and go to Step 2.

Step 5. Define $\sigma_k = \sigma_{k,j}$, $s^k = s^{k,j}$, and $x^{k+1} = x^k + s^k$, set $k \leftarrow k + 1$, and go to Step 1.

When it comes to the model minimization of Step 2, one should notice that the model $m_p(x^k, s, \sigma_{k,j})$ is a smooth function of s with bounded level sets, except, perhaps, for $j = 0$, when $\sigma_{k,j} = 0$. Whenever $m_p(x^k, s, \sigma_{k,j})$ has bounded level sets, it has at least one global minimizer, at which its functional value is upper bounded by $m_p(x^k, 0, \sigma_{k,j})$ and its gradient vanishes. Therefore, in this case, it is possible to find $s^{k,j}$ satisfying (3) and (4) within a finite number of operations using an algorithm for unconstrained minimization that possesses worst-case complexity.

Concerning the step control strategy of Step 3, a few considerations are in order. Since $T_p(x^k, 0) = f(x^k)$, the condition

$$\frac{T_p(x^k, 0) - T_p(x^k, s^{k,j})}{\max\left\{1, |T_p(x^k, 0)|\right\}} \leq \eta_1$$

in (5) discards, without computing $f(x^k + s^{k,j})$, steps $s^{k,j}$ for which the predicted decrease is much larger than a quantity proportional to the current value of the objective function, since this may suggest that the model is not a good approximation to the objective function. On the other hand, the condition

$$\frac{\|s^{k,j}\|_\infty}{\max\left\{1, \|x^k\|_\infty\right\}} \leq \eta_2 \tag{7}$$

is in line with the classical step control that is generally used in Newtonian methods. As with the previous condition, imposing (7) prevents evaluating the objective function at points of the form $x^k + s^{k,j}$ that are far from x^k, at which the model may not be a good approximation to the objective function. Both conditions, that are empirical and have no relation with the theoretical results concerning Algorithm 2.1, aim to discard unuseful steps, without evaluating the objective function. The maximum number of times the step control may be performed at every iteration k is stated as J. Since $J \geq 0$ is an input parameter of the algorithm, there is an upper bound (that does not depend on the iteration) on the number of times the regularization parameter σ may be increased without evaluating the objective function. The practical impact of the step control will be analyzed in the numerical experiments.

3 Well-definiteness and complexity results

Assumption A1 For all $k \geq 0$ and $j \geq 0$, there exist nonnegative scalars ξ_1 and ξ_2 such that the iterates x^k and the trial steps $s^{k,j}$ satisfy

$$f(x^k + s^{k,j}) - T_p(x^k, s^{k,j}) \leq \xi_1 \|s^{k,j}\|^{p+1} \tag{8}$$

and

$$\|\nabla f(x^k + s^{k,j}) - \nabla_s T_p(x^k, s^{k,j})\| \leq \xi_2 \|s^{k,j}\|^p. \tag{9}$$

Assumption A1 is fullfiled when f is $p+1$ times continuously differentiable on $\mathbb{R}^n$ and the $(p+1)$-th derivative of f is bounded; or when f is p times continuously differentiable and the p-th derivative is Lipschitz continuous (see, for example, [4]).

Assumption A2 For all $k \geq 0$, if $j = 0$ and $s^{k,0}$ satisfying (3) and (4) does not exist, this fact can be detected within a finite number of operations; and, for all $j > 0$, $s^{k,j}$ satisfying (3) and (4) can be computed within a finite number of operations.

Assumption A2 is reasonable in the light of a practical unconstrained minimization algorithm. As previously explained, in case the model $m_p(x^k, s, \sigma_{k,j})$ has bounded level sets, it is possible to find $s^{k,j}$ satisfying (3) and (4) within a finite number of operations. Otherwise, the practical algorithm will detect the unboundedness of the model by stopping when the model reaches a value smaller than a pre-established threshold.

The next result is technical and it will be used in Lemmas 2 and 3 .

Lemma 1 *Suppose that Assumptions A1 and A2 hold for Algorithm 2.1. Then*

$$\sigma_{k,j} \geq (p+1)(\xi_1 + \alpha) \tag{10}$$

implies that $s^{k,j}$ exists and it satisfies (6).

Proof The fact that $\sigma_{k,j} > 0$ means that $j > 0$ and, therefore, by Assumption A2, $s^{k,j}$ exists. Then, by (8) (in Assumption A1) and (10),

$$\begin{aligned}
f(x^k + s^{k,j}) &\leq T_p(x^k, s^{k,j}) + \xi_1 \|s^{k,j}\|^{p+1} \\
&= T_p(x^k, s^{k,j}) + \frac{\sigma_{k,j}}{p+1}\|s^{k,j}\|^{p+1} - \frac{\sigma_{k,j}}{p+1}\|s^{k,j}\|^{p+1} + \xi_1 \|s^{k,j}\|^{p+1} \\
&= m_p(x^k, s^{k,j}, \sigma_{k,j}) - \left(\frac{\sigma_{k,j}}{p+1} - \xi_1\right)\|s^{k,j}\|^{p+1} \\
&\leq m_p(x^k, 0, \sigma_{k,j}) - \alpha \|s^{k,j}\|^{p+1} \\
&= f(x^k) - \alpha \|s^{k,j}\|^{p+1}
\end{aligned}$$

as we wanted to prove. □

The lemma below shows that Algorithm 2.1 is well defined.

Lemma 2 *Suppose that Assumptions A1 and A2 hold for Algorithm 2.1. Then, for all $k \geq 0$, s^k is well defined.*

Proof To show that s^k is well defined means to show that s^k can be computed within a finite number of operations. By the definition of the algorithm, for all $k \geq 0$ and $j \geq 1$, $\sigma_{k,j} = \gamma^{j-1}\sigma_{\text{low}}$. Since $\gamma > 1$, this means that, for all $k \geq 0$, there exists a finite value $j_k \geq \max\{1, J\}$ such that (10) holds with $j = j_k$. By Assumption A2,

s^{k,j_k} satisfying (3), (4), and (5) can be computed within a finite number of operations and, by Lemma 1, s^{k,j_k} satisfies (6). This means that $s^k = s^{k,j}$, for some $j \leq j_k$, can be computed within a finite number of operations, concluding the proof. □

The next result is auxiliary to the complexity analysis of Algorithm 2.1.

Lemma 3 *Suppose that Assumptions A1 and A2 hold for Algorithm 2.1. Then, for all* $k \geq 0$,

$$\sigma_k \leq \sigma_{\max} \stackrel{\text{def}}{=} \max\left\{\gamma^{\max\{0,J-1\}}\sigma_{\text{low}}, (p+1)(\xi_1+\alpha)\right\} \tag{11}$$

and

$$\|s^k\| \geq \left(\frac{\|\nabla f(x^k+s^k)\|}{\xi_2+\theta+\sigma_{\max}}\right)^{1/p}. \tag{12}$$

Proof On the one hand, if $\sigma_{k,j} \geq \gamma^{\max\{0,J-1\}}\sigma_{\text{low}}$ then $j \geq \max\{0, J-1\}+1 = \max\{1, J\}$ and, therefore, (5) trivially holds. In this case $s^{k,j}$ exists by Assumption A2. On the other hand, if $\sigma_{k,j}$ satisfies (10) then, $s^{k,j}$ exists by Assumption A2 and, by Lemma 1, it satisfies (3), (4), and (6). Thus, every iteration k finishes with some $\sigma_{k,j}$ satisfying

$$\sigma_{k,j} \leq \max\left\{\gamma^{\max\{0,J-1\}}\sigma_{\text{low}}, (p+1)(\xi_1+\alpha)\right\}$$

from which (11) follows.

Let us write

$$\|\nabla f(x^k+s^k)\| = \|\nabla f(x^k+s^k) - \nabla_s m_p(x^k,s^k,\sigma_k) + \nabla_s m_p(x^k,s^k,\sigma_k)\|. \tag{13}$$

Since

$$\nabla_s m_p(x,s,\sigma) = \nabla_s T_p(x,s) + \sigma\|s\|^p \frac{s}{\|s\|}, \tag{14}$$

by substituting (14) in (13), the triangle inequality, (9), (4), and (11), we have that

$$\begin{aligned}\|\nabla f(x^k+s^k)\| &\leq \|\nabla f(x^k+s^k) - \nabla_s T_p(x^k,s^k)\| + \|\nabla_s m_p(x^k,s^k,\sigma_k)\| + \sigma_k\|s^k\|^p \\ &\leq (\xi_2+\theta+\sigma_{\max})\,\|s^k\|^p,\end{aligned}$$

from which (12) follows. □

Theorem 1 *Suppose that Assumptions A1 and A2 hold for Algorithm 2.1, and that* $f(x) \geq f_{\text{low}}$ *for all* $x \in \mathbb{R}^n$. *Then, given* $\epsilon > 0$, *Algorithm 2.1 performs, at most,*

$$\left\lfloor \left(\frac{(\xi_2+\theta+\sigma_{\max})^{\frac{p+1}{p}}}{\alpha}\right)\left(\frac{f(x^0)-f_{\text{low}}}{\epsilon^{\frac{p+1}{p}}}\right)\right\rfloor + 1 \tag{15}$$

iterations and at most

$$\left\lfloor \log_\gamma\left(\frac{\sigma_{\max}}{\sigma_{\text{low}}}\right)\right\rfloor + 1 \tag{16}$$

functional evaluations per iteration to produce an iterate x^{k+1} *such that* $\|\nabla f(x^{k+1})\| \leq \epsilon$.

Proof At every iteration k, Algorithm 2.1 computes an iterate $x^{k+1} = x^k + s^k$ such that s^k satisfies

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

and (12). Therefore, if $\|\nabla f(x^k + s^k)\| > \epsilon$ then iteration k produces a decrease of at least

$$f_{\text{decr}} \stackrel{\text{def}}{=} \alpha \left(\frac{\epsilon}{\xi_2 + \theta + \sigma_{\max}} \right)^{\frac{p+1}{p}}.$$

This decrease may occur at most $\lfloor (f(x^0) - f_{\text{low}})/f_{\text{decr}} \rfloor$ times. Therefore, for some $k \leq \lfloor (f(x^0) - f_{\text{low}})/f_{\text{decr}} \rfloor + 1$, we must have $\|\nabla f(x^k + s^k)\| \leq \epsilon$, from which (15) follows. By the definition of the algorithm and Lemma 3, we have that, for all $k \geq 0$ and all $j \geq 1$, $\gamma^{j-1}\sigma_{\text{low}} = \sigma_{k,j} \leq \sigma_{\max}$, from which (16) follows. □

4 A different rule for updating the regularization parameter

At the k-th iteration of Algorithm 2.1, the null regularization parameter $\sigma_{k,0} = 0$ is first considered. If it is not possible to compute $s^{k,0}$, or if $s^{k,0}$ is not accepted, the subsequent regularization parameters $\sigma_{k,j}$ are of the form $\gamma^{j-1}\sigma_{\text{low}}$, with $\gamma > 1$ and $\sigma_{\text{low}} > 0$ for $j = 1, 2, \ldots$. This means that σ_{low} is a lower bound for the non-null regularization parameters considered in Algorithm 2.1 at every iteration k. In the present section, we introduce another algorithm that differs from Algorithm 2.1 in the updating rules for the regularization parameter. The new updating rules are such that, on the one hand, the null regularization parameter still is the first trial at each iteration. On the other hand, there is no lower bound for the non-null subsequent regularization parameters. In fact, there is a lower bound σ_k^{ini} for the non-null regularization parameters at iteration k, but it may be the case that $\sigma_k^{\text{ini}} \to -\infty$ as $k \to \infty$.

Algorithm 4.1: $\text{AR}^{\text{UN}}_{p+1}$—Adaptive regularization of order $p+1$

Input. Let $x^0 \in \mathbb{R}^n$, $p \in \{1, 2, \ldots\}$, $\alpha > 0$, $\eta_1, \eta_2 > 0$, $\sigma_{\text{low}} > 0$, $\theta > 0$, $J \in \mathbb{N}$, and $0 < \gamma_1 < 1 < \gamma_2$ be given. Define $\sigma_0^{\text{ini}} = \sigma_{\text{low}}$ and initialize $k \leftarrow 0$.

Step 1. Define $\sigma_{k,0} = 0$ and initialize $j \leftarrow 0$.

Step 2. Compute, if possible, $s^{k,j}$ such that

$$m_p(x^k, s^{k,j}, \sigma_{k,j}) \leq m_p(x^k, 0, \sigma_{k,j}) \tag{17}$$

and

$$\|\nabla_s m_p(x^k, s^{k,j}, \sigma_{k,j})\| \leq \theta \|s^{k,j}\|^p. \tag{18}$$

If $j = 0$ and finding $s^{k,j}$ satisfying (17) and (18) was not possible, define $\sigma_{k,1} = \sigma_k^{\text{ini}}$, set $j \leftarrow 1$, and repeat Step 2.

Step 3. If

$$j \geq J \quad \text{or} \quad \left(\frac{T_p(x^k, 0) - T_p(x^k, s^{k,j})}{\max\left\{1, |T_p(x^k, 0)|\right\}} \leq \eta_1 \quad \text{and} \quad \frac{\|s^{k,j}\|_\infty}{\max\left\{1, \|x^k\|_\infty\right\}} \leq \eta_2 \right), \tag{19}$$

go to Step 4. Otherwise, define $\sigma_{k,j+1} = \max\{\sigma_k^{\text{ini}}, \gamma_2 \sigma_{k,j}\}$, set $j \leftarrow j + 1$, and go to Step 2.

Step 4. If

$$f(x^k + s^{k,j}) \leq f(x^k) - \alpha \|s^{k,j}\|^{p+1}, \tag{20}$$

go to Step 5. Otherwise, define $\sigma_{k,j+1} = \max\{\sigma_k^{\text{ini}}, \gamma_2 \sigma_{k,j}\}$, set $j \leftarrow j + 1$, and go to Step 2.

Step 5. Define $\sigma_k = \sigma_{k,j}$, $s^k = s^{k,j}$, $x^{k+1} = x^k + s^k$, and

$$\sigma_{k+1}^{\text{ini}} = \begin{cases} \gamma_1 \sigma_k^{\text{ini}}, & \text{if } \sigma_k = 0, \\ \gamma_1 \sigma_k, & \text{otherwise,} \end{cases}$$

set $k \leftarrow k + 1$, and go to Step 1.

The well definiteness and the complexity results for Algorithm 4.1 are given below, following similar arguments to the ones used for Algorithm 2.1.

Lemma 4 *Suppose that Assumptions A1 and A2 hold for Algorithm 4.1. Then*

$$\sigma_{k,j} \geq (p + 1)(\xi_1 + \alpha) \tag{21}$$

implies that $s^{k,j}$ exists and it satisfies (20).

Proof The proof follows exactly as the proof of Lemma 1. □

Lemma 5 *Suppose that Assumptions A1 and A2 hold for Algorithm 4.1. Then, for all $k \geq 0$, s^k is well defined.*

Proof By the definition of the algorithm, for all $k \geq 0$ and $j \geq 1$, $\sigma_{k,j} \geq \gamma_2^{j-1} \sigma_k^{\text{ini}} \geq \gamma_2^{j-1} \gamma_1^k \sigma_0^{\text{ini}} = \gamma_2^{j-1} \gamma_1^k \sigma_{\text{low}}$. Since $\gamma_2 > 1$, this means that, for all $k \geq 0$, there exists a finite value $j_k \geq \max\{1, J\}$ such that (21) holds with $j = j_k$. By Assumption A2, s^{k,j_k} satisfying (17), (18), and (19) can be computed in finite time and, by Lemma 4, s^{k,j_k} satisfies (20). This means that $s^k = s^{k,j}$ for some $j \leq j_k$ can be computed within a finite number of operations, as we wanted to prove. □

Lemma 6 *Suppose that Assumptions A1 and A2 hold for Algorithm 4.1. Then, for all $k \geq 0$,*

$$\sigma_k \leq \hat{\sigma}_{\max} \stackrel{\text{def}}{=} \gamma_2^{\max\{0, J-1\}} \max\{\sigma_{\text{low}}, (p + 1)(\xi_1 + \alpha)\} \tag{22}$$

and

$$\|s^k\| \geq \left(\frac{\|\nabla f(x^k + s^k)\|}{\xi_2 + \theta + \hat{\sigma}_{\max}} \right)^{1/p}. \tag{23}$$

Proof If $\sigma_{k,j}$ satisfies (21) then $j > 0$ and, by Assumption A2, $s^{k,j}$ satisfying (17) and (18) exists. Moreover, by Lemma 4, $s^{k,j}$ also satisfies (20). Note that assuming that $\sigma_{k,j}$ satisfies (21) implies $j \geq 1$ and that the satisfaction of (19) may require, in the worst-case, $j \geq J$, that would imply in $J-1$ additional executions of Step 3, i.e. increasing the value of the regularization parameter by a factor of $\gamma_2^{\max\{0,J-1\}}$, from which (22) follows noting that the max with σ_{low} comes from the definition of the algorithm. The upper bound (23) on $\|s^k\|$ follows exactly as in Lemma 3 substituting $\sigma_{\max}$ by $\hat{\sigma}_{\max}$. □

Theorem 2 *Suppose that Assumptions A1 and A2 hold for Algorithm 4.1, and that $f(x) \geq f_{\text{low}}$ for all $x \in \mathbb{R}^n$. Then, given $\epsilon > 0$, Algorithm 4.1 performs, at most,*

$$k_{\text{up}} = \left\lfloor \left(\frac{(\xi_2 + \theta + \hat{\sigma}_{\max})^{\frac{p+1}{p}}}{\alpha} \right) \left(\frac{f(x^0) - f_{\text{low}}}{\epsilon^{\frac{p+1}{p}}} \right) \right\rfloor + 1 \tag{24}$$

iterations and at most

$$2(k_{\text{up}} + 1) + \left\lfloor \log_{\gamma_2} \left(\frac{\hat{\sigma}_{\max}}{\sigma_{\text{low}}} \right) + k_{\text{up}} \left| \log_{\gamma_2}(\gamma_1) \right| \right\rfloor \tag{25}$$

functional evaluations to produce an iterate x^{k+1} such that $\|\nabla f(x^{k+1})\| \leq \epsilon$.

Proof The bound (24) on the number of iterations follows exactly as in Theorem 1. For all $k \geq 0$, let j_k be such that $\sigma_k = \sigma_{k,j_k}$. By the definition of the algorithm, for all $k \geq 0$, we have that (a) if $j_k = 0$ then $\sigma_{k+1}^{\text{ini}} = \gamma_1 \sigma_k^{\text{ini}}$, and (b) if $j_k > 0$ then $\sigma_{k+1}^{\text{ini}} = \gamma_1 \sigma_{k,j_k} = \gamma_1 \gamma_2^{j_k - 1} \sigma_k^{\text{ini}}$. Therefore, for all $k \geq 0$, we obtain $\sigma_{k+1}^{\text{ini}} = \gamma_1 \gamma_2^{\max\{0, j_k - 1\}} \sigma_k^{\text{ini}}$. Thus, for all $k \geq 0$, an inductive argument yields

$$\sigma_{k+1}^{\text{ini}} = \gamma_1^{k+1} \gamma_2^{\left(\sum_{\ell=0}^{k} \max\{0, j_\ell - 1\} \right)} \sigma_0^{\text{ini}}.$$

Since, on the other hand, by the definition of the algorithm, for all $k \geq 0$, $\sigma_{k+1}^{\text{ini}} \leq \gamma_1 \hat{\sigma}_{\max}$, it follows that, for all $k \geq 0$,

$$\sum_{\ell=0}^{k} \max\{0, j_\ell - 1\} \leq \left\lfloor \log_{\gamma_2} \left(\frac{\hat{\sigma}_{\max}}{\sigma_{\text{low}}} \right) + k \left| \log_{\gamma_2}(\gamma_1) \right| \right\rfloor. \tag{26}$$

By the definition of the algorithm, the number of functional evaluations at every iteration k is $j_k + 1$. Then, the total number of functional evaluations up to iteration k is given by $\sum_{\ell=0}^{k} (j_\ell + 1) = \sum_{\ell=0}^{k} (j_\ell - 1 + 2) \leq \sum_{\ell=0}^{k} (\max\{0, j_\ell - 1\} + 2) = 2(k+1) + \sum_{\ell=0}^{k} \max\{0, j_\ell - 1\}$. Therefore, by (26), the total number of functional evaluations up to iteration k is at most

$$2(k+1) + \left\lfloor \log_{\gamma_2} \left(\frac{\hat{\sigma}_{\max}}{\sigma_{\text{low}}} \right) + k \left| \log_{\gamma_2}(\gamma_1) \right| \right\rfloor,$$

from which (25) follows replacing k by k_{up}. □

5 Numerical experiments

We implemented Algorithms 2.1 and 4.1 with $p = 2$ and $p = 3$ in Fortran 2008. Numerical experiments were run in a computer with 3.4 GHz Intel® Core™ i5 processor and 8 GB 1600 MHz DDR3 RAM memory, running macOS Sierra (version 10.12.5). Codes were compiled using Gfortran compiler of GCC (version 6.3.0) with the –O3 optimization directive enabled.

In Algorithms 2.1 and 4.1, we arbitrarily set $\alpha = 10^{-8}$, $\sigma_{\text{low}} = 10^{-8}$, $\theta = 100$, $\gamma_1 = 0.5$, and $\gamma = \gamma_2 = 10$. Since analyzing the influence of the step control strategy is one of the objectives of the numerical experiments, we considered $J \in \{0, 10, 20\}$, $\eta_1 \in \{10, 10^2, 10^3, 10^4\}$, and $\eta_2 \in \{2, 3, 5, 10\}$. As a stopping criterion for both algorithms, we considered

$$\|\nabla f(x^k)\|_\infty \leq \epsilon,$$

with $\epsilon = 10^{-8}$. Algorithms also stop if $k > k_{\max} = 1{,}000$. In both algorithms, models are minimized using Gencan [2,12,13,16]. All codes, as well as the full tables with details of all the numerical results, are available at http://www.ime.usp.br/~egbirgin/.

5.1 Generalized performance profiles

Let $M_1, \ldots, M_m$ be m optimization methods whose performances on a set of q problems $P_1, \ldots, P_q$ need to be evaluated. Depending on the focus of the evaluation, the functional value obtained by each method may be used in the comparison or not. Although some authors believe that the capacity of finding stationary points is the main goal to be evaluated, adopting the point of view discussed in detail in [11] and [16, p. 196], we consider in the present work that the functional value must be taken into account in the comparison. Assume that when method M_i is applied to problem P_j it generates a sequence of iterates $\{x_{ij}^k\}_{k \in K_{ij}}$, with the corresponding index set K_{ij} ($i \in \{1, \ldots, m\}$ for the method and $j \in \{1, \ldots, q\}$ for the problem), and the associated quantities f_{ij}^k and t_{ij}^k, where $f_{ij}^k = f_j(x_i^k)$ is the value of the objective function f_j of problem P_j evaluated at the k-th iterate of method M_i and t_{ij}^k is a metric related to the (accumulated) cost of having obtained x_{ij}^k (t_{ij}^k may be CPU time, number of functional evaluations, or any other performance metric).

For a given problem P_j, let

$$f_{\text{best}}^j \stackrel{\text{def}}{=} \min_{i \in \{1,\ldots,m\}} \left\{ \min_{k \in K_{ij}} \left\{ f_{ij}^k \right\} \right\}. \tag{27}$$

In this context, f_{best}^j plays the role of an approximation to the global minimum of problem P_j, and the global minimum could be used in its place if it were known. Following [11], for a given tolerance $\varepsilon_f > 0$, we say that the method M_i *found an* ε_f*-approximation to a solution* of the problem P_j if there exists $k \in K_{ij}$ such that

$$\frac{f_{ij}^k - f_{\text{best}}^j}{\max\{1, |f_{\text{best}}^j|\}} \leq \varepsilon_f. \tag{28}$$

In order to cope with the case of problems whose objective functions are unbounded from below within the feasible region, we also say that a method M_i found a solution of the problem P_j if $f_{\text{best}}^j \leq f_{-\infty}$ with, let us say, $f_{-\infty} = -10^{10}$, and, for some $k \in K_{ij}$, $f_{ij}^k \leq f_{-\infty}$, suggesting that the objective function of problem P_j is unbounded from below within the feasible region, and that method M_i was able to detect this situation.

If a method M_i found an ε_f-approximation to a solution of the problem P_j, we say that the cost of finding such an approximate solution is given by

$$t_{ij}(\varepsilon_f) \stackrel{\text{def}}{=} \min_{k \in K_{ij}} \left\{ t_{ij}^k \mid f_{ij}^k \text{ satisfies (28)} \right\}. \tag{29}$$

Otherwise, we say that $t_{ij}(\varepsilon_f) = +\infty$. Note that this measure of the effort made by the method M_i to find an approximate solution (with precision ε_f) is different (and presumably much smaller than) the total effort required for the satisfaction of an underlying stopping criterion. Computing this measure of effort is only possible if we have access to the whole sequence generated by the method and not only to its final iterate or reported solution. Then, for a given problem P_j, and a given tolerance ε_f, let

$$t_{\text{min}}^j(\varepsilon_f) \stackrel{\text{def}}{=} \min_{i \in \{1,\ldots,m\}} \{t_{ij}(\varepsilon_f)\}. \tag{30}$$

Note that we may have $t_{\text{min}}^j = +\infty$ if $t_{ij}(\varepsilon_f) = +\infty$ for $i = 1, \ldots, m$. This will be the case if f_{best}^j assumes the value of the known global minimum of problem P_j, and none of the evaluated methods reach that value within the tolerance ε_f. On the other hand, if f_{best}^j is defined as in (27), we have that $t_{\text{min}}^j(\varepsilon_f) < +\infty$.

Performance Profiles (PP) [25] constitute a well-established tool to analyze the behavior of a set of methods when applied for solving a set of problems. In PP, the efficiency and the robustness of method M_i may be evaluated by analyzing the curve

$$\Gamma_i^{\varepsilon_f}(\tau) \stackrel{\text{def}}{=} \frac{\#\left\{ j \in \{1, \ldots, q\} \mid t_{ij}(\varepsilon_f) \leq \tau t_{\text{min}}^j(\varepsilon_f) < +\infty \right\}}{q}, \tag{31}$$

where $\#\mathcal{S}$ denotes the cardinality of the set $\mathcal{S}$. The value $\Gamma_i^{\varepsilon_f}(1)$ is the proportion of problems in which the method M_i was the most efficient in finding an ε_f-approximation to a solution, whereas $\Gamma_i^{\varepsilon_f}(\tau)$, for $\tau > 1$, is the proportion of problems in which the method M_i was able to find an ε_f-approximation to a solution with a cost up to τ times the cost demanded by the most efficient method. The value $\Gamma_i^{\varepsilon_f}(+\infty)$ is the proportion of problems for which the method M_i was able to find an ε_f-approximation to a solution, independently of the required effort, being thus a measure of the robustness of the method M_i.

In [11], it was pointed out that the analysis described above strongly depends on the choice of the tolerance ε_f used to determine whether a method found an ε_f-

approximation to a solution of a given problem or not. Moreover, this dependency may impair the comparison if first- and second-order methods are being simultaneously compared, or if the methods being evaluated have different stopping criteria. Based on these observations, *Relative Minimization Profiles* (RMP) were introduced in [23,31]. In PP, the constant ε_f is fixed. Similarly, in RMP, the constant τ, related to the tolerance with respect to the cost of the most efficient method, is fixed. Then, for a fixed $\tau \geq 1$, the RMP approach assesses the performance of the method M_i by the curve

$$\Gamma_i^{\tau}(\varepsilon_f) \stackrel{\text{def}}{=} \frac{\#\left\{j \in \{1,\ldots,q\} \mid t_{ij}(\varepsilon_f) \leq \tau t_{\min}^{j}(\varepsilon_f) < +\infty\right\}}{q}. \tag{32}$$

The value $\Gamma_i^{\tau}(\varepsilon_f)$ represents the fraction of problems for which the method M_i found an ε_f-approximate solution with a cost not greater than τ times the cost of the most efficient method that also found an ε_f-approximate solution.

Comparing (32) with (31), it is easy to see that both curves are cross-sections of the two-dimensional surface

$$\Gamma_i(\tau, \varepsilon_f) \stackrel{\text{def}}{=} \frac{\#\left\{j \in \{1,\ldots,q\} \mid t_{ij}(\varepsilon_f) \leq \tau t_{\min}^{j}(\varepsilon_f) < +\infty\right\}}{q}, \tag{33}$$

that we name *Generalized Performance Profiles* (GPP). Similarly to the analysis in [11], in which several curves (31) varying ε_f are considered, in [23] it is suggested to consider varying τ (named β in [23]), constituting what the authors called "β-RMP benchmark panel". Thus, the surface (33) may be seen as an extension of the proposals in [11] and [23].

Clearly, the same approach may be developed for a comparison that focuses on the capacity of finding approximate stationary points with precision ε_g. Moreover, the approach applies to unconstrained minimization, and to convex constrained minimization, if the methods being assessed preserve the feasibility of the iterates. In the more general nonlinear programming setting, feasibility may be handled by defining a problem to be successfully solved only if feasibility is attained up to a given tolerance, as it is already done in RMP and in PP (see, for example, [1,5–7,14,15]). Another possibility would be introducing a third dimension to deal with the feasibility tolerance.

It should be noted that the curves $\Gamma_i^{\varepsilon_f}(\tau)$ are non-decreasing as a function of τ. The $\Gamma_i^{\tau}(\varepsilon_f)$ curves, on the other hand, due to the definition of $t_{\min}^{j}(\varepsilon_f)$ in (30), do not necessarily possess a monotone pattern.

5.2 Numerical experiments with Moré, Garbow, and Hillstrom's test set

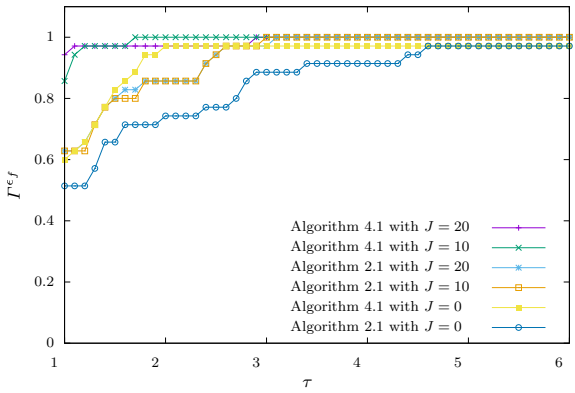
In the numerical experiments of the present section, we considered all the 35 unconstrained minimization problems of Moré, Garbow, and Hillstrom's test set [32]. In [32], only first-order derivatives were made available. Second-order derivatives were provided in [3]. To apply Algorithms 2.1 and 4.1 with $p = 3$, we computed third-

order derivatives of all the 35 problems of the test set. The implementation details are described in [8].

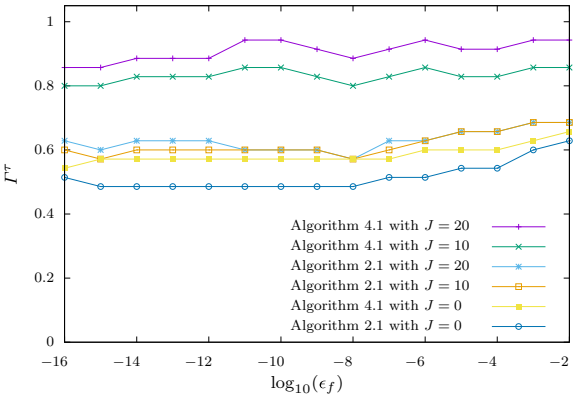
In a first experiment, we aim to evaluate the influence of the step control strategy, and of the updating rule for the regularization parameter in the behavior of the algorithms. With this purpose, we compare the behavior of Algorithms 2.1 and 4.1 with $p \in \{2, 3\}$ and $J \in \{0, 10, 20\}$. Since numerical experiments showed that, when $J > 0$, the performance of the algorithms is *not* very sensitive to variations of $\eta_1 \in \{10, 10^2, 10^3, 10^4\}$ and $\eta_2 \in \{2, 3, 5, 10\}$, only numerical results with $\eta_1 = 10^3$ and $\eta_2 = 3$ will be reported. Figures 1 and 2 present a comparison using generalized performance profiles for the cases $p = 2$ and $p = 3$, respectively, using the number of functional evaluations as performance metric. Figure 1a corresponds to performance profiles with $\varepsilon_f = 10^{-6}$ for the case $p = 2$. Its left-hand side shows that, for $p = 2$, Algorithm 4.1 is more efficient than Algorithm 2.1 and that variants with $J \in \{10, 20\}$ are more efficient than variants with $J = 0$. The legend obeys the efficiency order, from the most to the least. The right-hand side of the figure shows that there are no meaningful differences in the robustness of the variants (the existing small difference corresponds to a single problem). The "almost constant" curves in Fig. 1b, that never cross each other, show that the efficiency of the methods (displayed on the left-hand side of Fig. 1a) *do not* depend on the arbitrary choice $\varepsilon_f = 10^{-6}$, whereas the "almost constant" curves in Fig. 1c show that the robustness of the methods (displayed on the right-hand side of Fig. 1a) also *do not* depend on the arbitrary choice $\varepsilon_f = 10^{-6}$. Figures 2a–c show that similar results also hold for the case $p = 3$. The apparent lack of robustness of the variant of Algorithm 2.1 with $p = 3$ and $J = 0$ in Fig. 2a is, in fact, lack of efficiency. (Note that the abscissa stops at $\tau = 6$.) It should be noted that the efficiency of Algorithm 4.1, when compared against Algorithm 2.1, does not depend on the lack of a lower bound for the non-null regularization parameters, but relies on the fact that each iteration starts with a regularization parameter close to the successful one from the previous iteration, promoting a reduction in the number of functional evaluations.

A note on the usefulness of the curves $\Gamma^\tau(\varepsilon_f)$ displayed in Figs. 1b–c and 2b–c is in order. Figures 1b and 2b correspond to $\tau = 1$ and, thus, display the variation of the *efficiency* of the methods (i.e. $\Gamma^{\varepsilon_f}(1)$) as a function of ε_f. Figs. 1c and 2c correspond to $\tau = \infty$, displaying the variation of the *robustness* of the methods (i.e. $\Gamma^{\varepsilon_f}(\infty)$) as a function of ε_f. The four graphics show "almost constant" curves that practically never cross each other. As a result, by ranking the methods by their efficiency or their robustness, their relative positions within the ranking would not depend on the choice of ε_f. This is because the methods being compared here are in fact variations of the same method, they all use second-order information, and they all share the same stopping criterion. The situation might be different if methods of different nature were being compared, as, for example, methods that use first- and second-order information, or methods that consider different stopping criteria. See, for example, [11, p. 360, Fig. 5a–b] and [23, p. 26, Fig. 6].

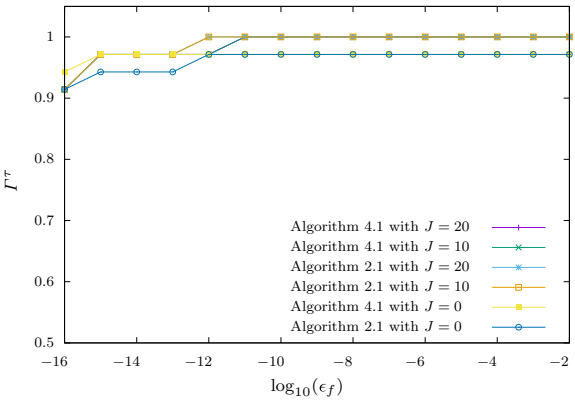
We now turn our attention to the question on whether using a third-order Taylor polynomial model with a fourth-order regularization term is more efficient than using a second-order model with a third-order regularization term. With this purpose, we compare the performance of Algorithm 4.1 with $J = 20$ and $p \in \{2, 3\}$. Fig-



(a) $\epsilon_f = 10^{-6}$

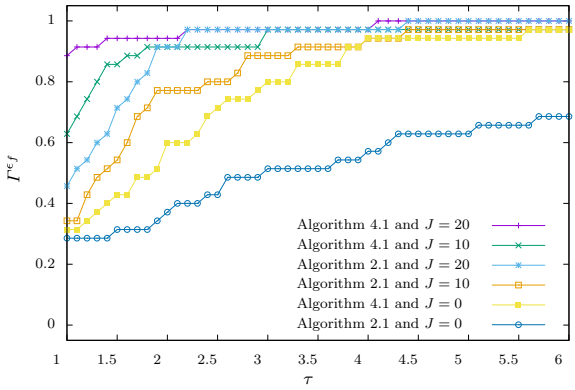


(b) $\tau = 1$

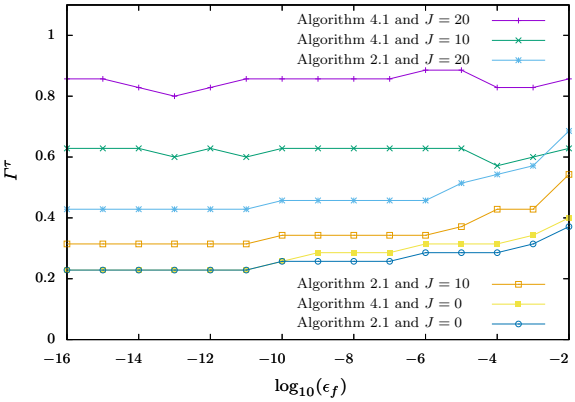


(c) $\tau = \infty$

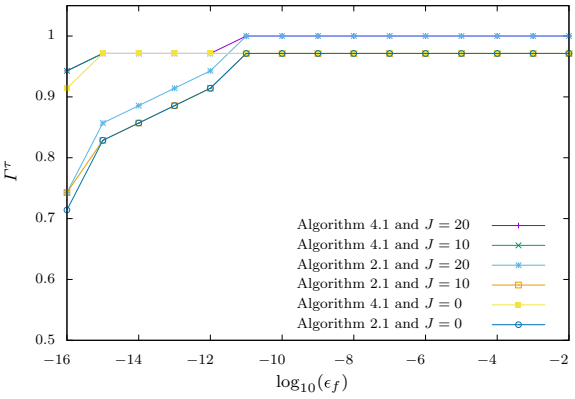
Fig. 1 **a**, **b** and **c** Cross-sections of the generalized performance profiles for Algorithms 2.1 and 4.1 with $p = 2$. The legend obeys the efficiency order, from the most to the least. (In **b** the curves of Algorithm 2.1 with $J = 10$ and $J = 20$ almost coincide. In **c** the curves of Algorithm 2.1 with $J = 0$ and $J = 10$ are identical, as well as the curves of Algorithm 4.1 with $J = 10$ and $J = 20$)



(a) $\epsilon_f = 10^{-6}$



(b) $\tau = 1$

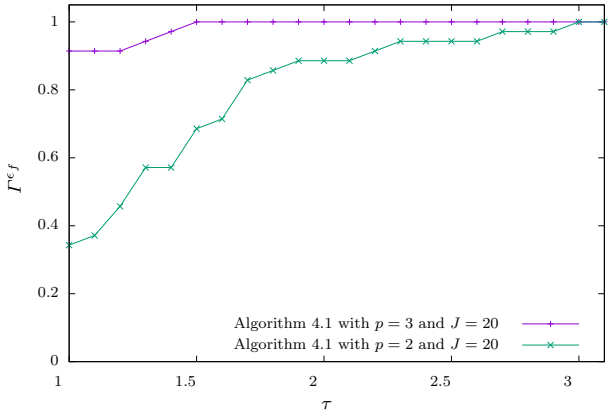


(c) $\tau = \infty$

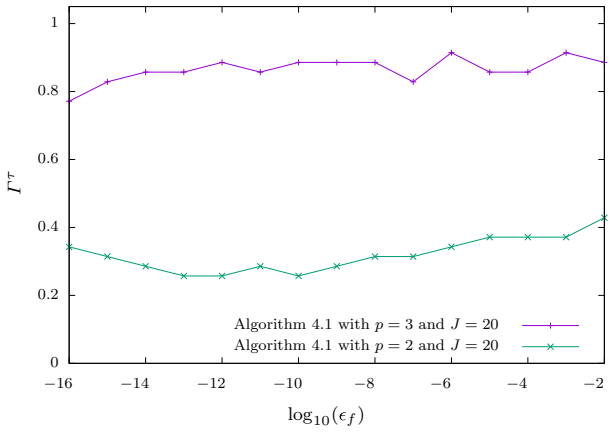
Fig. 2 **a**, **b** and **c** Cross-sections of the generalized performance profiles for Algorithms 2.1 and 4.1 with $p = 3$. The legend obeys the efficiency order, from the most to the least. (In **b** the curves of Algorithm 2.1 with $J = 0$ and Algorithm 4.1 with $J = 0$ are identical, as well as the curves of Algorithm 2.1 with $J = 0$ and $J = 10$ and the curves of Algorithm 4.1 with $J = 0$ and $J = 10$)

ure 3 depicts a comparison using generalized performance profiles with the number of functional evaluations as performance metric, whereas Table 1 presents the detailed results. Figure 3a shows that both variants are equally robust, and that when third-order derivatives are used, the method is much more efficient. Figure 3a shows that Algorithm 4.1 with $p = 3$ is more efficient than Algorithm 4.1 with $p = 2$ because the former uses fewer functional evaluations in approximately 91% of the problems, whereas the latter uses fewer functional evaluations in around 34% of the problems. (There are ties in approximately 25% of the problems.) On the other hand, in Fig. 3a, at $\tau \approx 3$, both curves collapse. Roughly speaking, this means that the variant with $p = 2$ never uses more than three times the number of functional evaluations required by the variant with $p = 3$. Note that these figures refer to the effort demanded by the methods to find a solution with precision $\varepsilon_f = 10^{-6}$, while the figures in Table 1 refer to the effort demanded to satisfy a stopping criteria. Figures 3b, c show, respectively, that the efficiency and the robustness observed in Fig. 3a do not depend on the arbitrary choice of $\varepsilon_f = 10^{-6}$.

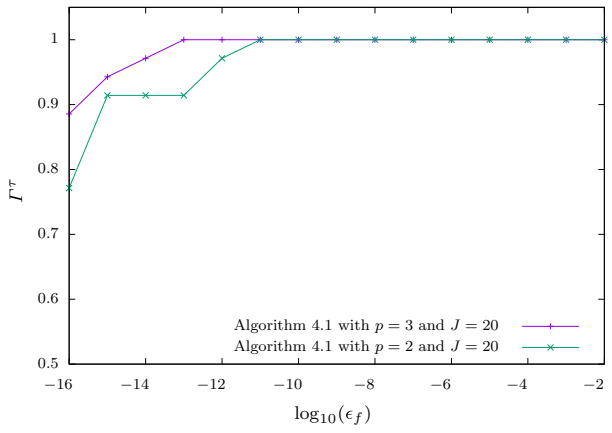
In Table 1, the first four columns identify a problem from the Moré, Garbow, and Hillstrom's test set [32], n being the number of variables of a function of the form $f(x) = \sum_{i=1}^{\widehat{m}} f_i(x)^2$. In the remaining columns of the table, $f(x^*)$ denotes the functional value at the final iterate; $\|\nabla f(x^*)\|$ denotes the sup-norm of the gradient at the final iterate; #it and #f denote, respectively, the demanded number of iterations and of functional evaluations; and "Time" is the CPU time in seconds. From the figures in the table, the spreading values (min, 25%, median, 75%, max) for the measure functional evaluations per iteration are (1.05, 1.13, 1.31, 1.59, 2.00) for $p = 2$, and (1.02, 1.2, 1.33, 1.65, 8.14) for $p = 3$. Consequently, the advantage of the case $p = 3$ in terms of number of functional evaluations, as observed in Fig. 3a–c, is due to a reduction in the number of iterations, being not associated with more effectiveness of the computed step. It is worth noting that, in the case $p = 2$, the amount of iterations with pure Newton steps (i.e. with $\sigma_k = 0$) within the total demanded, for the aforementioned five spreading values, is (0.18, 0.65, 0.90, 1.0, 1.0). In the case $p = 3$, we have used $\sigma_k = 0$ in fewer iterations, being the spreading values of the fraction of such an amount upon the total number of iterations given by (0.02, 0.31, 0.64, 0.93, 1.0). The differences may be related to the fact that, when $p = 2$, the exact solution of the quadratic model (with null regularization parameter) can be computed exactly, whereas, in the case $p = 3$, an approximate solution is computed iteratively. Moreover, in the latter case, there is no way of avoiding the iterative solver to diverge. In fact, in Table 1, the four results with a star in the column named $\|\nabla f(x^*)\|$ correspond to instances in which the iterative solver failed in computing $s^{k,j}$ satisfying (3)–(4) or (17)–(18). When the iterative solver is applied to the model minimization, the stopping criterion (4) or (18) is sought; but stopping achieving an imposed maximum number of iterations, or by lack of progress is also possible. In this cases, the algorithms proceed by increasing the regularization parameter as described at Step 2. There are three cases marked with stars corresponding to runs in which $\sigma_{k,j} \geq 10^{20}$ and the method was unable to compute a step $s^{k,j}$ satisfying (3)–(4) or (17)–(18), and a single case (problem BDF) that stopped due to lack of progress, as $\|s^{k,j}\|$ became too tiny.



(a) $\epsilon_f = 10^{-6}$



(b) $\tau = 1$



(c) $\tau = \infty$

Fig. 3 Cross-sections of the generalized performance profiles for Algorithm 4.1 with $p \in \{2, 3\}$ and $J = 20$. The legend obeys the efficiency order, from the most to the least

Table 1 Numerical results of the experiments with Algorithm 4.1 ($p \in \{2, 3\}$ and $J = 20$)

	Problem	n	$\widehat{m}$	Algorithm 4.1 ($p = 2$ and $J = 20$)					Algorithm 4.1 ($p = 3$ and $J = 20$)				
				$f(x^*)$	$\|\nabla f(x^*)\|$	$\#it$	$\#f$	Time	$f(x^*)$	$\|\nabla f(x^*)\|$	#it	#f	Time
1	ROS	2	2	1.281e−30	4.4e−14	22	32	0.08	2.619e−27	4.5e−13	13	18	0.07
2	FRF	2	2	4.898e+01	2.5e−13	7	8	0.04	4.898e+01	9.3e−13	5	6	0.03
3	PBS	2	2	0.000e+00	0.0e+00	101	159	0.32	2.415e−30	3.1e−15	53	63	0.32
4	BBS	2	3	0.000e+00	0.0e+00	44	46	0.17	0.000e+00	0.0e+00	56	57	0.18
5	BEA	2	3	1.080e−27	1.4e−13	14	15	0.03	3.944e−31	5.3e−15	8	9	0.04
6	JSF	2	10	1.243e+02	5.2e−12	10	11	0.03	1.243e+02	4.0e−12	6	7	0.06
7	HFV	3	3	2.183e−21	7.4e−10	12	16	0.06	1.409e−22	2.1e−10	7	8	0.05
8	BAR	3	15	8.214e−03	8.9e−09	10	12	0.04	8.214e−03	3.6e−12	7	8	0.04
9	GAU	3	15	1.127e−08	3.1e−09	2	3	0.01	1.127e−08	2.4e−16	2	3	0.00
10	MEY	3	16	8.794e+01	9.4e−05*	205	372	0.75	8.794e+01	3.0e−04*	257	339	3.11
11	GUL	3	10	5.546e−18	6.9e−09	34	50	0.14	8.690e−21	1.2e−09	20	28	0.08
12	BTD	3	10	6.524e−23	4.9e−12	26	28	0.07	3.081e−32	2.5e−16	15	18	0.05
13	PSF	4	4	1.283e−12	4.3e−09	21	22	0.04	5.199e−13	2.4e−09	14	19	0.09
14	WOD	4	6	6.309e−30	8.0e−14	41	62	0.15	6.437e−29	5.2e−14	38	50	0.20
15	KOF	4	11	3.075e−04	1.7e−10	11	14	0.05	3.075e−04	2.8e−12	10	12	0.05
16	BDF	4	20	8.582e+04	2.9e−10	8	9	0.02	8.582e+04	2.0e−04*	7	57	0.15
17	OS1	5	33	5.464e−05	2.9e−11	42	67	0.20	5.464e−05	2.5e−10	33	35	0.22
18	BIG	6	13	7.083e−15	6.8e−10	46	56	0.16	1.125e−14	1.7e−09	19	33	0.07
19	OS2	11	65	4.013e−02	2.1e−09	16	21	0.08	4.013e−02	4.8e−10	17	26	0.09
20	WAT	6	31	2.287e−03	6.3e−13	12	13	0.03	2.287e−03	3.8e−11	11	13	0.07

Table 1 continued

Problem		n	$\widehat{m}$	Algorithm 4.1 ($p = 2$ and $J = 20$)					Algorithm 4.1 ($p = 3$ and $J = 20$)				
				$f(x^*)$	$\|\nabla f(x^*)\|$	$\#it$	$\#f$	Time	$f(x^*)$	$\|\nabla f(x^*)\|$	#it	#f	Time
21	ERO	10	10	1.887e−24	1.7e−11	22	40	0.08	4.095e−20	1.6e−10	23	34	0.12
22	EPO	12	12	4.624e−12	4.4e−09	21	22	0.04	2.312e−12	4.5e−09	20	25	0.09
23	PE1	4	5	2.249e−05	1.3e−09	38	64	0.12	2.249e−05	1.0e−09	34	57	0.17
24	PE2	4	8	9.376e−06	5.1e−10	103	183	0.35	9.376e−06	8.2e−09	48	74	0.33
25	VDF	10	12	1.744e−26	2.6e−12	14	15	0.03	3.596e−18	3.7e−08*	9	13	0.13
26	TRI	10	10	2.795e−05	6.8e−09	16	29	0.05	2.795e−05	3.2e−13	7	12	0.06
27	BAL	40	40	5.286e−18	1.6e−09	5	6	0.02	4.198e−22	1.5e−11	4	7	0.28
28	DSB	10	10	1.857e−24	1.8e−13	3	4	0.01	7.999e−25	1.4e−12	3	4	0.01
29	DSI	10	10	2.034e−27	4.8e−14	5	6	0.02	5.788e−29	8.4e−15	3	4	0.01
30	BRT	10	10	2.001e−23	2.3e−11	6	7	0.02	6.177e−19	6.3e−09	4	5	0.01
31	BRB	10	10	2.523e−21	4.7e−10	8	9	0.02	1.046e−22	8.6e−11	5	6	0.05
32	LFF	10	10	0.000e+00	0.0e+00	1	2	0.01	0.000e+00	0.0e+00	1	2	0.01
33	LF1	10	10	2.142e+00	1.6e−11	1	2	0.01	2.142e+00	1.6e−11	1	2	0.01
34	LFZ	10	10	3.647e+00	5.5e−12	1	2	0.01	3.647e+00	7.4e−12	1	2	0.01
35	CHE	8	8	3.516e−03	9.4e−11	13	19	0.07	3.516e−03	1.5e−09	10	25	0.08

5.3 A large-scale problem

In this section, we consider the (feasibility) problem of packing N identical small spheres with radius $r > 0$ within a large sphere of radius $R \geq r$. By packing we mean that the small spheres must be placed within the large sphere without overlapping. (See, for example, [10,18,19].) Variables of the problem are the centers $c_1, \ldots, c_N \in \mathbb{R}^3$ of the small spheres. If we assume, without loss of generality, that the large sphere is centered at the origin of the Cartesian coordinate system, the problem can be modeled as

$$\underset{x \in \mathbb{R}^{3N}}{\text{Minimize}}\ f(x), \tag{34}$$

where $x = (c_1^T, \ldots, c_N^T)^T$ and

$$\begin{aligned} f(x) = & \underbrace{\sum_{i=1}^{N} \sum_{j=i+1}^{N} \max\left\{0, (2r)^2 - \|c_i - c_j\|_2^2\right\}^4}_{\text{overlapping between the small spheres}} \\ & + \underbrace{\sum_{i=1}^{N} \max\left\{0, \|c_i\|_2^2 - (R-r)^2\right\}^4}_{\text{fitting within the large sphere}}. \end{aligned} \tag{35}$$

A solution of the packing problem corresponds to a solution of problem (34) at which f vanishes.

At each iteration, Algorithms 2.1 and 4.1 with $p = 3$ compute an approximate minimizer of the regularized model m_3 defined in (2), whose Hessian is given by

$$\nabla_s^2 m_3(x, s, \sigma) = \nabla_s^2 T_3(x, s) + \nabla_s^2 \left[\frac{\sigma}{4}\|s\|^4\right].$$

Since a Newtonian method is used to approximately minimize model m_3, efficiently tackling large-scale instances of problem (34) depends on the sparsity of $\nabla_s^2 m_3$, for which necessary conditions are the sparsity of the second- and third-order derivatives of f, and sparsity of the regularized term of the Hessian. Although second- and third-order derivatives of f in (35) are structurally dense, they are numerically sparse and can be efficiently computed when the points c_i are "well distributed in the space" (see [19] for details). We now turn our attention to the sparsity of the regularized term of the Hessian or, in other words, to the choice of an adequate norm for the regularization term. If $\|\cdot\|$ represents the Euclidean norm then we have

$$\nabla_s^2 \left[\frac{\sigma}{4}\|s\|^4\right] = 2\sigma s s^T + \sigma \|s\|_2^2 I,$$

Table 2 Details of the numerical experiments with large-scale instances of the packing problem

N	R	Algorithm 4.1 with $p = 3$				Algencan			
		$f(x^*)$	#it	#f	Time	$f(x^*)$	#it	#f	Time
1,000	14.93	4.844e−13	46	76	18.26	4.820e−13	99	555	1.58
2,000	18.82	7.625e−12	71	128	82.80	2.559e−13	118	684	3.83
4,000	23.71	3.775e−12	83	144	149.94	8.556e−14	177	1126	6.99
6,000	27.14	1.431e−11	112	199	625.21	1.039e−13	196	1347	12.47
8,000	29.88	1.394e−11	118	207	792.38	1.889e−13	131	852	14.27
10,000	32.18	6.808e−11	156	282	1704.38	1.304e−13	246	1861	37.47

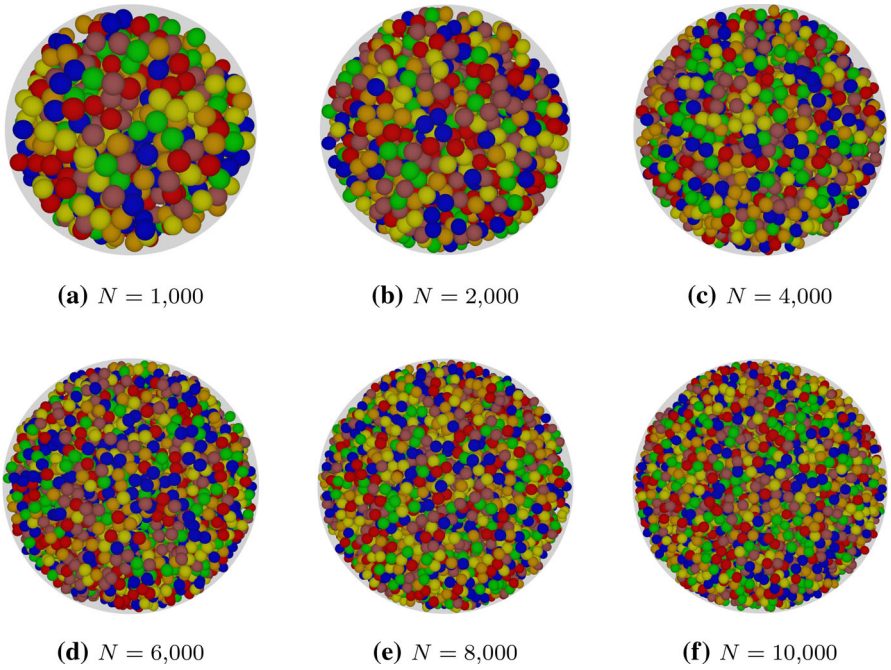
a potentially dense matrix, due to the term ss^T. On the other hand, if $\|s\| = \|s\|_4 = (\sum_{i=1}^n |s_i|^4)^{1/4}$ then

$$\nabla_s^2 \left[\frac{\sigma}{4}\|s\|^4\right] = 3\sigma \,\mathrm{diag}(s)^2,$$

where $\mathrm{diag}(s)$ is the diagonal matrix whose diagonal elements are $s_1, \ldots, s_n$. Thus, for the numerical experiments of the current section, we considered $\|\cdot\| = \|\cdot\|_4$ for the regularization term.

We remark that we have not run Algorithm 4.1 with $p = 2$, and the regularization term stated by the Euclidean norm for this class of problems, due to the structurally dense nature of the Hessian of the associated model obtained with these choices. Moreover, for $p = 2$, although setting the regularization term with the $\|\cdot\|_3$ would ensure sparsity for the model, the lack of differentiability becomes an issue. Therefore, we have just considered Algorithm 4.1 with $p = 3$ for the tests of the current section.

As pointed out in [16, §13.1], the kind of packing problem being considered is in the heart of the strategies of Packmol [30], a package for building initial configurations for molecular dynamic simulations. In this context, the radii r and R follow the relation $R \approx r\sqrt[3]{N/0.3}$, meaning that the small spheres occupy approximately 30% of the volume of the large sphere. Therefore, in the numerical experiments, we considered six instances with $N \in \{1{,}000; 2{,}000; 4{,}000; 6{,}000; 8{,}000; 10{,}000\}$ unitary-radius small spheres (i.e. $r = 1$) and R as specified in Table 2. We compared the performance of Algorithm 4.1 with $p = 3$ against Algencan [16]. In this case (unconstrained minimization), Algencan reduces to Gencan [2,12,13], that is the method used in the software Packmol. In the table, $f(x^*)$ denotes the functional value at the final iterate; #it and #f denote, respectively, the demanded number of iterations and of functional evaluations; and "Time" is the CPU time in seconds. The figures in the table show that both methods found equivalent solutions to the packing problem and that, although Algencan is faster than Algorithm 4.1 with $p = 3$, the later uses around 15% of the number of functional evaluations used by Algencan. Solutions are depicted in Fig. 4.



(a) $N = 1{,}000$ **(b)** $N = 2{,}000$ **(c)** $N = 4{,}000$

(d) $N = 6{,}000$ **(e)** $N = 8{,}000$ **(f)** $N = 10{,}000$

Fig. 4 Solutions of the packing problems with increasing dimensions

6 Conclusions

A practical algorithm that uses regularized third-order models for unconstrained optimization problems was introduced, implemented, and tested in this work. At variance with [9], the algorithm introduced here allows the use of zero regularization parameter, besides employing a step control strategy, and possessing the same worst-case complexity bounds. Numerical experiments with classical problems from the literature, and with an illustrative large-scale packing problem with a practical relevance in the context of molecular dynamics were given.

The numerical experiments showed that Algorithm 4.1 with $p = 3$ uses, in general, fewer *functional* evaluations than Algorithm 4.1 with $p = 2$, which may be considered a version of Newton's method with cubic regularization. The number of iterations employed by Algorithm 4.1 with $p = 3$ is also smaller, in general, than the one employed by Algorithm 4.1 with $p = 2$. However, it must be warned that, at each iteration, the algorithm with $p = 3$ evaluates third-order derivatives of the objective function, whereas in the case $p = 2$, one only evaluates first- and second-order derivatives. The difference in computer time between the two choices strongly depends on the form of the objective function. The consequence of these facts is that $p = 3$ should be preferable to $p = 2$, from the computer time point of view, only if the cost of functional evaluations strongly dominates both the cost of computing third-order derivatives, and the cost of solving the subproblems. Unfortunately, in our present implementation, the cost of solving the subproblems is very high, and dom-

inates the computer time. Indeed, profiling the CPU time in an additional run with problems from Table 1 having variable dimension, namely problems 21–31, setting $n = 500$, we observed that solving the subproblems demanded 98.19% and 91.82% of the total CPU time for $p = 2$ and $p = 3$, respectively. In this additional run, the computation of the third-order derivatives demanded 8.13%, amounting to 99.95% of the demanded CPU time.

The main conclusion of this work is that the use of $p = 3$ instead of the Newtonian choice $p = 2$ will be advantageous only if better methods for minimizing the regularized subproblems become available. The fact that the regularized subproblem with $p = 3$ is a quartic *polynomial* could represent an interesting feature but, up to now, we could not take advantage of such a structure.

Acknowledgements We are thankful to the anonymous reviewers, and to the Associate Editor, whose comments and suggestions improved the presentation of our work. This work has been partially supported by FAPESP (Grants 2013/03447-6, 2013/05475-7, 2013/07375-0, 2013/23494-9, 2016/01860-1, and 2017/03504-0) and CNPq (Grants 309517/2014-1, 303750/2014-6, and 302915/2016-8)

References

1. Andreani, R., Birgin, E.G., Martínez, J.M., Schuverdt, M.L.: Augmented Lagrangian methods under the constant positive linear dependence constraint qualification. Math. Program. **111**(1), 5–32 (2008)
2. Andretta, M., Birgin, E.G., Martínez, J.M.: Practical active-set Euclidian trust-region method with spectral projected gradients for bound-constrained minimization. Optimization **54**(3), 305–325 (2005)
3. Averbukh, V.Z., Figueroa, S., Schlick, T.: Remark on algorithm 566. ACM Trans. Math. Softw. **20**(3), 282–285 (1994)
4. Bertsekas, D.P.: Nonlinear Programming, 2nd edn. Athena Scientific, Belmont, Massachussets (1999)
5. Birgin, E.G., Bueno, L.F., Martínez, J.M.: Assessing the reliability of general-purpose inexact restoration methods. J. Comput. Appl. Math. **282**, 1–16 (2015)
6. Birgin, E.G., Bueno, L.F., Martínez, J.M.: Sequential equality-constrained optimization for nonlinear programming. Comput. Optim. Appl. **65**(3), 699–721 (2016)
7. Birgin, E.G., Castillo, R.A., Martínez, J.M.: Numerical comparison of augmented Lagrangian algorithms for nonconvex problems. Comput. Optim. Appl. **31**(1), 31–55 (2005)
8. Birgin, E.G., Gardenghi, J.L., Martínez, J.M., Santos, S.A.: Third-order derivatives of the Moré, Garbow, and Hillstrom test set problems. Technical Report MCDO010418, University of São Paulo (2018). http://www.ime.usp.br/~egbirgin/
9. Birgin, E.G., Gardenghi, J.L., Martínez, J.M., Santos, S.A., Toint, P.L.: Worst-case evaluation complexity for unconstrained nonlinear optimization using high-order regularized models. Math. Program. **163**(1), 359–368 (2017)
10. Birgin, E.G., Gentil, J.M.: New and improved results for packing identical unitary radius circles within triangles, rectangles and strips. Comput. Oper. Res. **37**(7), 1318–1327 (2010)
11. Birgin, E.G., Gentil, J.M.: Evaluating bound-constrained minimization software. Comput. Optim. Appl. **53**(2), 347–373 (2012)
12. Birgin, E.G., Martínez, J.M.: A box-constrained optimization algorithm with negative curvature directions and spectral projected gradients. Computing [Suppl] **15**, 49–60 (2001)
13. Birgin, E.G., Martínez, J.M.: Large-scale active-set box-constrained optimization method with spectral projected gradients. Comput. Optim. Appl. **23**(1), 101–125 (2002)
14. Birgin, E.G., Martínez, J.M.: Structured minimal-memory inexact quasi-Newton method and secant preconditioners for augmented Lagrangian optimization. Comput. Optim. Appl. **39**(1), 1–16 (2008)
15. Birgin, E.G., Martínez, J.M.: Augmented Lagrangian method with nonmonotone penalty parameters for constrained optimization. Comput. Optim. Appl. **51**(3), 941–965 (2012)
16. Birgin, E.G., Martínez, J.M.: Practical Augmented Lagrangian Methods for Constrained Otimization, Fundamentals of Algorithms, vol. 10. SIAM, Philadelphia, PA (2014)

17. Birgin, E.G., Martínez, J.M.: The use of quadratic regularization with a cubic descent condition for unconstrained optimization. SIAM J. Optim. **27**(2), 1049–1074 (2017)
18. Birgin, E.G., Martínez, J.M., Ronconi, D.P.: Optimizing the packing of cylinders into a rectangular container: a nonlinear approach. Eur. J. Oper. Res. **160**(1), 19–33 (2005)
19. Birgin, E.G., Sobral, F.N.C.: Minimizing the object dimensions in circle and sphere packing problems. Comput. Oper. Res. **35**(7), 2357–2375 (2008)
20. Cartis, C., Gould, N.I.M., Toint, P.L.: On the complexity of steepest descent, Newton's and regularized Newton's methods for nonconvex unconstrained optimization problems. SIAM J. Optim. **20**(6), 2833–2852 (2010)
21. Cartis, C., Gould, N.I.M., Toint, P.L.: Adaptive cubic regularisation methods for unconstrained optimization. Part I: motivation, convergence and numerical results. Math. Program. **127**(2), 245–295 (2011)
22. Cartis, C., Gould, N.I.M., Toint, P.L.: Adaptive cubic regularisation methods for unconstrained optimization. Part II: worst-case function- and derivative-evaluation complexity. Math. Program. **130**(2), 295–319 (2011)
23. Curtis, F.E., Mitchell, T., Overton, M.L.: A BFGS-SQP method for nonsmooth, nonconvex, constrained optimization and its evaluation using relative minimization profiles. Optim. Methods Softw. **32**(1), 148–181 (2017)
24. Curtis, F.E., Robinson, D.P., Samadi, M.: A trust region algorithm with a worst-case iteration complexity of $\mathcal{O}(\epsilon^{-3/2})$ for nonconvex optimization. Math. Program. **162**(1), 1–32 (2017)
25. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Math. Program. **91**(2), 201–213 (2002)
26. Dussault, J.P.: ARCq: a new adaptive regularization by cubics. Optim. Methods Softw. **33**(2), 322–335 (2018)
27. Grapiglia, G.N., Yuan, J., Yuan, Y.: On the convergence and worst-case complexity of trust-region and regularization methods for unconstrained optimization. Math. Program. **152**(1), 491–520 (2015)
28. Griewank, A.: The modification of Newton's method for unconstrained optimization by bounding cubic terms. Technical Report. NA/12, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, Cambridge, England (1981)
29. Martínez, J.M., Raydan, M.: Cubic-regularization counterpart of a variable-norm trust-region method for unconstrained minimization. J. Glob. Optim. **68**(2), 367–385 (2017)
30. Martínez, L., Andrade, R., Birgin, E.G., Martínez, J.M.: PACKMOL: a package for building initial configurations for molecular dynamics simulations. J. Comput. Chem. **30**(13), 2157–2164 (2009)
31. Mitchell, T.: Robust and efficient methods for approximation and optimization of stability measures. Ph.D. thesis, Department of Computer Science, New York University, New York (2014)
32. Moré, J.J., Garbow, B.S., Hillstrom, K.E.: Testing unconstrained optimization software. ACM Trans. Math. Softw. **7**(1), 17–41 (1981)
33. Nesterov, Y., Polyak, B.T.: Cubic regularization of Newton method and its global performance. Math. Program. **108**(1), 177–205 (2006)