Secant acceleration of sequential residual methods for solving large-scale nonlinear systems of equations*

E. G. Birgin† J. M. Martínez‡

December 24, 2020

Abstract

Sequential Residual Methods try to solve nonlinear systems of equations $F(x) = 0$ by iteratively updating the current approximate solution along a residual-related direction. Therefore, memory requirements are minimal and, consequently, these methods are attractive for solving large-scale nonlinear systems. However, the convergence of these algorithms may be slow in critical cases; therefore, acceleration procedures are welcome. Anderson-like accelerations are widely used in electronic structure calculations to improve a fixed point algorithm for finding Self Consistent Field (SCF) solutions of Hartree-Fock models. In this paper, it is showed how to apply this type of acceleration to Sequential Residual Methods. The performance of the resulting algorithm is illustrated by applying it to the solution of very large problems coming from the discretization of partial differential equations.

Key words: Nonlinear systems of equations, Sequential Residual Methods, acceleration, large-scale problems.

AMS subject classifications: 65H10, 65K05, 90C53.

1 Introduction

In the process of solving many real-life problems, it is necessary to handle large-scale nonlinear systems of equations. The most obvious choice for solving these systems is Newton's method, which requires to solve a possibly large and sparse linear system of equations at each iteration. Although being very effective in many cases, Newton's method cannot be employed for solving very large problems when the Jacobian is unavailable or when it has an unfriendly structure that makes its factorization unaffordable. On the other hand, Inexact Newton methods, that solve the Newtonian linear system approximately at each iteration, are usually effective [14, 17, 18]. Inexact-Newton methods based on linear iterative solvers as GMRES may need many matrix-vector products per iteration. Usually, matrix-vector products of the form $J(x)v$ are replaced with incremental quotients $[F(x + hv) - F(x)]/h$, a procedure that does not deteriorate the overall performance of GMRES [10, 41]. However, when GMRES requires many matrix-vector products for providing a suitable approximate solution to the Newtonian linear system, the number of residual evaluations per inexact-Newton iteration may be big. Additional residual evaluations may also be necessary to decide acceptance of trial points at every iteration.

This state of facts led to the introduction of algorithms in which the number of residual evaluations used to compute trial points at each iteration is minimal, as well as the memory used to store directions and the computer effort of linear algebra calculations. DF-SANE [30] was introduced for solving large problems and was used for solving equilibrium models for the determination of industrial prices [37], multifractal analysis of spot prices [45], elastoplastic contact problems [20, 21], and PDE equations in reservoir simulations [38], among

*This work was supported by FAPESP (grants 2013/07375-0, 2016/01860-1, and 2018/24293-0) and CNPq (grants 302538/2019-4 and 302682/2019-8).

†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. e-mail: egbirgin@ime.usp.br

‡Department of Applied Mathematics, Institute of Mathematics, Statistics, and Scientific Computing (IMECC), State University of Campinas, 13083-859 Campinas SP, Brazil. e-mail: martinez@ime.unicamp.br

others. Improved versions of DF-SANE were given in [29, 36]. However, the pure form of DF-SANE may be ineffective for some large problems in which it is necessary to perform many backtrackings per iteration in order to obtain sufficient descent. Therefore, on the one hand, it is necessary to investigate alternative choices of trial steps and, on the other hand, acceleration procedures are welcome.

Acceleration devices for iterative algorithms are frequent in the Numerical Analysis literature [1, 7, 8, 26, 47, 48]. They incorporate useful information from previous iterations instead of expending evaluations at the current one. In particular, Anderson's acceleration introduced in [1] is known to produce very good results when associated with fixed-point iterations [4, 12, 19, 26, 47], specifically those originated in Self-Consistent Field (SCF) approaches for electronic structure calculations [32, 42]. Anderson's acceleration is closely related to some quasi-Newton methods [11, 13, 25, 19, 34] and multipoint secant algorithms [2, 23, 27, 35, 40, 49]. The recent survey [9] sheds a lot of light on the properties of Anderson's acceleration, generalizations, and relations with other procedures for accelerating the convergence of sequences.

This work introduces a generalized and accelerated version of DF-SANE. The generalization consists of allowing non-residual (although residual-related) directions. The acceleration is based on the multipoint secant idea and Anderson's acceleration, taking advantage of the residual-like direction steps. Global convergence results that extend the theory of DF-SANE are given.

The paper is organized as follows. Section 2 introduces the accelerated sequential residual methods. Global convergence is established in Section 3. Section 4 describes the acceleration process in detail. Implementation features and numerical experiments are given in Sections 5 and 6, respectively. The last section presents the conclusions.

Notation. The symbol $\|\cdot\|$ denotes the Euclidean norm. $\mathbb{N} = \{0, 1, 2, \dots\}$ denotes the set of natural numbers. $J(x)$ denotes the Jacobian matrix of $F : \mathbb{R}^n \to \mathbb{R}^n$ computed at x. For all $x \in \mathbb{R}^n$, we denote $g(x) = J(x)^T F(x) = \nabla \frac{1}{2}\|F(x)\|_2^2$. If $\{z_k\}_{k\in\mathbb{N}}$ is a sequence and $K = \{k_1, k_2, k_3, \dots\}$ is an infinite sequence of natural numbers such that $k_i < k_j$ if $i < j$, we denote

$$\lim_{k\in K} z_k = \lim_{j\to\infty} z_{k_j}.$$

2 Accelerated sequential residual methods

Given $F : \mathbb{R}^n \to \mathbb{R}^n$, consider the problem of finding $x \in \mathbb{R}^n$ such that

$$F(x) = 0. \tag{1}$$

A radical iterative approach for solving (1) is to employ only residuals as search directions. Given $\sigma > 0$, problem (1) is clearly equivalent to $x = x - \sigma F(x)$. This trivial observation motivates the introduction of a fixed-point method given by $x^{k+1} = x^k - \sigma_k F(x^k)$, where σ_k is defined at every iteration. Methods based on this approach will be called Sequential Residual Methods (SRM) in the present paper.

Popular SRM were inspired by the Barzilai-Borwein or spectral choice for the minimization of functions [3, 43, 44]. Defining

$$s^k = x^{k+1} - x^k \text{ and } y^k = F(x^{k+1}) - F(x^k), \tag{2}$$

algorithms SANE [31] and DF-SANE [30] compute

$$\sigma_{k+1} = \|s^k\|^2/(y^k)^T s^k, \tag{3}$$

safeguarded in such a way that $|\sigma_{k+1}|$ is bounded and bounded away from zero. This formula had been used in the context of self-scaling variable metric methods for minimization [39] as it provides a scale invariant diagonal first approximation of the Hessian. The choice of σ_{k+1} may be justified with the same arguments that Raydan [43] employed for the choice of the Barzilai-Borwein or spectral step in minimization problems. After the computation of x^{k+1}, we consider the (generally unsolvable) problem of satisfying the secant equation [16] $B_{k+1}s^k = y^k$ subject to $B_{k+1} = cI$. This leads to the minimization of $\|cIs^k - y^k\|^2$, whose solution, if $s^k \neq 0$, is $c = (y^k)^T s^k/\|s^k\|^2$. Therefore, a "natural" residual-based iteration for solving problem (1) could be given by $x^{k+1} = x^k - \sigma_k F(x^k)$, with σ_0 arbitrary and σ_{k+1} defined by a safeguarded version of (3) for all $k \geq 0$.

However, unlike the case of unconstrained minimization, in which $F(x^k)$ is a gradient, the direction $d^k = -\sigma_k F(x^k)$ may not be a descent direction for the natural merit function $f(x)$ defined by

$$f(x) = \frac{1}{2}\|F(x)\|^2 \text{ for all } x \in \mathbb{R}^n.$$

In SANE [31], a test is performed in order to verify whether $F(x^k)$ is a descent direction. If this is the case, since $\nabla f(x) = J(x)^T F(x)$, we should have

$$F(x^k)^T J(x^k) F(x^k) < 0.$$

In order to avoid the employment of derivatives, SANE employs the approximation

$$J(x^k)F(x^k) \approx \frac{F(x^k + hF(x^k)) - F(x^k)}{h},$$

for a small $h > 0$. In this way, the descent test is equivalent to

$$F(x^k)^T F(x^k + hF(x^k)) < \|F(x^k)\|^2,$$

which requires and auxiliary functional evaluation per iteration. The necessity of an auxiliary residual evaluation per iteration in SANE motivated the introduction of DF-SANE [30]. Roughly speaking, in DF-SANE, one gets descent by starting with the trial point $x^k - \sigma_k F(x^k)$ and proceeding to a double backtracking scheme along positive and negative directions, aiming that $\|F(x^{k+1})\|$ be sufficiently smaller than the maximum value of the residual norm in M consecutive past iterations, where M is given.

The description of the SRM algorithm provided in this section aims to emphasize the aspects that influence theoretical convergence properties. For this reason, acceleration steps appear only as a small detail in the description of the algorithm, although, in practice, they are essential for the algorithm robustness and efficiency. The description of the algorithm follows.

Algorithm 2.1. Let $\gamma \in (0,1)$, $0 < \sigma_{\min} < \sigma_{\max} < \infty$, $0 < \tau_{\min} < \tau_{\max} < 1$, a positive integer M, a sequence $\{\eta_k\}$ such that $\eta_k > 0$ for all $k \in \mathbb{N}$ and

$$\lim_{k\to\infty} \eta_k = 0, \tag{4}$$

and $x_0 \in \mathbb{R}^n$ be given. Set $k \leftarrow 0$.

Step 1. If $F(x^k) = 0$, then terminate the execution of the algorithm.

Step 2. Choose σ_k such that $|\sigma_k| \in [\sigma_{\min}, \sigma_{\max}]$ and $v^k \in \mathbb{R}^n$ such that $\|v^k\| = \|F(x^k)\|$. Compute

$$\bar{f}_k = \max\{f(x^k), \ldots, f(x^{\max\{0,k-M+1\}})\}. \tag{5}$$

Step 2.1. Set $\alpha_+ \leftarrow 1$ and $\alpha_- \leftarrow 1$.

Step 2.2. Set $d \leftarrow -\sigma_k v^k$ and $\alpha \leftarrow \alpha_+$. Consider

$$f(x^k + \alpha d) \le \bar{f}_k + \eta_k - \gamma\alpha^2 f(x^k). \tag{6}$$

If (6) holds, then define $d^k = d$ and $\alpha_k = \alpha$ and go to Step 3.

Step 2.3. Set $d \leftarrow \sigma_k v^k$ and $\alpha \leftarrow \alpha_-$. If (6) holds, then define $d^k = d$ and $\alpha_k = \alpha$ and go to Step 3.

Step 2.4. Choose $\alpha_+^{\text{new}} \in [\tau_{\min}\alpha_+, \tau_{\max}\alpha_+]$ and $\alpha_-^{\text{new}} \in [\tau_{\min}\alpha_-, \tau_{\max}\alpha_-]$, set $\alpha_+ \leftarrow \alpha_+^{\text{new}}$, $\alpha_- \leftarrow \alpha_-^{\text{new}}$, and go to Step 2.2.

Step 3. Compute x^{k+1} such that $f(x^{k+1}) \le f(x^k + \alpha_k d^k)$, set $k \leftarrow k+1$, and go to Step 1.

As in DF-SANE, the sufficient decrease in (6) corresponds to a nonmonotone strategy that combines the ones introduced in [24] and [33]. The main differences of Algorithm 2.1 with respect to DF-SANE are the presence of accelerations at Step 3 and the choice of the non-accelerated step in a residual-related way, but not necessarily in the residual direction. The DF-SANE method presented in [30] is the particular case of Algorithm 2.1 in which $v^k = F(x^k)$ and $x^{k+1} = x^k + \alpha_k d^k$.

3 Global convergence

In this section we prove global convergence properties of Algorithm 2.1. Our main purpose is to find solutions of $F(x) = 0$ or, at least, points at which the residual norm $\|F(x)\|$ is as small as desired, given an arbitrary tolerance. However, this purpose could be excessively ambitious because, in the worst case, solutions of the system, or even approximate solutions, may not exist. For this reason we analyze the situations in which convergence to a (stationary) point, at which the gradient of the sum of squares vanishes, occur. The option of taking directions that are not residuals but are residual-related in the sense that their norms coincide with those of the residuals is crucial for this purpose. Roughly speaking, we will prove that, taking random residual-related directions an infinite number of times (but not at all iterations) convergence to stationary points necessarily takes place.

In Lemma 3.1 we prove that, given an arbitrary tolerance $\varepsilon > 0$, either Algorithm 2.1 finds an approximate solution such that $\|F(x^k)\| \leq \varepsilon$ in a finite number of iterations or produces an infinite sequence of steps $\{\alpha_k\}$ that tends to zero. For this purpose we will only use continuity of F. The lemma begins with a simple proof that the iteration is well defined.

Lemma 3.1 *Assume that F is continuous, $x^k \in \mathbb{R}^n$ is an arbitrary iterate of Algorithm 2.1, and $\|F(x^k)\| \neq 0$. Then, the iterate x^{k+1} satisfying (6) is well defined. Moreover, if the algorithm generates an infinite sequence $\{x^k\}$, then there exists an infinite subset of indices $K_1 \subset \mathbb{N}$ such that*

$$\lim_{k \in K_1} \|F(x^k)\| = 0 \tag{7}$$

or

$$\lim_{k \to \infty} \alpha_k = 0. \tag{8}$$

Proof: The fact that x^{k+1} is always well defined follows from the continuity of F using that $\eta_k > 0$ and that α_k can be as small as needed. Assume that the algorithm generates an infinite sequence $\{x^k\}$. Suppose that (7) does not hold. Then, there exists $c_1 > 0$ such that

$$\|F(x^k)\| > c_1 \text{ for all } k \in \mathbb{N}. \tag{9}$$

If (8) does not hold either, then there exists a subsequence $K_2 \subset \mathbb{N}$ and $c_2 > 0$ such that

$$\alpha_k > c_2 \text{ for all } k \in K_2.$$

Then, by (9),

$$\alpha_k^2 f(x^k) > (c_1 c_2)^2 \text{ for all } k \in K_2. \tag{10}$$

Then, since x^{k+1} is well defined, for all $k \in K_2$,

$$f(x^{k+1}) \leq \bar{f}_k + \eta_k - \gamma \alpha_k^2 f(x^k) \leq \bar{f}_k + \eta_k - \gamma (c_1 c_2)^2, \tag{11}$$

where

$$\bar{f}_k = \max\{f(x^k), \ldots, f(x^{\max\{0, k-M+1\}})\}. \tag{12}$$

Since $\eta_k \to 0$, then there exists $k_1 \in \mathbb{N}$ such that for all $k \in K_2$ and $k \geq k_1$,

$$f(x^{k+1}) \leq \bar{f}_k - \gamma (c_1 c_2)^2 / 2. \tag{13}$$

By (12) and (13), there exists a subsequence for which $f(x^k) \to -\infty$, which contradicts the fact that $f(x) \geq 0$ for all $x \in \mathbb{R}^n$. □

In Lemma 3.2 we prove that we have two possibilities: Either, given an arbitrary $\varepsilon > 0$, we find an iterate such that $\|F(x^k)\| \leq \varepsilon$ or, for every limit point of the sequence $\{x^k\}$, the gradient of f is orthogonal to every possible limit of search directions. Of course, the second possibility necessarily occurs when the system has no solutions.

Lemma 3.2 *Assume that $F(x)$ admits continuous derivatives for all $x \in \mathbb{R}^n$ and $\{x^k\}$ is generated by Algorithm 2.1. Assume that $\{\|F(x^k)\|\}$ is bounded away from zero and $x_* \in \mathbb{R}^n$ is a limit point of $\{x^k\}$ such that $\lim_{k\in K_1} x^k = x_*$. Then, the set of limit points of $\{v^k\}_{k\in K_1}$ is nonempty. Moreover, for every limit point of $\{v^k\}_{k\in K_1}$, we have that*

$$\langle J(x_*)v, F(x_*)\rangle = \langle v, J(x_*)^T F(x_*)\rangle = 0. \tag{14}$$

Proof: By Lemma 3.1,

$$\lim_{k\to\infty} \alpha_k = 0. \tag{15}$$

By continuity, since $\lim_{k\in K_1} x^k = x_*$ we have that $\lim_{k\in K_1} F(x^k) = F(x_*)$. Therefore, the sequence $\{F(x^k)\}_{k\in K_1}$ is bounded. Since $\|v^k\| = \|F(x^k)\|$ for all k, we have that the sequence $\{v^k\}_{k\in K_1}$ is bounded too; therefore, it admits a limit point $v \in \mathbb{R}^n$, as we wanted to prove.

Let K_2 be an infinite subset of K_1 and $v \in \mathbb{R}^n$ be such that $\lim_{k\in K_2} v^k = v$. Since the first trial value for α_k at each iteration is 1, (15) implies that there exists K_3, an infinite subset of K_2, $\alpha_{k,+} > 0$, $\alpha_{k,-} > 0$, $\sigma_k \in [\sigma_{\min}, \sigma_{\max}]$, and $d^k = -\sigma_k v^k$ such that

$$\lim_{k\in K_3} \sigma_k = \sigma \in [\sigma_{\min}, \sigma_{\max}],$$

$$\lim_{k\in K_3} \alpha_{k,+} = \lim_{k\in K_2} \alpha_{k,-} = 0,$$

and, for all $k \in K_3$,

$$f(x^k + \alpha_{k,+}d^k) > \bar{f}_k + \eta_k - \gamma\alpha_{k,+}^2 f(x^k),$$

and

$$f(x^k - \alpha_{k,-}d^k) > \bar{f}_k + \eta_k - \gamma\alpha_{k,-}^2 f(x^k).$$

Therefore, by the definition (5) of $\bar{f}_k$, for all $k \in K_3$,

$$f(x^k + \alpha_{k,+}d^k) > f(x^k) + \eta_k - \gamma\alpha_{k,+}^2 f(x^k)$$

and

$$f(x^k - \alpha_{k,-}d^k) > f(x^k) + \eta_k - \gamma\alpha_{k,-}^2 f(x^k).$$

So, since $\eta_k > 0$,

$$\frac{f(x^k + \alpha_{k,+}d^k) - f(x^k)}{\alpha_{k,+}} > -\gamma\alpha_{k,+} f(x^k)$$

and

$$\frac{f(x^k - \alpha_{k,-}d^k) - f(x^k)}{\alpha_{k,-}} > -\gamma\alpha_{k,-} f(x^k)$$

for all $k \in K_3$. Thus, by the Mean Value Theorem, there exist $\xi_{k,+} \in [0, \alpha_{k,+}]$ and $\xi_{k,-} \in [0, \alpha_{k,-}]$ such that

$$\langle \nabla f(x^k + \xi_{k,+}d^k), d^k\rangle > -\gamma\alpha_{k,+} f(x^k) \tag{16}$$

and

$$\langle \nabla f(x^k - \xi_{k,-}d^k), d^k\rangle > -\gamma\alpha_{k,-} f(x^k) \tag{17}$$

for all $k \in K_3$. Taking limits for $k \in K_3$ in both sides of (16) and (17) we get

$$\langle \nabla f(x_*), \sigma v\rangle = 0. \tag{18}$$

Therefore, (14) is proved. □

Theorem 3.1 states a sufficient condition for the annihilation of the gradient of $f(x)$ at a limit point of the sequence. For that purpose, we will assume that the absolute value of the cosine of the angle determined by v^k and $J(x^k)^T v^k$ is bigger than a tolerance $\omega > 0$ infinitely many times. Note that we do not require this condition to hold for every $k \in \mathbb{N}$. The distinction between these two alternatives is not negligible. For example, if v^k is chosen infinitely many times as a random vector the required angle condition holds with probability 1. Conversely, if we require the fulfillment of the angle condition for all $k \in \mathbb{N}$ and we choose random directions, the probability of fulfillment would be zero.

Theorem 3.1 *Assume that $F(x)$ admits continuous derivatives for all $x \in \mathbb{R}^n$ and $\{x^k\}$ is generated by Algorithm 2.1. Suppose that $\sum_{k=0}^{\infty} \eta_k = \eta < \infty$, the level set $\{x \in \mathbb{R}^n \mid f(x) \leq f(x^0) + \eta\}$ is bounded, and there exists $\omega > 0$ such that*

$$|\langle v^k, J(x^k)^T F(x^k)\rangle| \geq \omega \|v^k\| \|J(x^k)^T F(x^k)\| \tag{19}$$

for infinitely many indices $k \in K \subset \mathbb{N}$. Then, for any given $\varepsilon > 0$, there exists $k \in \mathbb{N}$ such that $\|J(x^k)^T F(x^k)\| \leq \varepsilon$. Moreover, there exists a limit point x_* of the sequence generated by the algorithm such that $\|J(x_*)^T F(x_*)\| = 0$.

Proof: If the sequence generated by the algorithm stops at some k such that $\|F(x^k)\| = 0$ the thesis obviously holds. By the boundedness of the level set defined by $f(x^0) + \eta$, the sequence $\{x^k\}$ is bounded. By the continuity of F, the sequence $\{F(x^k)\}$ is bounded as well. By the definition of v^k, the sequence $\{v^k\}$ is also bounded. Therefore, there exist an infinite set $K_1 \subset K$, $x_* \in \mathbb{R}^n$, and $v \in \mathbb{R}^n$ such that $\lim_{k \in K_1} x^k = x_*$ and $\lim_{k \in K_1} v^k = v$. Therefore, by Lemma 3.2,

$$\langle v, J(x_*)^T F(x_*)\rangle = 0.$$

So, by the convergence of $\{v^k\}$ and $\{x^k\}$ for $k \in K_1$ and the continuity of F and J,

$$\lim_{k \in K_1} \langle v^k, J(x^k)^T F(x^k)\rangle = 0.$$

Therefore, by (19), since $K_1 \subset K$,

$$\lim_{k \in K_1} \|v^k\| \|J(x^k)^T F(x^k)\| = 0. \tag{20}$$

If $\|F(x^k)\|$ tends to zero for some subsequence of K_1 the thesis obviously holds. Otherwise, there exists $c > 0$ such that $\|F(x^k)\| > c$ for all $k \in K_1$. Then, $\|v^k\| \geq c$ for all $k \in K_1$. So, by (20),

$$\lim_{k \in K_1} \|J(x^k)^T F(x^k)\| = 0. \tag{21}$$

Therefore, the thesis is proved. □

Corollary 3.1 *Suppose that the assumptions of Theorem 3.1 hold. Assume, moreover, that there exists $\gamma > 0$ such that for all $k \in \mathbb{N}$,*

$$\|J(x^k)^T F(x^k)\| \geq \gamma \|F(x^k)\|. \tag{22}$$

Then, given $\varepsilon > 0$, there exists an iterate k such that $\|F(x^k)\| \leq \varepsilon$. Moreover, there exists a limit point x_* of the sequence generated by the algorithm such that $\|F(x_*)\| = 0$.

Proof: The thesis of this corollary follows directly from (22) and Theorem 3.1. □

Corollary 3.2 *Suppose that the assumptions of Theorem 3.1 hold and $J(x)$ is nonsingular with $\|J(x)^{-1}\|$ uniformly bounded for all $x \in \mathbb{R}^n$. Then, for all $\ell = 1, 2, \ldots$, there exists an iterate $k(\ell)$ such that $\|F(x^{k(\ell)})\| \leq 1/\ell$. Moreover, at every limit point x_* of the sequence $\{x^{k(\ell)}\}$ we have that $\|F(x_*)\| = 0$.*

Proof: Since $\|F(x^k)\| = \|J(x^k)^{-T} J(x^k)^T F(x^k)\| \leq \|J(x^k)^{-1}\| \|J(x^k)^T F(x^k)\|$, a sufficient condition for the fulfillment of (22) is the existence and uniform boundedness of $\|J(x^k)^{-1}\|$. □

The lemma below shows that, if $\|F(x^k)\| \to 0$ for a subsequence, then it goes to zero for the whole sequence. We will need the following Assumptions A and B. Assumption A guarantees that the difference between consecutive iterates is not greater than a multiple of the residual norm. Note that such assumption holds trivially when the iteration is not accelerated and, so, $x^{k+1} = x^k + \alpha_k d^k$. Therefore, Assumption A needs to be stated only with respect to accelerated steps. Assumption B merely states that F is uniformly continuous at least restricted to the set of iterates.

Assumption A *There exists $c > 0$ such that, for all $k \in \mathbb{N}$, whenever $x^{k+1} \neq x^k + \alpha_k d^k$, we have that*

$$\|x^{k+1} - x^k\| \leq c \|F(x^k)\|.$$

Assumption B *For all $\varepsilon > 0$, there exists δ such that, whenever $\|x^{k+1} - x^k\| \leq \delta$ one has that $\|F(x^{k+1}) - F(x^k)\| \leq \varepsilon$.*

Lemma 3.3 below is stronger than Theorem 2 of [30] because here we do not assume the existence of a limit point.

Lemma 3.3 *Suppose that F is continuous, $\sum_{k=0}^{\infty} \eta_k < \infty$, Algorithm 2.1 generates the infinite sequence $\{x^k\}$, Assumptions A and B hold, and there exists an infinite subsequence defined by the indices in K_1 such that*

$$\lim_{k \in K_1} \|F(x^k)\| = 0.$$

Then,

$$\lim_{k \to \infty} \|F(x^k)\| = 0 \tag{23}$$

and

$$\lim_{k \to \infty} \|x^{k+1} - x^k\| = 0. \tag{24}$$

Proof: Assume that (23) is not true. Then, there exists an infinite subsequence $\{x^k\}_{k \in K_2}$ such that

$$f(x^k) > c > 0 \tag{25}$$

for all $k \in K_2$.

Let us prove first that

$$\lim_{k \in K_1} \|F(x^{k+1})\| = 0. \tag{26}$$

For this purpose suppose that $\varepsilon > 0$. By the hypothesis, there exists $k_1 \in K_1$ such that for all $k \in K_1, k \geq k_1$, we have that

$$\|F(x^k)\| \leq \varepsilon/2. \tag{27}$$

By Assumption B, there exists $\delta > 0$ such that, whenever $\|x^{k+1} - x^k\| \leq \delta$, one has that

$$\|F(x^{k+1}) - F(x^k)\| \leq \varepsilon/2. \tag{28}$$

By Assumption A, there exists $k_2 \geq k_1$ such that, whenever $k \geq k_2$, $k \in K_1$, we have that

$$\|x^{k+1} - x^k\| \leq \delta. \tag{29}$$

Then, by (27), (28), and (29), we have that, whenever $k \geq k_2$ and $k \in K_1$,

$$\|F(x^{k+1})\| \leq \varepsilon.$$

This proves (26).

By induction, we deduce that for all $j = 1, \ldots, M-1$,

$$\lim_{k \in K_1} \|F(x^{k+j})\| = 0.$$

Therefore,

$$\lim_{k \in K_1} \max\{f(x^k), \ldots, f(x^{k+M-1})\} = 0. \tag{30}$$

But

$$\begin{aligned} f(x^{k+M}) &\leq \max\{f(x^k), \ldots, f(x^{k+M-1})\} + \eta_{k+M-1} - \alpha_{k+M-1} f(x^{k+M-1}) \\ &\leq \max\{f(x^k), \ldots, f(x^{k+M-1})\} + \eta_{k+M-1}. \end{aligned}$$

Analogously,

$$\begin{aligned} f(x^{k+M+1}) &\leq \max\{f(x^{k+1}), \ldots, f(x^{k+M})\} + \eta_{k+M} - \alpha_{k+M} f(x^{k+M}) \\ &\leq \max\{f(x^{k+1}), \ldots, f(x^{k+M})\} + \eta_{k+M} \\ &\leq \max\{f(x^k), \ldots, f(x^{k+M-1})\} + \eta_{k+M-1} + \eta_{k+M}; \end{aligned}$$

and, inductively,

$$f(x^{k+M+j}) \leq \max\{f(x^k), \ldots, f(x^{k+M-1})\} + \eta_{k+M-1} + \eta_{k+M} + \cdots + \eta_{k+M+j-1}$$

for all $j = 0, 1, \ldots, M-1$. Therefore,

$$\max\{f(x^{k+M}), \ldots, f(x^{k+2M-1})\} \leq \max\{f(x^k), \ldots, f(x^{k+M-1})\} + \sum_{j=k+M-1}^{k+2M-2} \eta_j. \tag{31}$$

By induction on $\ell = 1, 2, \ldots$, we obtain that

$$\max\{f(x^{k+\ell M}), \ldots, f(x^{k+(\ell+1)M-1})\} \leq \max\{f(x^k), \ldots, f(x^{k+M-1})\} + \sum_{j=k+M-1}^{k+\ell M-2} \eta_j. \tag{32}$$

Therefore, for all $\ell = 1, 2, \ldots$ we have:

$$\max\{f(x^{k+\ell M}), \ldots, f(x^{k+(\ell+1)M-1})\} \leq \max\{f(x^k), \ldots, f(x^{k+M-1})\} + \sum_{j=k+M-1}^{\infty} \eta_j. \tag{33}$$

By (30), the summability of η_j and (33), there exists $k_1 \in K_1$ such that for all $k \in K_1$ such that $k \geq k_1$,

$$\max\{f(x^k), \ldots, f(x^{k+M-1})\} + \sum_{j=k+M-1}^{\infty} \eta_j \leq c/2.$$

Therefore, by (33), for all $k \in K_1$ such that $k \geq k_1$ and all $\ell = 1, 2, \ldots$,

$$\max\{f(x^{k+\ell M}), \ldots, f(x^{k+(\ell+1)M-1})\} \leq c/2. \tag{34}$$

Clearly, this is incompatible with the existence of a subsequence such that (25) holds. Therefore, (23) is proved. Then, by Assumption A and (23), (24) also holds. □

Theorem 3.2 *Suppose that the assumptions of Theorem 3.1 hold, $J(x)$ is nonsingular and $\|J(x)^{-1}\|$ is uniformly bounded for all $x \in \mathbb{R}^n$. Then, there exists $x_* \in \mathbb{R}^n$ such that $F(x_*) = 0$ and $\lim_{k\to\infty} x^k = x_*$.*

Proof: By Corollary 3.2, there exists a subsequence $\{F(x^k)\}_{k\in K_1}$ that converges to a point x_* such that $F(x_*) = 0$. Then, by Lemma 3.3,

$$\lim_{k\to\infty} F(x^k) = 0 \tag{35}$$

and

$$\lim_{k\to\infty} \|x^{k+1} - x^k\| = 0.$$

Since $J(x_*)$ is nonsingular, by the Inverse Function Theorem, there exists $\delta > 0$ such that $\|F(x)\| > 0$ whenever $0 < \|x - x_*\| \leq \delta$. Therefore, given $\varepsilon \in (0, \delta]$ arbitrary, there exists $c > 0$ such that

$$\|F(x)\| \geq c \text{ whenever } \varepsilon \leq \|x - x_*\| \leq \delta.$$

By (35), the number of iterates in the set defined by $\varepsilon \leq \|x - x_*\| \leq \delta$ is finite. On the other hand, if k is large enough, one has that $\|x^{k+1} - x^k\| \leq \delta - \varepsilon$. Since there exists a subsequence of $\{x^k\}$ that tends to x_* it turns out that $\|x^k - x_*\| \leq \varepsilon$ if k is large enough. Since ε was arbitrary this implies that $\lim_{k\to\infty} x^k = x_*$ as we wanted to prove. □

We finish this section with a theorem that states that, under some assumptions, if in Algorithm 2.1 we have $x^{k+1} = x^k + \alpha_k d^k$ for all k, i.e. without accelerations, then $\{x^k\}$ converges superlinearly to a solution. A similar result was proved in [22, Thm. 3.1] with respect to a spectral gradient methods with retards for quadratic unconstrained minimization. We will need two assumptions. In Assumption C, the sequence generated by Algorithm 2.1 is assumed to be generated by the spectral residual choice with the first step accepted at each iteration and without acceleration. In Assumption D, the Jacobian $J(x)$ is assumed to be Lipschitz-continuous.

Assumption C *Assume that, at Step 2 of Algorithm 2.1, we choose*

$$v^k = F(x^k)$$

and that there exists $k_0 \geq 1$ such that for all $k \geq k_0$,

$$(s^{k-1})^T y^{k-1} \neq 0$$

and

$$\sigma_k = \frac{(s^{k-1})^T s^{k-1}}{(s^{k-1})^T y^{k-1}},$$

where s^k and y^k are defined by (2). Assume, moreover, that, for all $k \geq k_0$, (6) holds with $\alpha = 1$ and, at Step 3, $x^{k+1} = x^k + \alpha_k d^k$.

Assumption D *There exists $L > 0$ such that for all x, z in an open and convex set that contains the whole sequence $\{x^k\}$ generated by Algorithm 2.1, the Jacobian is continuous and satisfies*

$$\|J(z) - J(x)\| \leq L\|z - x\|.$$

Theorem 3.3 *Assume that the sequence $\{x^k\}$, generated by Algorithm 2.1, does not terminate at Step 1 and that Assumptions C and D hold. Suppose that $x_* \in \mathbb{R}^n$ is such that*

$$\lim_{k\to\infty} x^k = x_*, \tag{36}$$

the Jacobian $J(x_*)$ is nonsingular and $s \in \mathbb{R}^n$ is such that

$$\lim_{k\to\infty} \frac{s^k}{\|s^k\|} = s. \tag{37}$$

Then,

$$\lim_{k\to\infty} \frac{(s^k)^T y^k}{(s^k)^T s^k} = s^T J(x_*) s, \tag{38}$$

$$F(x_*) = 0, \tag{39}$$

and

$$x^k \text{ converges Q-superlinearly to } x_*. \tag{40}$$

Proof: By Assumption D, there exists an open and convex set that contains the whole sequence $\{x^k\}$ such that for all x, z in that set,

$$F(z) = F(x) + J(x)(z - x) + O(\|z - x\|^2). \tag{41}$$

By Assumption C, the sequence $\{x^k\}$ is such that for all $k \geq k_0$,

$$x^{k+1} = x^k - \frac{1}{\kappa_k} F(x^k), \tag{42}$$

where

$$\kappa_{k+1} = \frac{(s^k)^T y^k}{(s^k)^T s^k}. \tag{43}$$

By (41),

$$y^k = J(x^k) s^k + O(\|s^k\|^2) \tag{44}$$

for all $k \geq k_0$. Therefore, by (44),

$$\kappa_{k+1} = \frac{(s^k)^T [J(x^k) s^k + O(\|s^k\|^2)]}{(s^k)^T s^k}. \tag{45}$$

Thus, by (36), (37), and the continuity of $J(x)$,

$$\lim_{k\to\infty} \kappa_{k+1} = s^T J(x_*)s. \tag{46}$$

Therefore, (38) is proved.

By (41) we have that

$$F(x^{k+1}) = F(x^k) + J(x^k)s^k + O(\|s^k\|^2).$$

Then, by (42),

$$-\kappa_{k+1}s^{k+1} = -\kappa_k s^k + J(x^k)s^k + O(\|s^k\|^2)$$

for all $k \in \mathbb{N}$. Therefore, for all $k \in \mathbb{N}$,

$$s^{k+1} = \frac{\kappa_k s^k - J(x^k)s^k}{\kappa_{k+1}} - \frac{O(\|s^k\|^2)}{\kappa_{k+1}}.$$

Therefore,

$$s^{k+1} = -\frac{1}{\kappa_{k+1}}[(J(x^k) - \kappa_k I)s^k + O(\|s^k\|^2)].$$

Then

$$\frac{s^{k+1}}{\|s^{k+1}\|} = \frac{-[(J(x^k) - \kappa_k I)s^k + O(\|s^k\|^2)]}{\| - [(J(x^k) - \kappa_k I)s^k + O(\|s^k\|^2)]\|}.$$

So,

$$\frac{s^{k+1}}{\|s^{k+1}\|} = \frac{-[(J(x^k) - \kappa_k I)s^k/\|s^k\| + O(\|s^k\|^2)/\|s^k\|]}{\| - [(J(x^k) - \kappa_k I)s^k/\|s^k\| + O(\|s^k\|^2)/\|s^k\|]\|}. \tag{47}$$

Define

$$\kappa_* = s^T J(x_*)s. \tag{48}$$

Assume that

$$\|(J(x_*) - \kappa_* I)s\| \neq 0. \tag{49}$$

Then, by (49), (38), and (37), taking limits in both sides of (47),

$$s = \frac{-[(J(x_*) - \kappa_* I)s]}{\| - [(J(x_*) - \kappa_* I)s\|}. \tag{50}$$

Pre-multiplying both sides of (50) by s^T, as $s^T s = 1$, we obtain:

$$1 = \frac{-[(s^T J(x_*)s - \kappa_*]}{\| - [(J(x_*) - \kappa_* I)s\|}. \tag{51}$$

Then, by (48), we get a contradiction. This implies that the assumption (49) is false. So,

$$\|(J(x_*) - \kappa_* I)s\| = 0. \tag{52}$$

Therefore,

$$\lim_{k\to\infty} \frac{\kappa_k I s^k - J(x_*)s^k}{\|s^k\|} = 0. \tag{53}$$

But (53) is the Dennis-Moré condition [15] related with the iteration $x^{k+1} = x^k - B_k^{-1}F(x^k)$ with $B_k = \kappa_k I$. Then, we have that $F(x_*) = 0$ and the convergence is superlinear. □

4 Acceleration scheme

In this section, we describe the way in which, at iteration k of Algorithm 2.1, using an acceleration scheme, we compute x^{k+1} at Step 3. The algorithmic description follows below.

Algorithm 4.1. Let the integer number $p \geq 1$ be given.

Step 1. Define $x_{\text{trial}}^{k+1} = x^k + \alpha_k d^k$.

Step 2. Define $\underline{k} = \max\{0, k-p+1\}$,

$$\begin{aligned} s^j &= x^{j+1} - x^j \text{ for } j = \underline{k}, \ldots, k-1, \\ y^j &= F(x^{j+1}) - F(x^j) \text{ for } j = \underline{k}, \ldots, k-1, \\ s^k &= x_{\text{trial}}^{k+1} - x^k, \\ y^k &= F(x_{\text{trial}}^{k+1}) - F(x^k), \\ S_k &= (s^{\underline{k}}, \ldots, s^{k-1}, s^k), \\ Y_k &= (y^{\underline{k}}, \ldots, y^{k-1}, y^k), \end{aligned}$$

and

$$x_{\text{accel}}^{k+1} = x^k - S_k Y_k^\dagger F(x^k), \tag{54}$$

where $Y_k^\dagger$ is the Moore-Penrose pseudoinverse of Y_k.

Step 3. Choose $x^{k+1} \in \left\{x_{\text{trial}}^{k+1}, x_{\text{accel}}^{k+1}\right\}$ such that $\|F(x^{k+1})\| = \min\left\{\|F(x_{\text{trial}}^{k+1})\|, \|F(x_{\text{accel}}^{k+1})\|\right\}$.

The theorem below helps to understand the behavior of Algorithm 2.1 with x^{k+1} given by Algorithm 4.1, called Algorithm 2.1–4.1 from now on. Briefly speaking, we are going to prove that, under some assumptions, the sequence generated by Algorithm 2.1–4.1 converges superlinearly to a solution of $F(x) = 0$. We do not mean that these assumptions are "reasonable", in the sense that they usually, or even frequently, hold. The theorem aims to show the correlation between different properties of the method, which is probably useful to understand the algorithm and, perhaps, to seek modifications and improvements.

Theorem 4.1 *Assume that $\{x^k\}$ is the sequence generated by Algorithm 2.1–4.1. Suppose, in addition, that*

H1: *For all k large enough we have that $x^{k+1} = x_{\text{accel}}^{k+1}$.*

H2: *There exists a positive sequence $\beta_k \to 0$ such that for all k large enough,*

$$\left\|(S_{k+1}Y_{k+1}^\dagger - S_k Y_k^\dagger)y^k\right\| \leq \beta_k \|y^k\|. \tag{55}$$

H3: *There exists $c > 0$ such that*

$$\|S_k Y_k^\dagger F(x^{k+1})\| \geq c\|F(x^{k+1})\| \tag{56}$$

for k large enough.

Then, $\|F(x^k)\|$ converges to 0 Q-superlinearly.

Proof: By (54) and **H1**, we have that, for k large enough,

$$x^{k+1} = x^k - S_k Y_k^\dagger F(x^k). \tag{57}$$

But, by the definition of S_k and Y_k, for all $k \in \mathbb{N}$,

$$S_{k+1}Y_{k+1}^\dagger y^k = s^k.$$

Therefore, by (55) in **H2**,

$$\|S_k Y_k^\dagger y^k - s^k\| \leq \beta_k \|y^k\|. \tag{58}$$

Then, by (57),

$$\|S_k Y_k^\dagger F(x^{k+1})\| \leq \beta_k \|y^k\|. \tag{59}$$

Thus, by (56) in **H3**,

$$c\|F(x^{k+1})\| \leq \beta_k \|y^k\|. \tag{60}$$

So,

$$c\|F(x^{k+1})\| \leq \beta_k(\|F(x^{k+1})\| + \|F(x^k)\|).$$

Thus,

$$(c-\beta_k)\|F(x^{k+1})\| \leq \beta_k \|F(x^k)\|.$$

Therefore,

$$\|F(x^{k+1})\| \leq \frac{\beta_k}{c-\beta_k}\|F(x^k)\|$$

for k large enough, which means that $F(x^k)$ tends to zero Q-superlinearly. □

The rest of this section aims to shed some light on the meaning of the acceleration formula (54). Assume, for a moment, that A is a an approximation of the Jacobian $J(x^k)$ such that $L(x) = Ax + b$ satisfies the interpolation conditions $Ax^{k,j} + b = F(x^{k,j})$ for $j = 1, \ldots, p+1$. Then, it is natural to seek the point $\widehat{x}$ in the affine subspace generated by $Ax^{k,1} + b, \ldots, Ax^{k,p+1} + b$ that minimizes $\|Ax + b\|^2$. This amounts to solve the optimization problem

$$\underset{x\in\mathbb{R}^n,\, \theta\in\mathbb{R}^{p+1}}{\text{Minimize}} \|Ax+b\|^2 \text{ subject to } Ax + b = \theta_1 F(x^{k,1}) + \cdots + \theta_{p+1}F(x^{k,p+1}) \text{ and } \sum_{j=1}^{p+1}\theta_j = 1. \tag{61}$$

Problem (61) is clearly equivalent to

$$\underset{\theta\in\mathbb{R}^{p+1}}{\text{Minimize}} \|\theta_1 F(x^{k,1}) + \cdots + \theta_{p+1}F(x^{k,p+1})\|^2 \text{ subject to } \sum_{j=1}^{p+1}\theta_j = 1. \tag{62}$$

Then, by the interpolation conditions, $A\widehat{x} + b = \theta_1 F(x^{k,1}) + \cdots + \theta_{p+1}F(x^{k,p+1})$ implies that

$$\widehat{x} = \theta_1 x^{k,1} + \cdots + \theta_{p+1}x^{k,p+1}. \tag{63}$$

Therefore, solutions to (61) are obtained by solving (62) and, then, defining $\widehat{x}$ as in (63). *Note that solutions to (61) do not depend on the Jacobian approximation A.* The DIIS acceleration method [42] is usually presented directly in the form (62,63) with $x^{k,j} = x^{k-(p+1)+j}$ for $j = 1, \ldots, p+1$. The accelerated point x^{k+1}_{accel} in (54) corresponds to the minimum-norm solution to (61) with $x^{k,j} = x^{k-p+j}$ for $j = 1, \ldots, p$ and $x^{k,p+1} = x^{k+1}_{\text{trial}}$. Including x^{k+1}_{trial} is important. The reason is that, if we accelerate without considering the residual $F(x^{k+1}_{\text{trial}})$, then the accelerated point lies in the affine subspace generated by the last p iterates. Therefore, the sequence never leaves an affine subspace with dimension p which, in general, does not contain the solution. In other words, even when the acceleration provides a big improvement with respect to the basic residual method trials, residuals at these trials are necessary to define the accelerations.

The Sequential Secant Method (SSM), called Secant Method of $n+1$ points in the classical book of Ortega and Rheinboldt [2, 23, 27, 35, 40, 49], is a classical procedure for solving nonlinear systems of equations. Given $n+1$ consecutive iterates $x^k, x^{k-1}, \ldots, x^{k-n}$, the Sequential Secant Method computes an affine function $L : \mathbb{R}^n \to \mathbb{R}^n$ such that $L(x^j) = F(x^j)$ for $j = k, \ldots, k-n$; and defines x^{k+1} as the unique solution of $L(x) = 0$. The inconveniences of this method are obvious. On the one hand, the affine function $L(x)$ is guaranteed to exist only if the iterates $x^k, x^{k-1}, \ldots, x^{k-n}$ are affinely independent. On the other hand the linear system of equations $L(x) = 0$ may have no solutions or may have infinitely many solutions. When the residuals $F(x^k), F(x^{k-1}), \ldots, F(x^{k-n})$ are affinely independent, the iteration of SSM is well defined and

$$x^{k+1} - x^k = (s^{k-1}, \ldots, s^{k-n})(y^{k-1}, \ldots, y^{k-n})^{-1}F(x^k), \tag{64}$$

where $s^j = x^{j+1} - x^j$ and $y^j = F(x^{j+1}) - F(x^j)$ for $j = 1, \ldots, n$. Note that (64) may be well defined even in cases in which the increments $s^{k-1}, \ldots, s^{k-n}$ are not linearly independent. In fact, lack of linear independence of

$s^{k-1}, \ldots, s^{k-n}$ is not unusual. If some of the components of $F(x)$ are affine functions, the corresponding linear equations tend to be satisfied after a small number of iterations and, from those iterations on, all the iterates lie in the affine subspace determined by the previous iterates, whose dimension is smaller than n. Lack of linear independence of $y^{k-1}, \ldots, y^{k-n}$ also occurs, for example, when the system is linear and the Jacobian matrix is singular. Some authors developed different variations of SSM in order to overcome these inconveniences. See [35] and references therein.

5 Implementation

In this section, we discuss the implementation of Algorithm 2.1–4.1.

5.1 Stopping criterion

At Step 1 of Algorithm 2.1, given $\varepsilon > 0$, we replace the stopping criterion $\|F(x^k)\| = 0$ with

$$\|F(x^k)\|_2 \leq \varepsilon. \tag{65}$$

5.2 Choice of the direction and the scaling factor

At Step 2 of Algorithm 2.1, we must choose σ_k and v_k. For this purpose, we considered the residual choice $v_k = -F(x^k)$ for all k setting, arbitrarily, $\sigma_0 = 1$. A natural choice for σ_k ($k \geq 1$) would be the spectral step given by

$$\sigma_k^{\text{spg}} = \frac{(x^k - x^{k-1})^T (x^k - x^{k-1})}{(x^k - x^{k-1})^T (F(x^k) - F(x^{k-1}))}$$

with safeguards that guarantee that $|\sigma_k|$ belong to $[\sigma_{\min}, \sigma_{\max}]$. It turns out that defining $\sigma_{\max} > 1$, preliminary numerical experiments showed that, *in the problems considered in the present work*, Step 2 of Algorithm 2.1 with this choice of σ_k performs several backtrackings per iteration that result in a total number of functional evaluations one order of magnitude larger than the number of iterations; which is an unusual behavior of nonmonotone methods based on the Barzilai-Borwein spectral choice [6, 31, 30, 43, 44]. On the other hand, it was also observed that, at Step 3 of Algorithm 4.1, the acceleration step was always chosen. This means that the costly obtained x_{trial}^{k+1} was always beaten by x_{accel}^{k+1}. These two observations suggested that a more conservative, i.e. small, scaling factor σ_k should be considered. A small σ_k could result in a trial point $x_{\text{trial}}^{k+1} = x^k + \alpha_k d^k$ with $\alpha_k = \pm 1$ that satisfies (6), i.e. no backtracking, and that provides the information required to compute a successful x_{accel}^{k+1} at Step 2 of Algorithm 4.1.

The conservative choice of σ_k employed in our implementation is as follows. We consider an algorithmic parameter

$$h_{\text{init}} > 0 \tag{66}$$

and we define:

$$\sigma_{\min} = \sqrt{\epsilon},\ \sigma_{\max} = 1, \tag{67}$$

$$\sigma_0 = 1, \tag{68}$$

$$\bar{\sigma}_k = h_{\text{init}} \frac{\|x^k - x^{k-1}\|}{\|F(x^k)\|} \text{ for all } k \geq 1, \tag{69}$$

and

$$\sigma_k = \bar{\sigma}_k \text{ if } k \geq 1 \text{ and } \bar{\sigma}_k \in [\max\{1, \|x^k\|\}\sigma_{\min}, \sigma_{\max}]. \tag{70}$$

Finally, if $k \geq 1$ and $\bar{\sigma}_k \notin [\max\{1, \|x^k\|\}\sigma_{\min}, \sigma_{\max}]$, we define

$$\bar{\bar{\sigma}}_k = h_{\text{init}} \frac{\|x^k\|}{\|F(x^k)\|} \tag{71}$$

and we compute σ_k as the projection of $\bar{\bar{\sigma}}_k$ onto the interval $[\max\{1, \|x^k\|\}\sigma_{\min}, \sigma_{\max}]$.

5.3 Procedure for reducing the step

At Step 2.4 of Algorithm 2.1, we compute α_+^{new} as the minimizer of the the univariate quadratic $q(\alpha)$ that interpolates $q(0) = f(x^k)$, $q(\alpha_+) = f(x^k - \alpha_+\sigma_k F(x^k))$, and $q'(0) = -\sigma_k F(x^k)^T \nabla f(x^k) = -\sigma_k F(x^k)^T J(x^k) F(x^k)$. Following [30], since we consider $J(x^k)$ unavailable, we consider $J(x^k) = I$. Thus,

$$\alpha_+^{\text{new}} = \max\left\{\tau_{\min}\alpha_+, \min\left\{\frac{\alpha_+^2 f(x^k)}{f(x^k - \alpha_+\sigma_k F(x^k)) + (2\alpha_+ - 1)f(x^k)}, \tau_{\max}\alpha_+\right\}\right\}.$$

Analogously,

$$\alpha_-^{\text{new}} = \max\left\{\tau_{\min}\alpha_-, \min\left\{\frac{\alpha_-^2 f(x^k)}{f(x^k + \alpha_-\sigma_k F(x^k)) + (2\alpha_- - 1)f(x^k)}, \tau_{\max}\alpha_-\right\}\right\}.$$

5.4 Computing the acceleration

In Step 2 of Algorithm 4.1, computing x_{accel}^{k+1} as defined in (54) is equivalent to computing the minimum-norm least-squares solution $\bar{\omega}$ to the linear system $Y_k\omega = F(x_{\text{trial}}^{k+1})$ and defining $x_{\text{accel}}^{k+1} = x_{\text{trial}}^{k+1} - S_k\bar{\omega}$. In practice, we compute the minimum-norm least-squares solution with a complete orthogonalization of Y_k. Computing this factorization from scratch at each iteration could be expensive. However, by definition, Y_k corresponds to a slight variation of Y_{k-1}. In practice, when Y_k has not full numerical rank, one extra column is added to it; and if Y_k has null numerical rank, a new Y_k is computed from scratch. For completeness, the practical implementation of Algorithm 4.1 is given below.

Algorithm 5.1. Let $0 < h_{\text{small}} < h_{\text{large}}$ and $p \geq 1$ be given. Set $r_{\max} \leftarrow 0$ and $\ell \leftarrow 1$.

Step 1. Define $x_{\text{trial}}^{k+1} = x^k + \alpha_k d^k$.

Step 2.

Step 2.1. If $k = 0$, the set S_k and Y_k as matrices with zero columns. Otherwise, set $S_k \leftarrow S_{k-1}$ and $Y_k \leftarrow Y_{k-1}$.

Step 2.2. If S_k and Y_k have p columns, then remove the leftmost column of S_k and Y_k.

Step 2.3. Add $s^k = x_{\text{trial}}^{k+1} - x^k$ and $y^k = F(x_{\text{trial}}^{k+1}) - F(x^k)$ as rightmost column of S_k and Y_k, respectively; and set $r_{\max} \leftarrow \max\{r_{\max}, \text{rank}(Y_k)\}$.

Step 2.4. If $\text{rank}(Y_k) < r_{\max}$, then execute Steps 2.4.1–2.4.2.

Step 2.4.1. If S_k and Y_k have p columns, then remove the leftmost column of S_k and Y_k.

Step 2.4.2. Add $s_{\text{extra}} = x_{\text{extra}} - x^k$ and $y_{\text{extra}} = F(x_{\text{extra}}) - F(x^k)$ as rightmost column of S_k and Y_k, respectively, where $x_{\text{extra}} = x^k + h_{\text{small}}\, e_\ell$. Set $r_{\max} \leftarrow \max\{r_{\max}, \text{rank}(Y_k)\}$ and $\ell \leftarrow \text{mod}(\ell, n) + 1$.

Step 2.5. If $\text{rank}(Y_k) \neq 0$, then execute Steps 2.5.1–2.5.3.

Step 2.5.1. Compute the minimum-norm least-squares solution $\bar{\omega}$ to the linear system $Y_k\omega = F(x_{\text{trial}}^{k+1})$ and define $x_{\text{accel}}^{k+1} = x_{\text{trial}}^{k+1} - S_k\bar{\omega}$.

Step 2.5.2. If Step 2.4.2 was executed, then remove the rightmost column of S_k and Y_k, i.e. columns s_{extra} and y_{extra}.

Step 2.5.3. If $x_{\text{accel}}^{k+1} \neq x^k$, $\|x_{\text{accel}}^{k+1}\| \leq 10\max\{1, \|x^k\|\}$, and $\|F(x_{\text{accel}}^{k+1})\| < \|F(x_{\text{trial}}^{k+1})\|$, then redefine $x_{\text{trial}}^{k+1} = x_{\text{accel}}^{k+1}$, substitute the rightmost column of S_k and Y_k, i.e. columns s^k and y^k, with $s_{\text{accel}} = x_{\text{accel}}^{k+1} - x^k$ and $y_{\text{accel}} = F(x_{\text{accel}}^{k+1}) - F(x^k)$, respectively, and set $r_{\max} \leftarrow \max\{r_{\max}, \text{rank}(Y_k)\}$.

Step 2.6. If $\text{rank}(Y_k) = 0$, then execute Steps 2.6.1–2.6.3.

Step 2.6.1. Redefine matrices S_k and Y_k as matrices with zero columns.

Step 2.6.2. Execute Steps 2.6.2.1 $p - 1$ times.

Step 2.6.2.1. Add $s_{\text{extra}} = x_{\text{extra}} - x_{\text{trial}}^{k+1}$ and $y_{\text{extra}} = F(x_{\text{extra}}) - F(x_{\text{trial}}^{k+1})$ as rightmost column of S_k and Y_k, respectively, where $x_{\text{extra}} = x^k + h_{\text{large}}\, e_\ell$. Set $r_{\max} \leftarrow \max\{r_{\max}, \text{rank}(Y_k)\}$ and $\ell \leftarrow \text{mod}(\ell, n) + 1$.

Step 2.6.3. Add $s^k = x_{\text{trial}}^{k+1} - x^k$ and $y^k = F(x_{\text{trial}}^{k+1}) - F(x^k)$ as rightmost column of S_k and Y_k, respectively; and set $r_{\max} \leftarrow \max\{r_{\max}, \text{rank}(Y_k)\}$.

Step 2.7. Compute the minimum-norm least-squares solution $\bar{\omega}$ to the linear system $Y_k\omega = F(x_{\text{trial}}^{k+1})$ and define $x_{\text{accel}}^{k+1} = x_{\text{trial}}^{k+1} - S_k\bar{\omega}$.

Step 2.8. If $x_{\text{accel}}^{k+1} \neq x^k$, $\|x_{\text{accel}}^{k+1}\| \leq 10\max\{1, \|x^k\|\}$, and $\|F(x_{\text{accel}}^{k+1})\| < \|F(x_{\text{trial}}^{k+1})\|$, then redefine $x_{\text{trial}}^{k+1} = x_{\text{accel}}^{k+1}$ and substitute the rightmost column of S_k and Y_k, i.e. columns s^k and y^k, with $s_{\text{accel}} = x_{\text{accel}}^{k+1} - x^k$ and $y_{\text{accel}} = F(x_{\text{accel}}^{k+1}) - F(x^k)$, respectively, and set $r_{\max} \leftarrow \max\{r_{\max}, \text{rank}(Y_k)\}$.

Step 3. Define $x^{k+1} = x_{\text{trial}}^{k+1}$.

In general, Step 2.6 is not executed. If iteration k is such that Step 2.6 was not executed in the previous p iterations, then matrices S_k and Y_k correspond to removing the leftmost column from and adding a new rightmost column to S_{k-1} and Y_{k-1}, respectively. The leftmost column is not removed if the maximum number of columns p was not reached yet. When $k = 0$, Y_k is a single-column matrix for which matrices P, Q, and R of the rank-revealing QR decomposition $Y_kP = QR$ can be trivially computed. For $k > 0$, the QR decomposition of Y_k can be obtained with time complexity $O(n\,p)$ by updating, via Givens rotations, the QR decomposition of Y_{k-1}. Moreover, by using the QR decomposition of Y_k, the minimum-norm least-squares solution $\bar{\omega}$ can be computed with time complexity $O(n + p^2)$ if Y_k is full-rank and with time complexity $O(n + p^3)$ in the rank-deficient case. Summing up, by assuming that $p \ll n$ and that p does not depend on n, iterations of Algorithm 5.1 can be implemented with time complexity $O(n)$. The space complexity is $O(n\,p + p^2)$ and it is related to the fact that we must save matrix $S_k \in \mathbb{R}^{n\times p}$ and matrices $Q \in \mathbb{R}^{n\times p}$ and $R \in \mathbb{R}^{p\times p}$ of the QR decomposition of Y_k. Of course, the permutation matrix P can be saved in an array of size p. Thus, the space complexity of Algorithm 5.1 is also $O(n)$ under the same assumptions. When Step 2.6 is executed, the QR decomposition of Y_k must be computed from scratch, with time complexity $O(n\,p^2)$.

6 Numerical experiments

We implemented Algorithm 2.1 and Algorithm 5.1 (the practical version of Algorithm 4.1) in Fortran 90. In the numerical experiments, we considered the standard values $\gamma = 10^{-4}$, $\tau_{\min} = 0.1$, $\tau_{\max} = 0.5$, $M = 10$, $\sigma_{\min} = \sqrt{\epsilon}$, $\sigma_{\max} = 1/\sqrt{\epsilon}$, and $\eta_k = 2^{-k}\min\{\frac{1}{2}\|F(x^0)\|, \sqrt{\|F(x^0)\|}\}$, where $\epsilon \approx 10^{-16}$ is the machine precision. More crucial to the method performance are the choices of h_{init}, h_{small}, h_{large}, and p, whose values are mentioned below. All tests were conducted on a computer with a 3.4 GHz Intel Core i5 processor and 8GB 1600 MHz DDR3 RAM memory, running macOS Mojave (version 10.14.6). Code was compiled by the GFortran compiler of GCC (version 8.2.0) with the -O3 optimization directive enabled.

6.1 2D and 3D Bratu problem

In a first experiment, we considered 2D and 3D versions of the Bratu problem

$$-\Delta u + \theta e^u = \phi(\bar{u}) \text{ in } \Omega \tag{72}$$

with boundary conditions $u = \bar{u}$ on $\partial\Omega$. In the 2D case, $\Omega = [0,1]^2$ and, following [28], we set $\bar{u} = 10u_1u_2(1-u_1)(1-u_2)e^{u_1^{4.5}}$; while in the 3D case, $\Omega = [0,1]^3$ and $\bar{u} = 10u_1u_2u_3(1-u_1)(1-u_2)(1-u_3)e^{u_1^{4.5}}$. In both cases, $\phi(u) = -\Delta u + \theta e^u$; so the problem has $\bar{u}$ as known solution. Considering n_p discretization points in each dimension and approximating

$$\Delta u(x) \approx \frac{u(x \pm he_1) + u(x \pm he_2) - 4u(x)}{h^2}$$

and

$$\Delta u(x) \approx \frac{u(x \pm he_1) + u(x \pm he_2) + u(x \pm he_3) - 6u(x)}{h^2},$$

where $h = 1/(n_p-1)$ and e_i is the ith canonical vector in the corresponding space ($\mathbb{R}^2$ or $\mathbb{R}^3$), we obtain nonlinear systems of equations with $n = (n_p - 2)^2$ and $n = (n_p - 2)^3$ variables in the 2D and 3D cases, respectively. Starting from $u = 0$, fixing $\theta = -100$, and varying $n_p \in \{100, 125, \ldots, 400\}$ and $n_p \in \{10, 15, \ldots, 70\}$ in the 2D and 3D cases respectively, we run NITSOL (file NITSOL,11-1-05.TAR.GZ downloaded from `https://users.wpi.edu/~walker/NITSOL/` on October 26th, 2020) and the method proposed in the present work, which will be called Accelerated DF-SANE from now on. As stopping criterion, we considered (65) with $\varepsilon = 10^{-6}\sqrt{n}$. For NITSOL, we use all its default parameters, except the maximum number of (nonlinear) iterations, which was increased in order to avoid premature stops. By default, NITSOL corresponds to an Inexact-Newton method in which Newtonian systems are solved with GMRES (maximum Krylov subspace dimension equal to 20), approximating the Jacobian-vector products by finite-differences. For Accelerated DF-SANE, based on preliminary experiments, we set $h_{\text{small}} = 10^{-4}$, $h_{\text{large}} = 0.1$, and $h_{\text{init}} = 0.01$ in the 2D case and $h_{\text{small}} = h_{\text{large}} = 0.1$ and $h_{\text{init}} = 1$ in the 3D case. In both cases, we considered $p = 5$.

Tables 1 and 2 show the results. In the tables, $\#\text{it}_1$ corresponds to the nonlinear iterations (outer iterations) of NITSOL; while $\#\text{it}_2$ corresponds to its linear iterations (inner or GMRES iterations). For both methods, "fcnt" stands for number of evaluations of F, "Time" stands for CPU time in seconds, and "SC" stands for "Stopping criterion". Remaining columns are self-explanatory. In the case of NITSOL, stopping criterion equal to 0 means success; while 6 means "failure to reach an acceptable step through backtracking". Accelerated DF-SANE satisfied the stopping criterion related to success in all problems. Regarding the 2D problems, it should be first noted that NITSOL failed in satisfying the stopping criterion for $n_p \geq 225$. Accelerated DF-SANE is faster than NITSOL in all problems that both methods solved. In the larger problem that both methods solved, Accelerated DF-SANE is around thirty times faster than NITSOL. Regarding the 3D problems (Table 2), both methods satisfied the stopping criterion related to success in all problems. Accelerated DF-SANE is faster than NITSOL in all problems in the table. In the larger problem in the table ($n_p = 70$), Accelerated DF-SANE is more than twenty times faster than NITSOL.

n_p	n	NITSOL (Newton-GMRES)						Accelerated DF-SANE			
		SC	$\|F(x_*)\|_2$	$\#\text{it}_1$	$\#\text{it}_2$	fcnt	Time	$\|F(x_*)\|_2$	#it	fcnt	Time
100	9,604	0	9.7e−05	220	197,732	197,953	39.63	9.8e−05	5,219	10,688	4.87
125	15,129	0	1.2e−04	631	473,616	474,248	151.88	1.2e−04	2,612	5,489	3.80
150	21,904	0	1.5e−04	379	315,338	315,718	165.07	1.5e−04	2,946	6,007	6.59
175	29,929	0	1.7e−04	401	331,380	331,782	265.53	1.7e−04	4,475	10,007	17.46
200	39,204	0	2.0e−04	4,421	846,957	851,385	983.97	2.0e−04	6,775	14,385	32.24
225	49,729	6	3.3e+01	3,944	173,310	177,283	261.03	2.2e−04	4,328	8,927	26.72
250	61,504	6	4.1e+01	105	46,897	47,019	82.69	2.5e−04	12,661	26,353	94.47
275	74,529	6	5.3e+01	1,411	94,194	95,635	224.74	2.7e−04	8,809	19,583	88.07
300	88,804	6	7.2e+01	1,341	104,735	106,112	317.71	3.0e−04	15,858	34,194	190.49
325	104,329	6	9.6e+01	2,430	194,640	197,126	705.53	3.2e−04	10,791	23,403	160.02
350	121,104	6	1.4e+02	195	82,940	83,151	319.00	3.5e−04	10,805	25,915	176.52
375	139,129	6	1.7e+02	1,127	121,100	122,243	649.94	3.7e−04	16,335	38,648	331.03
400	158,404	6	2.2e+02	2,936	144,460	147,415	851.67	4.0e−04	21,106	55,901	515.65

Table 1: Performances of NITSOL and Accelerated DF-SANE in the 2D Bratu problem.

The difficulty of the Bratu problem varies with the value of θ, that may be positive or negative. For the formulation given in (72), negative values of θ correspond to more difficult problem. Therefore, results in Tables 1 and 2 raise the question of how the performances of the methods compare to each other for different values of θ. So, we arbitrarily considered the 3D Bratu problem with $n_p = 40$, which is affordable for both methods and varied $\theta \in [-100, 10]$. Figure 1 shows the results of applying NITSOL and Accelerated DF-SANE. The graphic shows that for $\theta \geq -20$ both methods use less than a second and NITSOL outperforms Accelerated DF-SANE. On the other hand, in the most difficult problems (i.e. $\theta < -20$), where up to one thousand seconds are required, Accelerated DF-SANE outperforms NITSOL by approximately an order of magnitude. (Note that the y-axis is in logarithmic scale.) Considering the number of functional evaluations as a performance metric, instead of the CPU time, results are mostly the same.

Another relevant question is how the performance of Accelerated DF-SANE varies as a function of its parameter p. So, considering again the 3D Bratu problem with $\theta = -100$ and $n_p = 40$, we ran Accelerated DF-SANE with $p \in \{3, 4, \ldots, 17\}$. Figure 2 shows the results. The figure on the left shows that the number of

n_p	n	NITSOL (Newton-GMRES)					Accelerated DF-SANE			
		$\|F(x_*)\|_2$	$\#\text{it}_1$	$\#\text{it}_2$	fcnt	Time	$\|F(x_*)\|_2$	#it	fcnt	Time
10	512	1.7e−05	5	213	219	0.00	2.1e−05	126	308	0.00
15	2,197	3.7e−05	7	1,621	1,629	0.07	4.7e−05	223	662	0.05
20	5,832	6.1e−05	11	7,157	7,169	0.97	7.2e−05	1,221	4,271	0.92
25	12,167	8.8e−05	19	15,893	15,913	4.62	1.1e−04	529	1,840	0.78
30	21,952	1.2e−04	25	21,991	22,017	12.72	1.4e−04	812	3,012	2.29
35	35,937	1.6e−04	38	34,935	34,974	36.31	1.9e−04	1,210	4,530	6.37
40	54,872	2.2e−04	70	66,115	66,186	118.56	2.3e−04	1,109	4,379	9.35
45	79,507	2.8e−04	147	137,626	137,774	386.09	2.8e−04	1,315	5,444	20.06
50	110,592	3.3e−04	217	199,042	199,260	759.61	3.3e−04	1,574	6,501	30.38
55	148,877	3.8e−04	357	312,023	312,381	1736.95	3.8e−04	1,706	7,254	53.10
60	195,112	4.4e−04	504	418,201	418,706	3074.25	4.4e−04	1,821	8,019	70.29
65	250,047	5.0e−04	988	691,192	692,181	6715.18	4.9e−04	2,061	9,379	109.39
70	314,432	5.6e−04	609	486,651	487,261	5735.90	5.6e−04	1,836	8,431	118.13

Table 2: Performances of NITSOL and Accelerated DF-SANE in the 3D Bratu problem.

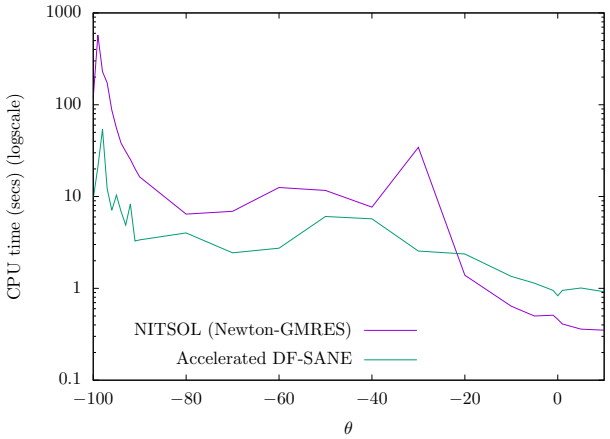

Figure 1: This figure shows the CPU time (in seconds) used by NITSOL and Accelerated DF-SANE to solve the 3D Bratu problem with $n_p = 40$ and $\theta \in [-100, 10]$.

iterations of Accelerated DF-SANE is nearly constant as a function of p; while the figure on the right shows, as expected, that, the larger p, the larger the CPU time per iteration. Note that for the considered values of p, the variation on the number of iterations is up to 20%; while the variation in CPU time is up to 300%.

Another relevant question regarding Accelerated DF-SANE is how it compares against DF-SANE, i.e. the original method without the acceleration being proposed in the present work. We were unable to run DF-SANE in the set of problems considered in Tables 1 and 2, because DF-SANE is unable to reach the stopping criterion (65) with $\varepsilon = 10^{-6}\sqrt{n}$ within an affordable time. As an alternative, we considered the four instances of the 3D Bratu problem with $n_p \in \{40, 70\}$ and $\theta \in \{-100, 10\}$. In the two instances with $\theta = 10$, DF-SANE was able to satisfy the imposed stopping criterion within an affordable time. This result was in fact expected because for this value of θ the Bratu problem resembles the minimization of a convex quadratic function; and Barzilai-Borwien or spectral step based methods are expected to perform especially well in this case. In the two instances with $\theta = -100$, we first run Accelerated DF-SANE and then we run DF-SANE using as CPU time limit the time consumed by Accelerated DF-SANE. Figure 3 shows the results. It is very clear from the four graphics that the acceleration process is very efficient in its purpose of accelerating DF-SANE. (In the cases with $\theta = -100$, with an excruciatingly slow progress, DF-SANE is far from reaching convergence after a couple of hours of CPU time.)

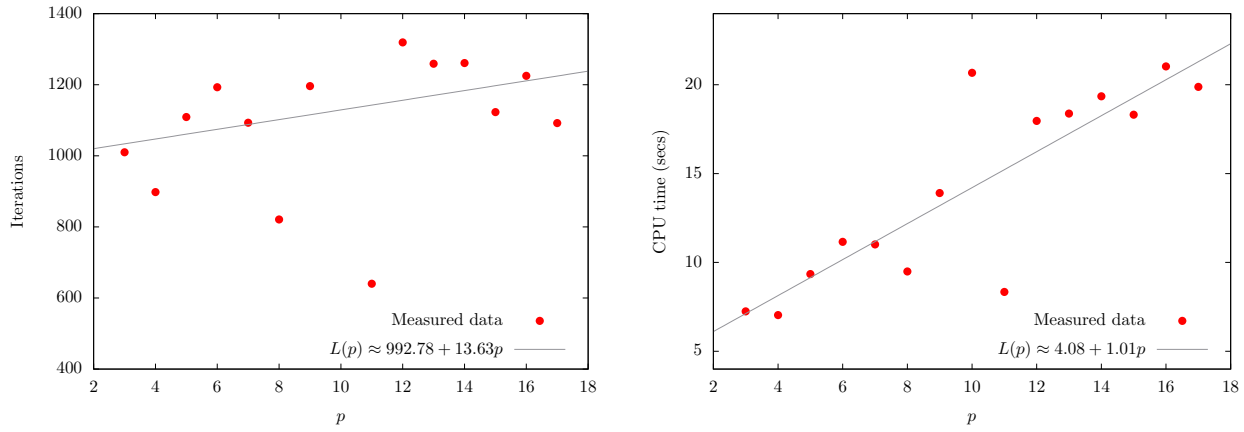

Figure 2: Performance of Accelerated DF-SANE in the 3D Bratu problem with $n_p = 40$ and $\theta = -100$ varying the number p of columns in matrices S_k and Y_k.

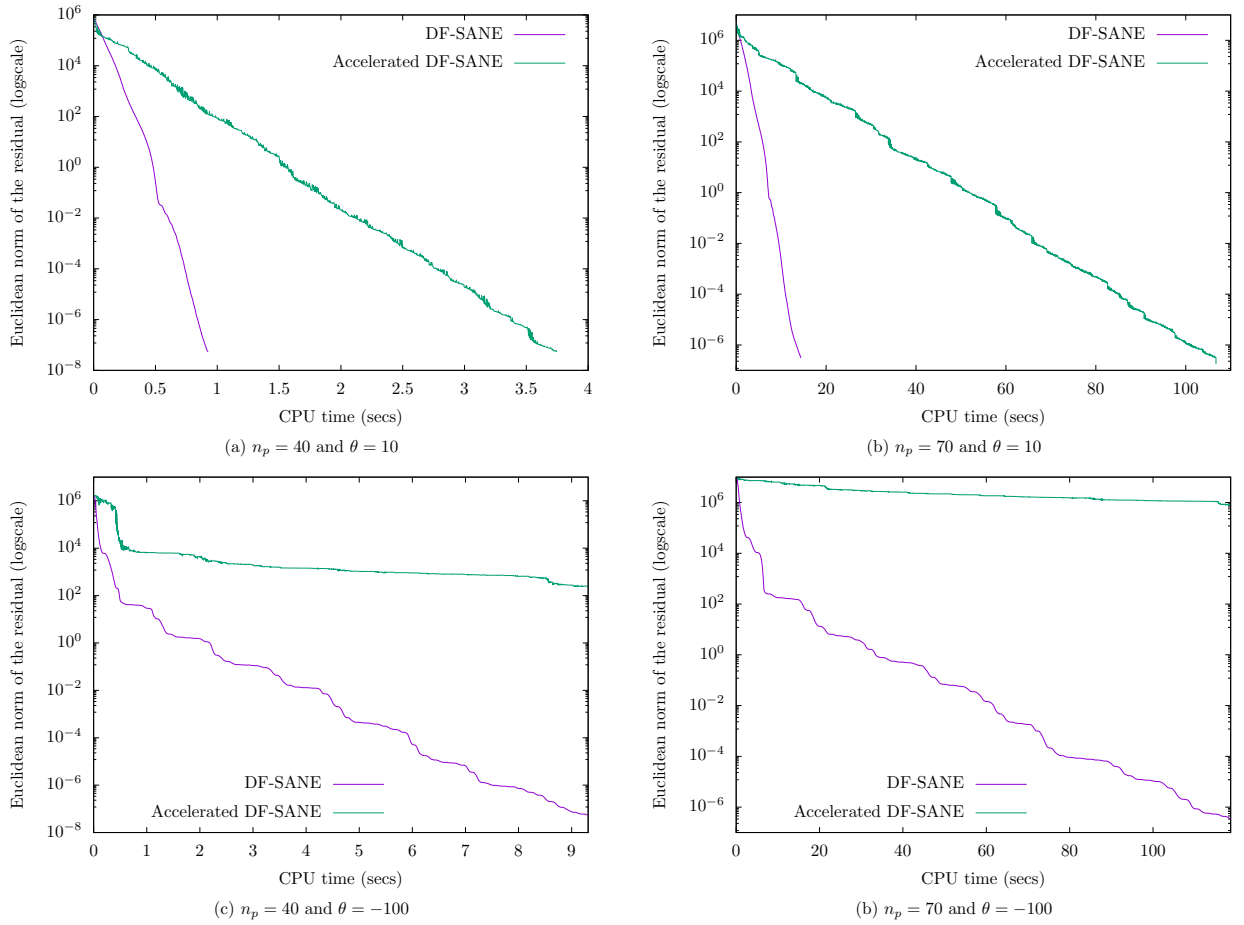

Figure 3: Performance of DF-SANE and Accelerated DF-SANE in four instances of the 3D Bratu problem with $\theta \in \{-100, 10\}$ and $n_p \in \{40, 70\}$.

6.2 Modified Bratu, Driven Cavity, and Flow in a Porous Media problems

In a second set of experiments, we considered three PDE-based problems described in [41]. Fortran implementations of these problems are included as examples of usage in the NITSOL distribution. The three problems are discretization of 2D PDE-based systems of nonlinear equations. All parameters of the problems were set as suggested in [41]. We varied the value of n_p for the three problems trying to illustrate the behavior of the methods and solving as large as possible problems with both methods within an affordable time. NITSOL was run with all its default parameters, as in the previous section; while Accelerated DF-SANE was run with the same parameters already reported for the 2D Bratu problem in the previous section. Tables 3–5 show the results. Table 3 shows that NITSOL failed in satisfying the stopping criterion for $n_p \geq 135$ in the Driven Cavity problem. Accelerated DF-SANE was faster than NITSOL in all instances that both methods solved. In the largest instance that both methods solved, Accelerated DF-SANE is more than thirty times faster than NITSOL. Table 4 shows that NITSOL is faster than Accelerated DF-SANE in the smaller instances of the Generalized 2D Bratu problem (n_p up to 750). In the largest instance in the table ($n_p = 1,500$), Accelerated DF-SANE takes around half of the time required by NITSOL. Finally, Table 5 shows that NITSOL is faster than Accelerated DF-SANE in the smaller instances (n_p up to 400) of the Flow in a Porous Media problem. In the largest instance in the table ($n_p = 1,500$), Accelerated DF-SANE takes around half of the time required by NITSOL.

n_p	n	NITSOL (Newton-GMRES)						Accelerated DF-SANE			
		SC	$\|F(x_*)\|_2$	#it$_1$	#it$_2$	fcnt	Time	$\|F(x_*)\|_2$	#it	fcnt	Time
10	100	0	7.7e−06	5	194	200	0.00	9.6e−06	96	200	0.00
15	225	0	1.0e−05	6	988	995	0.00	1.4e−05	190	392	0.00
20	400	0	1.6e−05	6	2,739	2,746	0.02	1.9e−05	278	582	0.01
25	625	0	2.0e−05	10	6,273	6,284	0.09	2.4e−05	473	1,044	0.03
30	900	0	2.4e−05	15	11,939	11,955	0.24	2.9e−05	663	1,519	0.06
35	1,225	0	3.1e−05	23	20,058	20,082	0.57	3.4e−05	889	2,118	0.11
40	1,600	0	3.8e−05	36	32,819	32,856	1.19	3.9e−05	1,141	2,954	0.20
45	2,025	0	4.4e−05	56	52,200	52,257	2.52	4.4e−05	1,431	3,706	0.33
50	2,500	0	5.0e−05	80	74,981	75,062	4.68	5.0e−05	1,747	4,653	0.50
55	3,025	0	5.2e−05	114	106,481	106,596	8.27	5.5e−05	2,093	6,010	0.75
60	3,600	0	5.8e−05	154	142,222	142,377	13.09	6.0e−05	2,470	7,346	1.06
65	4,225	0	6.3e−05	217	195,849	196,067	21.29	6.5e−05	2,880	8,824	1.48
70	4,900	0	6.8e−05	278	244,149	244,428	30.60	7.0e−05	3,297	10,533	1.99
75	5,625	0	7.4e−05	392	328,630	329,023	48.01	7.5e−05	3,781	12,557	2.74
80	6,400	0	7.8e−05	568	442,630	443,199	72.66	8.0e−05	4,274	14,564	3.48
85	7,225	0	8.4e−05	790	563,030	563,821	104.84	8.5e−05	4,811	16,905	4.47
90	8,100	0	9.0e−05	1,135	706,031	707,167	147.14	9.0e−05	5,369	19,423	5.67
95	9,025	0	9.5e−05	1,661	856,631	858,293	200.43	9.5e−05	5,949	22,052	7.10
100	10,000	0	1.0e−04	3,051	1040,211	1,043,263	269.07	1.0e−04	6,563	24,901	8.76
105	11,025	0	1.0e−04	13,222	1233,251	1,246,474	357.23	1.0e−04	7,208	27,977	10.74
110	12,100	0	1.1e−04	20,017	1415,812	1,435,830	452.47	1.1e−04	7,593	30,394	12.62
115	13,225	0	1.1e−04	18,717	1290,772	1,309,495	458.66	1.1e−04	8,543	34,375	15.81
120	14,400	0	1.2e−04	18,350	1461,992	1,480,349	568.50	1.2e−04	9,265	37,904	18.57
125	15,625	0	1.2e−04	26,341	1628,992	1,655,343	696.52	1.2e−04	10,036	41,724	22.07
130	16,900	0	1.3e−04	13,114	1895,392	1,908,981	903.98	1.3e−04	10,824	45,718	26.53
135	18,225	6	1.0e−01	6,604	370,973	377,706	190.84	1.3e−04	11,639	49,969	30.57
140	19,600	6	1.3e−01	6,616	302,593	309,288	171.16	1.4e−04	12,406	53,796	35.35
145	21,025	6	1.5e−01	6,661	278,673	285,397	170.61	1.4e−04	12,814	56,423	40.04
150	22,500	6	1.7e−01	7,048	269,053	276,151	178.26	1.5e−04	13,741	61,042	47.26
155	24,025	6	1.9e−01	6,531	273,073	279,666	197.21	1.5e−04	14,651	65,802	54.88
160	25,600	6	2.0e−01	6,204	261,354	267,616	203.83	1.6e−04	15,509	70,613	63.10
165	27,225	6	2.1e−01	6,897	265,594	272,543	221.86	1.6e−04	15,310	69,910	64.23
170	28,900	6	2.2e−01	5,214	286,714	292,023	257.46	1.7e−04	17,338	80,401	79.21
175	30,625	6	2.6e−01	5,618	255,294	260,978	244.98	1.7e−04	18,015	83,984	87.44
180	32,400	6	2.9e−01	5,260	237,754	243,072	242.00	1.8e−04	18,510	86,308	96.99
185	34,225	6	3.1e−01	5,460	233,394	238,906	253.32	1.8e−04	19,457	91,745	114.16
190	36,100	6	3.4e−01	5,741	217,814	223,590	259.29	1.9e−04	20,846	99,200	132.56
195	38,025	6	3.5e−01	5,539	226,995	232,580	280.39	1.9e−04	20,135	97,440	138.37
200	40,000	6	3.5e−01	4,969	241,335	246,370	311.54	2.0e−04	21,875	107,288	146.27

Table 3: Performances of NITSOL and Accelerated DF-SANE in the "Driven Cavity Problem".

n_p	n	NITSOL (Newton-GMRES)					Accelerated DF-SANE			
		$\|F(x_*)\|_2$	#it$_1$	#it$_2$	fcnt	Time	$\|F(x_*)\|_2$	#it	fcnt	Time
100	10,000	7.9e−05	4	212	217	0.03	9.9e−05	174	349	0.15
150	22,500	1.1e−04	4	321	326	0.14	1.5e−04	257	515	0.54
200	40,000	1.6e−04	4	464	469	0.40	2.0e−04	335	671	1.42
250	62,500	1.9e−04	4	655	660	0.98	2.5e−04	405	811	2.77
300	90,000	2.4e−04	4	741	746	1.59	3.0e−04	473	947	4.95
350	122,500	2.8e−04	4	1,054	1,059	3.15	3.4e−04	543	1,087	7.96
400	160,000	3.2e−04	4	1,315	1,320	5.17	4.0e−04	613	1,227	12.47
450	202,500	3.4e−04	4	1,518	1,523	7.54	4.3e−04	684	1,369	17.74
500	250,000	4.1e−04	3	1,607	1,611	10.02	4.9e−04	754	1,509	25.21
550	302,500	4.5e−04	3	2,447	2,451	19.94	5.5e−04	823	1,647	34.23
600	360,000	5.0e−04	3	2,793	2,797	29.30	5.9e−04	893	1,787	43.80
650	422,500	5.2e−04	4	3,502	3,507	45.42	6.3e−04	962	1,925	56.59
700	490,000	5.6e−04	4	3,691	3,696	59.89	7.0e−04	1,030	2,061	71.45
750	562,500	6.2e−04	4	3,702	3,707	72.01	7.4e−04	1,098	2,197	91.20
800	640,000	6.4e−04	5	4,871	4,877	113.52	8.0e−04	1,164	2,329	111.31
850	722,500	6.9e−04	5	4,826	4,832	132.01	8.5e−04	1,229	2,459	132.07
900	810,000	7.2e−04	6	5,965	5,972	187.16	9.0e−04	1,292	2,585	157.64
950	902,500	7.6e−04	6	5,695	5,702	200.12	9.5e−04	1,353	2,707	184.62
1,000	1,000,000	8.0e−04	8	7,386	7,395	292.89	1.0e−03	1,408	2,817	212.07
1,050	1,102,500	8.4e−04	7	6,901	6,909	304.10	1.0e−03	1,465	2,931	243.72
1,100	1,210,000	8.8e−04	9	8,399	8,409	405.89	1.1e−03	1,528	3,057	277.72
1,150	1,322,500	9.2e−04	8	7,977	7,986	423.56	1.1e−03	1,592	3,185	317.39
1,200	1,440,000	1.0e−03	10	9,980	9,991	578.89	1.2e−03	1,656	3,313	358.84
1,250	1,562,500	1.0e−03	10	9,678	9,689	608.95	1.2e−03	1,720	3,441	404.81
1,300	1,690,000	1.3e−03	11	10,960	10,972	747.45	1.3e−03	1,783	3,567	451.14
1,350	1,822,500	1.1e−03	11	10,935	10,947	803.39	1.3e−03	1,846	3,693	514.03
1,400	1,960,000	1.1e−03	13	12,932	12,946	1021.67	1.4e−03	1,909	3,819	618.58
1,450	2,102,500	1.2e−03	13	12,636	12,650	1074.93	1.4e−03	1,971	3,943	630.03
1,500	2,250,000	1.4e−03	14	13,920	13,935	1267.96	1.5e−03	2,032	4,065	701.65

Table 4: Performances of NITSOL and Accelerated DF-SANE in "Generalized 2D Bratu Problem".

7 Conclusions

The Sequential Secant Method, which is the most obvious multidimensional generalization of the secant method for solving nonlinear equations, seems to have been introduced by Wolfe in 1959 [49]. See also [2]. This method has been analyzed in classical books and papers [40, 46, 27] where, under suitable conditions, local convergence with R-order equal to the unique solution of $t^{n+1} - t^n - 1 = 0$ was proved. Given the consecutive iterates $x^{k-n}, \ldots, x^{k-1}, x^k$, the Sequential Secant Method computes

$$x^{k+1} = x^k - (s^{k-n}, \ldots, s^{k-1})(y^{k-1}, \ldots, y^{k-1})^{-1} F(x^k). \tag{73}$$

This iteration is well defined if the matrix $(y^{k-n}, \ldots, y^{k-1})$ is nonsingular. Moreover the good local convergence properties need uniformly linear independence of the increments $s^{k-n}, \ldots, s^{k-1}$. The method has been updated in several ways in order to fix these drawbacks while maintaining its convergence properties. The natural limited memory version of (73) is defined by

$$x^{k+1} = x^k - (s^{k-p}, \ldots, s^{k-1})(y^{k-p}, \ldots, y^{k-1})^{\dagger} F(x^k), \tag{74}$$

where $1 \leq p \ll n$. This formula is inconvenient for solving nonlinear systems because s^k necessarily belongs to the subspace generated by $s^{k-p}, \ldots, s^{k-1}$ which implies that x^{k+j} is in the affine subspace determined by $x^{k-p}, \ldots, x^{k-1}, x^k$, for all j, and, so, convergence to a solution cannot occur unless the solution belongs to the same affine subspace. This is the reason why, in the present paper, we do not use the method defined by (74). Instead, we compute x^{k+1}_{trial} using the sequential residual approach, we define $s^k = x^{k+1}_{\text{trial}} - x^k$, $y^k =$

n_p	n	NITSOL (Newton-GMRES)					Accelerated DF-SANE			
		$\|F(x_*)\|_2$	#it$_1$	#it$_2$	fcnt	Time	$\|F(x_*)\|_2$	#it	fcnt	Time
100	10,000	7.9e−05	11	1,806	1,818	0.22	1.0e−04	684	1,376	0.52
150	22,500	1.2e−04	10	2,666	2,677	0.92	1.5e−04	845	1,698	1.55
200	40,000	1.7e−04	10	3,634	3,645	2.48	2.0e−04	1,255	2,521	4.68
250	62,500	2.4e−04	12	5,817	5,830	6.83	2.5e−04	1,378	2,798	8.43
300	90,000	2.6e−04	13	7,272	7,286	12.84	3.0e−04	1,595	3,212	14.81
350	122,500	3.4e−04	13	7,890	7,904	19.29	3.5e−04	1,634	3,310	21.28
400	160,000	3.9e−04	13	8,655	8,669	28.19	4.0e−04	1,616	3,349	29.48
450	202,500	4.2e−04	15	10,911	10,927	45.61	4.5e−04	1,723	3,642	40.69
500	250,000	4.7e−04	15	10,999	11,015	61.56	5.0e−04	1,858	3,916	55.15
550	302,500	4.7e−04	18	14,065	14,084	103.30	5.5e−04	1,954	3,997	72.60
600	360,000	5.3e−04	18	14,140	14,159	132.62	6.0e−04	2,043	4,145	90.35
650	422,500	6.3e−04	17	13,152	13,170	151.83	6.5e−04	2,171	4,508	116.74
700	490,000	6.4e−04	20	16,127	16,148	229.96	7.0e−04	2,271	4,874	144.36
750	562,500	7.3e−04	18	14,164	14,183	243.82	7.5e−04	2,377	5,263	180.96
800	640,000	7.2e−04	20	16,127	16,148	344.09	8.0e−04	2,502	5,630	220.13
850	722,500	7.6e−04	20	16,147	16,168	396.81	8.5e−04	2,629	6,037	263.11
900	810,000	8.8e−04	20	16,129	16,150	431.85	9.0e−04	2,750	6,398	309.54
950	902,500	9.3e−04	21	17,094	17,116	521.95	9.5e−04	2,909	6,856	367.30
1,000	1,000,000	9.7e−04	20	16,144	16,165	549.77	1.0e−03	2,986	7,227	419.13
1,050	1,102,500	9.9e−04	22	18,098	18,121	679.61	1.0e−03	3,048	7,581	489.69
1,100	1,210,000	1.0e−03	22	18,111	18,134	753.17	1.1e−03	3,125	8,017	548.80
1,150	1,322,500	1.1e−03	22	18,026	18,049	892.35	1.1e−03	3,264	8,448	631.76
1,200	1,440,000	1.2e−03	23	19,006	19,030	1056.51	1.2e−03	3,434	9,051	748.79
1,250	1,562,500	1.1e−03	24	19,983	20,008	1183.49	1.2e−03	3,371	9,061	815.04
1,300	1,690,000	1.3e−03	24	20,010	20,035	1207.67	1.3e−03	3,377	9,249	891.39
1,350	1,822,500	1.3e−03	26	21,969	21,996	1440.00	1.3e−03	3,302	8,892	875.23
1,400	1,960,000	1.3e−03	26	21,947	21,974	1538.56	1.4e−03	3,234	8,477	893.25
1,450	2,102,500	1.4e−03	27	22,938	22,966	2118.16	1.4e−03	3,284	8,566	978.04
1,500	2,250,000	1.4e−03	27	22,934	22,962	2108.73	1.5e−03	3,314	8,569	1047.10

Table 5: Performances of NITSOL and Accelerated DF-SANE in "Flow in a Porous Media Problem".

$F(x_{\text{trial}}^{k+1}) - F(x^k)$,

$$x_{\text{accel}}^{k+1} = x_{\text{trial}}^{k+1} - (s^{k-p+1}, \ldots, s^{k-1}, s^k)(y^{k-p+1}, \ldots, y^{k-1}, y^k)^\dagger F(x_{\text{trial}}^{k+1}) \tag{75}$$

and we choose x^{k+1} as the best of the trials x_{trial}^{k+1} and x_{accel}^{k+1}. In this way, we preserve the good properties of the Sequential Secant Method associated with the good global behavior of Sequential Residual approaches. The freedom on the choice of the residual step favors the employment of preconditioners when they are available.

The most popular methods for solving nonlinear systems of equations in which the application of Newton's method is impossible or extremely expensive are based on the Inexact Newton approach with Krylov-subspace methods (as GMRES) for solving approximately the Newtonian linear systems at each iteration. These methods have a long tradition and, very likely, they deserve to be the preferred ones by practitioners in the numerical PDE community. Nevertheless, in this paper we showed that, for some very large interesting problems, an approach based in sequential-secant-like accelerations of a residual method is more efficient than a standard implementation of Newton-GMRES with its default algorithmic parameters. This indicates that those problems possess characteristics that favor the application of the secant paradigm over the inexact Newton one. Of course, the opposite situation probably occurs in many cases. This means, therefore, that efficiency for solving practical problems will be increased if practitioners have easy access to both types of methods. The codes used in this paper, that make it possible the reproducibility of all the experiments, as well as the instructions for using our algorithm may be found in `http://www.ime.usp.br/~egbirgin/sources/accelerated-df-sane/`.

Future work will include the employment of the new methods to the acceleration of KKT solvers that are necessary in the Augmented Lagrangian approach for constrained optimization [5].

References

[1] D. G. Anderson, Iterative procedures for nonlinear integral equations, *Journal of the Association for Computing Machinery* 12, pp. 547–560, 1965.

[2] J. G. P. Barnes, An algorithm for solving nonlinear equations based on the secant method, *Computer Journal* 8, pp. 66–72, 1965.

[3] J. Barzilai and J. M. Borwein, Two point step size gradient methods, *IMA Journal of Numerical Analysis* 8, pp. 141–148, 1988.

[4] W. Bian, X. Chen and C. T. Kelley, Anderson acceleration for a class of nonsmooth fixed-point problems, *SIAM Journal on Scientific Computing*, to appear.

[5] E. G. Birgin and J. M. Martínez, *Practical Augmented Lagrangian Methods for Constrained Optimization*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2014.

[6] E. G. Birgin, J. M. Martínez, and M. Raydan, Nonmonotone spectral projected gradient methods on convex sets, *SIAM Journal on Optimization* 10, pp. 1196–1211, 2000.

[7] C. Brezinski, Convergence acceleration during the 20th century, *Journal of Computational and Applied Mathematics* 122, pp. 1–21, 2000.

[8] C. Brezinski and M. Redivo-Zaglia, *Extrapolation Methods Theory and Practice*, North– Holland, Amsterdam, 1991.

[9] C. Brezinski, M. Redivo-Zaglia, and Y. Saad, Shanks sequence transformations and Anderson acceleration, *SIAM Review* 60, pp. 646–669, 2018.

[10] P. N. Brown, H. F. Walker, R. Wasyk, and C. S. Woodward, On using approximate finite-differences in matrix-free Newton-Krylov methods, *SIAM Journal on Numerical Analysis* 46, 1892–1911, 2008.

[11] O. Burdakov and A. Kamandi, Multipoint secant and interpolation methods with nonmonotone line search for solving systems of nonlinear equations, *Applied Mathematics and Computation* 338, pp. 421-431, 2018.

[12] X. Chen and C. T. Kelley, Convergence of the EDIIS algorithm for nonlinear equations, *SIAM Journal on Scientific Computing* 4, A365–A379, 2019.

[13] J. Degroote, K.-J. Bathe, and J. Vierendeels, Performance of a new partitioned procedure versus a monolithic procedure in fluid-structure interaction, *Computers and Structures* 97, pp. 793–801, 2009.

[14] R. S. Dembo, S. C. Eisenstat, and T. S. Steihaug, Inexact Newton methods, *SIAM Journal on Numerical Analysis* 19, pp. 400–408, 1982.

[15] J. E. Dennis and J. J. Moré, Quasi-Newton methods: Motivation and Theory, *SIAM Review* 19, pp. 46–89, 1977.

[16] J. E. Dennis Jr. and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Society for Industrial and Applied Mathematics, Philadelphia, 1996.

[17] S. C. Eisenstat and H. F. Walker, Globally convergent Inexact Newton methods, *SIAM Journal on Optimization* 4, pp. 393–422, 1994.

[18] S. C. Eisenstat and H. F. Walker, Choosing the forcing terms in an Inexact Newton method, *SIAM Journal on Scientific Computing* 17, pp. 16–32, 1996.

[19] H. Fang and Y. Saad, Two classes of multisecant methods for nonlinear acceleration, *Numerical Linear Algebra and Applications* 16, pp. 197–221, 2009.

[20] L. Frérot, M. Bonnet, J.-F. Molinari, and G. Anciaux, A Fourier-accelerated volume integral method for elastoplastic contact, *Computer Methods in Applied Mechanics and Engineering* 351, pp. 951–976, 2019.

[21] L. Frérot, G. Anciaux, and J.-F. Molinari, Crack nucleation in the adhesive wear of an elastic-plastic half-space, *Journal of the Mechanics and Physics of Solids* 145, article number 104100, 2020.

[22] A. Friedlander, J. M. Martínez, B. Molina, and M. Raydan, Gradient methods with retards and generalizations, *SIAM Journal on Numerical Analysis* 36, pp. 275-289, 1998.

[23] W. B. Gragg and G. W. Stewart, A stable variant of the secant method for solving nonlinear equations, *SIAM Journal on Numerical Analysis* 13, pp. 889–903, 1976.

[24] L. Grippo, F. Lampariello, and S. Lucidi, A nonmonotone line search technique for Newton's method, *SIAM Journal on Numerical Analysis* 23, pp. 707–716, 1986.

[25] R. Haelterman, J. Degroote, D. van Heule, and J. Vierendeels, The quasi-Newton least squares method: A new and fast secant method analyzed for linear systems, *SIAM Journal on Numerical Analysis* 47, pp. 2347–2368, 2009.

[26] N. Ho, S. D. Olson, and H. F. Walker, Accelerating the Uzawa algorithm, *SIAM Journal on Scientific Computing* 39, pp. 461–476, 2017.

[27] J. Jankowska, Theory of Multivariate Secant Methods, *SIAM Journal on Numerical Analysis* 16, pp. 547–562, 1979.

[28] C. T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*, Society for Industrial and Applied Mathematics, Philadelphia, 1995.

[29] W. La Cruz, A spectral algorithm for large-scale systems of nonlinear monotone equations, *Numerical Algorithms* 76, pp. 1109–1130, 2017.

[30] W. La Cruz, J. M. Martínez, and M. Raydan, Spectral residual method without gradient information for solving large-scale nonlinear systems of equations, *Mathematics of Computation* 75, pp. 1429–1448, 2006.

[31] W. La Cruz and M. Raydan, Nonmonotone spectral methods for large-scale nonlinear systems, *Optimization Methods and Software* 18, pp. 583–599, 2003.

[32] C. Le Bris, Computational chemistry from the perspective of numerical analysis, *Acta Numerica* 14, pp. 363–444, 2005.

[33] D.H. Li and M. Fukushima, A derivative-free line search and global convergence of Broyden-like method for nonlinear equations, *Optimization Methods and Software* 13, pp. 181–201, 2000.

[34] F. Lindner, M. Mehl, K. Scheufele, and B. Uekermann, A comparison of various quasi-Newton schemes for partitioned fluid-structure interaction, in *ECCOMAS Coupled Problems in Science and Engineering*, Venice, B. A. Schrefler, E. Oñate, and M. Papadrakakis, eds., Barcelona, 2015, DIMNE, pp. 477–488.

[35] J. M. Martínez, Three new algorithms based on the sequential secant method, *BIT* 19, pp. 236–243, 1979.

[36] E. Meli, B. Morini, M. Porcelli, and C. Sgattoni, Solving nonlinear systems of equations via spectral residual methods: stepsize selection and applications, arXiv:2005.05851v2, 2020.

[37] N. H. Miller and M. Osborne, Spatial differentiation and price discrimination in the cement industry: evidence from a structural model, *The RAND Journal of Economics* 45, pp. 221–247, 2014.

[38] L. N. H. G. Oliveira, Private communication, 2019.

[39] S. S. Oren, Self-scaling variable metric algorithms without line search for unconstrained minimization, *Mathematics of Computation* 27, 873–885, 1973.

[40] J. M. Ortega and W. C. Rheinboldt, *Iterative solution of nonlinear equations in several variables*, Academic Press, 1970.

[41] M. Pernice and H. F. Walker, NITSOL: a Newton iterative solver for nonlinear systems, *SIAM Journal on Scientific Computing* 19, 302–318, 1998.

[42] P. Pulay, Convergence acceleration of iterative sequences: the case of SCF iteration, *Chemical Physics Letters* 73, pp. 393–398, 1980.

[43] M. Raydan, On the Barzilai and Borwein choice of steplength for the gradient method, *IMA Journal of Numerical Analysis* 13, pp. 321–326, 1993.

[44] M. Raydan, The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem, *SIAM Journal on Optimization* 7, pp. 26–33, 1997.

[45] M. Rypdal and O. Løvsletten, Modeling electricity spot prices using mean-reverting multifractal processes, *Physica A: Statistical Mechanics and its Applications* 392, pp. 194–207, 2013.

[46] H. Schwetlick, Numerische Losung Nichtlinearer Gleichungen, Deutscher Verlag, 1978.

[47] H. F. Walker and P. Ni, Anderson acceleration for fixed-point iterations, *SIAM Journal on Numerical Analysis* 49, pp. 1715–1735, 2011.

[48] H. F. Walker, C. S. Woodward, and U. M. Yang, An accelerated fixed-point iteration for solution of variably saturated flow, Proceedings of the XVIII International Conference on Water Resources, CMWR 2010, J. Carrera, ed., CIMNE, Barcelona, 2010.

[49] P. Wolfe, The secant method for simultaneous nonlinear equations, *Communications of ACM* 2, pp. 12–13, 1959.