A new choice for the forcing term and a global convergent inexact Newton method

Márcia Aparecida Gomes-Ruggiero * Véra Lucia Rocha Lopes †
Julia Victoria Toledo-Benavides ‡

Abstract

Inexact Newton methods for solving $F(x) = 0$, $F : D \subset I\!R^n \to I\!R^n$ with $F \in C^1(D)$,where D is an open and convex set, find approximation to the step s_k of the Newton's systems $J(x_k)s = -F(x_k)$, instead of solving this system exactly as done by Newton's method. This means that s_k must satisfy a condition like $||F(x_k) + J(x_k)s_k|| \leq \eta_k||F(x_k)||$ for a forcing term $\eta_k \in [0, 1]$ ([7]). Many authors have presented possible choices for η_k (see [11]). In this work, a new choice for η_k is introduced, the new method obtained is globalized by the introduction of a robust backtracking strategy (see [2], [9]), and its convergence properties are proved. The numerical performance of the new method is presented by plotting the performance profile of the method as proposed in [10]. The results obtained show a competitive new inexact Newton method.

Key words: inexact Newton method, forcing term, global convergence, iterative linear system solver, GMRES.

1 Introduction

Consider the nonlinear system

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

where $F \in C^1(D, I\!R^n)$ and assume also that there is $x_* \in D$ such that $F(x_*) = 0$ with $J(x_*)$ nonsingular, where $J(y)$ is the Jacobian matrix of F at y. From now on, $\|\cdot\|$ is a norm in $I\!R^n$ and also its corresponding matrix norm in $I\!R^{n\times n}$. Let $V(x, r)$ denote an r-neighborhood of x and let, for $\gamma > 0$,

$$Lip_\gamma(D) = \{g \text{ such that } \|g(x) - g(y)\| \leq \gamma\|x - y\|, \ \forall \ x, y \in D\}.$$

*DMA-IMECC-UNICAMP, 13083-970 Campinas, SP, Brazil. This author was supported by FAPESP (Grant 2000-00375-4), PRONEX - Optimization 76.79.1008-00 and CNPq (Grant 140710/99-0). e-mail: marcia@ime.unicamp.br.

†DMA-IMECC-UNICAMP, 13083-970 Campinas, SP, Brazil. This author was supported by PRONEX - Optimization 76.79.1008-00 and FAPESP (Grants 2001-07987-8 and 2001-04597-4). e-mail: vlopes@ime.unicamp.br

‡DMA-IMECC-UNICAMP, 13083-970 Campinas, SP, Brazil. This author was supported by CNPq (Grant 300603/99-1). e-mail: julia@ime.unicamp.br.

We will assume further that J is $\lambda-$Lipschitzian at x_*, that is, there exists a constant $\lambda > 0$, such that

$$||J(x) - J(x_*)|| \leq \lambda \, ||x - x_*|| \tag{2}$$

for x sufficiently close to x_*. Let F have a solution in the open, convex set $D \subset I\!R^n$. The most popular method for solving problem (1) is Newton's method. The kth step of this method consists on: given x_k, find s_k, the exact solution of

$$J(x_k)s = -F(x_k), \tag{3}$$

where $J(x)$ denotes the Jacobian matrix of F at x. Then,

$$x_{k+1} = x_k + s_k. \tag{4}$$

It is well known that Newton's method has local quadratic convergence; however, it has some drawbacks: at each iteration the Jacobian matrix at x_k must be computed and the solution of the linear system (3) is required. Therefore, Newton's method may be computationally expensive. To get rid of these drawbacks, several modifications of Newton's method were proposed in the last 40 years, such as the quasi-Newton methods, the discrete Newton's method and the inexact Newton methods.

At each iteration, instead of solving (3), the quasi-Newton methods solve $B_k s = -F(x_k)$, where B_k is an approximation of $J(x_k)$ chosen in a specific way.

A very common way to choose B_k is to update these matrices imposing the secant equation

$$B_{k+1}s_k = y_k = F(x_{k+1}) - F(x_k). \tag{5}$$

With this approximation, there is no need to compute derivatives and, in some cases, the solution of the linear systems is simplified. However, there is a price to be paid: the convergence of quasi-Newton methods is at most superlinear. Some well-known methods of this class are Broyden's method [4], the Column-Updating method, [13], [14], [19] and the Inverse Column-Updating method [18], [20].

Another modification of Newton's method is given by the discrete Newton's method. In this case the system (3) is solved with the Jacobian matrix approximated by finite differences. This process requires n extra function evaluations by iteration, since now each column of B_k is obtained by

$$(B_k)_j \;=\; \frac{F(x_k + he_j) - F(x_k)}{h}\,, \; 1 \leq j \leq n,$$

where e_j is the jth vector of the canonical basis of $I\!R^n$. Choosing conveniently the size of h, and assuming some other properties on F, quadratic convergence can be obtained. In the case of large-scale sparse systems with a convenient sparsity pattern, the computation of B_k can be simplified by means of the following strategy, proposed by Curtis, Powell and Reid, [6]. The columns of B_k are separated in p groups and just one function evaluation is used to update a whole group. The condition imposed on the columns to belong to a group is that they have all their nonzero entries in different lines. This finite difference way of approximating $J(x_k)$ in

general means a significant decrease in the total number of function evaluations for large scale structured systems. See also [12] and [21].

The idea of inexact Newton methods is different from the ones previously cited. For solving $F(x) = 0$, $F : D \in I\!R^n \to I\!R^n$ with D an open, convex set and $F \in C^1(D)$ with an inexact Newton method, the step s_k in the kth iteration of the Newton's method (called an outer iteration) is found approximately, by applying an iterative linear system solver, whose iterations are called inner iterations. Solving approximately means that s_k must satisfy a condition like $||F(x_k)+J(x_k)s_k|| \leq \eta_k||F(x_k)||$ for a forcing term $\eta_k \in [0, 1)$ ([7]). As an iterative linear system solver, the most popular method used is the Generalized Minimum Residual (GMRES), [23].

Inexact Newton methods are the subject of this work. We make a brief description of the most used choices for the forcing term η_k, as were proposed by Eisenstat and Walker in [11]. A new choice is then introduced and its geometrical motivation is presented, as well as the algorithm incorporating a globalizing backtracking strategy. These subjects compose Section 2 as well as convergence results and their proofs. In Section 3 we present and analyze the numerical performance of this new inexact Newton method. This is done by plotting the performance profile [10] of the new method, in comparison with the ones proposed in [11]. Finally we make some comments and present some conclusions, in Section 4.

2 The New Method

The inexact Newton method proposed in this work introduces a new way of choosing the forcing term η_k initially motivated geometrically. In the algorithm a backtracking strategy is incorporated, increasing its robustness. These features of the method will be described and studied in the next subsections, after a brief review of existing inexact Newton methods.

2.1 About inexact Newton methods

Consider $F : D \subset I\!R^n \to I\!R^n$, D an open convex set in $I\!R^n$ and assume also that there is $x_* \in D$ such that $F(x_*) = 0$ with $J(x_*)$ nonsigular and λ-Lipschitzian at x_*. Dembo, Eisenstat and Steihaug [7] proposed an algorithm for an inexact Newton method for finding x_*, which can be formulated as:

Algorithm 1 (Inexact Newton Method)
`Step 1:` Choose $x_0 \in I\!R^n$, the initial approximation.
`Step 2:` For $k = 1$ until "convergence"
`step 2.1:` Choose $\eta_k \in [0, 1)$ and find s_k satisfying

$$||F(x_k) - J(x_k)s_k|| \leq \eta_k||F(x_k)||, \tag{6}$$

using an iterative solver for (3)
`Step 2.2:` $x_{k+1} = x_k + s_k$

Besides meaning a certain accuracy in solving the system (3), which motivates the denomination of *forcing term*, η_k also controls the number of inner iterations to be performed for each

outer iteration. Many times, values of η_k close to zero ($\eta_k = 0$ is Newton's method), imply a large number of iterations of the linear solver for each outer iteration, e. g., when x_k is far from the solution. So, a too small choice of η_k does not guarantee a decrease of $||F(x)||$. Eisenstat and Walker [11] call this fact an "oversolving" of the Newtonian system. Until now, the main purpose of introducing good choices of η_k is to avoid "oversolvings;" obviously, another purpose is the achievement of fast local convergence also. In [11] the authors introduce two choices for η_k which completely accomplish the desired objectives, and they prove convergence results for both choices.

Their first choice of η_k reflects the agreement between the function and its local linear model:

Choice 1: For $\alpha = (1 + \sqrt{5})/2$ and $\eta_0 \in [0, 1)$,

$$\eta_k = \frac{\|F(x_k) - F(x_{k-1}) - J(x_{k-1})s_{k-1}\|}{\|F(x_{k-1})\|}, \qquad k = 1, 2, 3, \cdots \tag{7}$$

or

$$\eta_k = \frac{| \, \|F(x_k)\| - \|F(x_{k-1}) + J(x_{k-1})s_{k-1}\| \, |}{\|F(x_{k-1})\|}, \qquad k = 1, 2, 3, \cdots \tag{8}$$

using as safeguard

$$\eta_k = \max\{\eta_k, \eta_{k-1}^{\alpha}\} \qquad \text{each time} \qquad \eta_{k-1}^{\alpha} > 0.1. \tag{9}$$

The formulae (7), (8) reflect in different ways the agreement between the function F and its linear model at the previous step. The choice of η_k in (8) is at least as small as the one in (7).

The second choice made by Eisenstat and Walker in [11] measures the decreasing factor in the value of $||F||$:

Choice 2: Given $\gamma \in [0, 1]$, $\alpha \in (1, 2]$ and $\eta_0 \in [0, 1)$, choose

$$\eta_k = \gamma(\frac{\|F(x_k)\|}{\|F(x_{k-1})\|})^{\alpha} \qquad k = 1, 2, ... \tag{10}$$

using the safeguard

$$\eta_k = \max\{\eta_k, \gamma\eta_{k-1}^{\alpha}\} \quad \text{each time} \quad \gamma\eta_{k-1}^{\alpha} > 0.1 \tag{11}$$

In both cases some other practical safeguards were also used for not allowing η_k to become too small too fast, which causes oversolving.

Under the standing assumptions on F made at the beginning of this subsection, and assuming also that $F(x_k) \neq 0$ for all k, the authors in [11] proved superlinear convergence results for the algorithm with Choice 1, and for the second Choice the convergence is proved to be of $q-$order α if $\gamma < 1$, and if $\gamma = 1$ the convergence is of $r-$order α and $q-$order p for every $p \in [1, \alpha)$. The inexact Newton method as presented in Algorithm 1 is a locally quadratically convergent method. In order to get a globally convergent algorithm a line search is usually introduced with which $x_{k+1} = x_k + \alpha_k s_k$, where $\alpha_k > 0$ must be such that a sufficient decrease in $||F(x)||$ is achieved.

2.2 The new choice for the forcing term η_k

As was already said, the main concern in the choice of the forcing term has been to avoid inner oversolvings. Nevertheless, our motivation to look for a new choice of η_k involved also the outer iterations; they are in general expensive because they need a new evaluation of the Jacobian matrix and also because for each outer iteration, the line search in the linear solver will require extra function evaluations, which also contribute for the computational price of the inexact Newton method. Therefore, we decided to look for a choice of η_k that could deal well with the inner oversolving and also with the *outer oversolving*. We call outer oversolving the need for many outer iterations due to a poor step generated by the linear solver. A new choice for η_k and the inclusion of a line search, strong enough to minimize both the inner and the outer iterations were our goal in this work. We then designed a choice for η_k taking into account the relation between the change in the value of $||F||$ during one outer iteration and the number of inner iterations performed to complete an outer iteration. We introduced in the algorithm the global line search used in [2] and [9], focusing on minimizing both the inner and the outer number of iterations.

2.3 The geometrical motivation

When trying to choose the next value for η, η_{k+1}, it seems to be important to consider the information: the variation in the norm of F, $\|F_{k+1}\| - \|F_k\|$ and the computational cost at the iteration k. Since each inner iteration involves the solution of a linear system and a line search procedure, we define the computational cost (price_k) as the number of iterations performed by the iterative linear solver (iterin_k) plus the number of function evaluations (feval_k), that is $\text{price}_k = \text{iterin}_k + \text{feval}_k$. Note that both ($\text{iterin}_k$) and ($\text{feval}_k$) are computed from the beginning of the process until the step k.

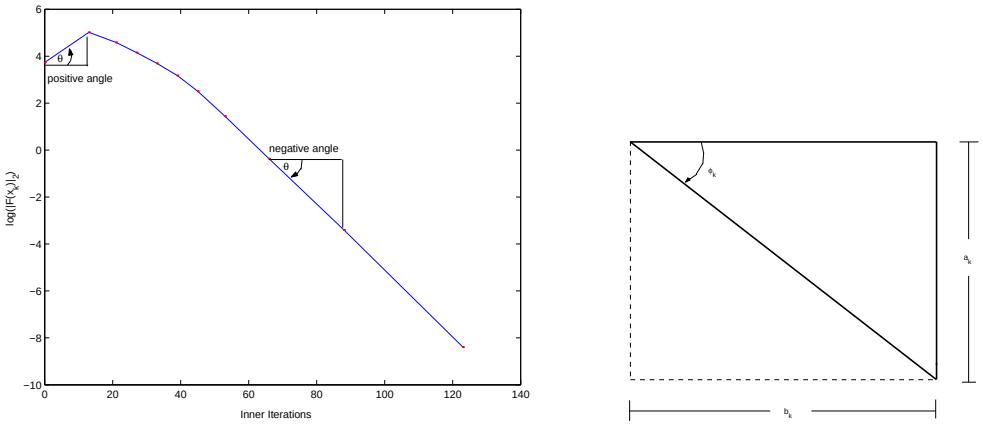

Figure 1: Geometrical motivation for choosing η_k and detail of the triangle

We used these measures at the ratio:

$$\frac{a_k}{\sqrt{a_k^2 + b_k^2}}, \tag{12}$$

where $a_k = (\log_{10}\|F_{k+1}\| - \log_{10}\|F_k\|)$ and $b_k = \log_{10}(\text{price}_k) - \log_{10}(\text{price}_{k-1})$, seen triangle ABC in Figure 1, right. This Figure also shows that the ratio (12) is the cosine of the angle θ_k, ($\theta_k \in (-\pi/2\,,\,\pi/2)$). It seems to be an important ingredient to be used when trying to choose η_k. If this ratio is small, then we get a significant decrease in $\|F\|$ for a comparatively small cost. In other words, it is advantageous to perform several inner iterations before going to the outer iteration. On the other hand, if the ratio is large, the inner iterations are not achieving much, so we should quickly update x_k at the outer iteration.

When $||F||$ increases from one iteration to the next (see Figure 1, left) , $\theta_k > 0$ and if $\|F\|$ decreases, $\theta_k < 0$. Notice also that θ_k becomes close to $-\pi/2$ when $||F||$ decreases sharply and that $\theta_k > 0$ or too close to 0 means oversolving.

Therefore, our strategy is to tie the choice of η_k to the variations of θ_k in the following way: the value of η_k is decreased when θ_k is close to $-\pi/2$; otherwise, its value is increased. Keeping also in mind the convergence of the algorithm, we introduce our choice:

$$\eta_k \;=\; [1/(k+1)]^{1.1}\cos^2(\theta_k)\frac{||F(x_k)||}{||F(x_{k-1})||} \tag{13}$$

This choice for η_k consists of three terms. The introduction of the first one, $[1/(k+1)]^{1.1}$ has the purpose of yielding a superlinear convergence rate to the algorithm. It is known (see [7]) that superlinear convergence rate is attained if $\eta_k \to 0$ when $k \to \infty$. Besides that, this factor constitutes also a weight for the term $\cos^2(\theta_k)$. Observe that, for the same angle θ, $[1/(k+1)]^{1.1}\cos^2(\theta)$ decreases as k increases, ensuring superlinear convergence. The second term, $\cos^2(\theta_k)$ has the geometrical motivation already described. We use the power 2 to skip the computation of a square root and also to accelerate the convergence of the process, since the function $\cos(x)$ decreases very slowly from 1 to 0, when $x \in [0\,,\,\pi/2]$. Finally, the factor $\|F(x_k)\|/\|F(x_{k-1})\|$ has the objective of considering the decreasing rate in $||F||$ from iteration $(k-1)$ to iteration k. Although this expression for η_k is similar to that of Choice 2 from Eisenstat and Walker ([11]), we observe that there exists a crucial difference between them: in our choice γ changes dinamically during the process, while in Choice 2 of [11] γ remains constant.

2.4 About the line search

The line search used, besides being a global strategy, is intended to avoid outer oversolvings. This is how it works: let $\sigma \in (0,1)$, $\varrho_{min} < \varrho_{max} < 1$ and $\{\mu_k\}$ $(k = 0, 1, 2, ...)$ be a sequence of positive numbers, such that

$$\Sigma_{k=0}^{\infty}\mu_k = \mu < \infty, \tag{14}$$

and let $x_0 \in I\!R^n$ be a initial guess for the solution of $F(x) = 0$.

Given $x_k \in I\!R^n$, the kth approximation to the solution, do

Algorithm 2 (Line Search [2], [9])
`Step 1:` Compute the search direction s_k;
`Step 2:` $\xi = 1$;
`Step 3:`
`step 3.1:` While

$$\|F(x_k + \xi s_k)\| > [1 - \xi\sigma]\|F(x_k)\| + \mu_k, \tag{15}$$

step 3.2: take $\xi_{new} \in [\varrho_{min}\xi,\ \varrho_{max}\xi]$,
step 3.3: $\xi = \xi_{new}$;
Step 4:
step 4.1: $\xi_k = \xi$;
step 4.2: $x_{k+1} = x_k + \xi_k s_k$;

This nonmonotone strategy is similar to the one introduced by Li and Fukushima in [17], but less prone to scaling problems than that.

Observe that this iteration is well defined and that s_k is allowed to be equal to zero. As will be stressed in the section of numerical experiments, an adequate line search improves the numerical performance of the algorithms.

2.5 The Algorithm

Now we are ready to introduce the new algorithm; its numerical performance is presented in the section Numerical Experiments. The following assumptions are needed to define the algorithm and also in the proofs of the convergence results in the next section. Assume that $F : I\!R^n \to I\!R^n$ is continuously differentiable in $I\!R^n$. Assume also that $\sigma \in (0\,,\,1)$, $0 < \rho_{min} < \rho_{max} < 1$ and that $\{\mu_k\}$ is a sequence such that $\mu_k > 0$ for all $k = 0, 1, 2, \ldots$ and $\sum_{k=0}^{\infty} \mu_k = \mu < \infty$.

Let $x_0 \in I\!R^n$ be an arbitrary initial point, set $k = 0$. Given $x_k \in I\!R^n$, the kth iterate of the algorithm and the precision $\varepsilon > 0$, the steps for obtaining the new iterate x_{k+1} are the following:

Algorithm 3. (Inexact Newton method):
While $\|F(x_k)\|_2 > \varepsilon$,
Step 1: Choose η_k;
Step 2: Find s_k such that

$$\|F(x_k) + J(x_k)s_k\|_2 \leq \eta_k \|F(x_k)\|_2;$$

Step 3: $x_{aux} = x_k + s_k$ and then compute $F(x_{aux})$. Set $\xi = 1$.
Step 4: While

$$\|F(x_{aux})\|_2 > [1 - \xi\sigma]\|F(x_k)\|_2 + \mu_k, \tag{16}$$

step 4.1: compute $\xi_{new} \in [\varrho_{min}\xi, \varrho_{max}\xi]$, and
step 4.2: $\xi_k = \xi_{new}$;
step 4.3: $x_{aux} = x_k + \xi_k s_k$
Step 5: $x_{k+1} = x_{aux}$; $k = k + 1$;

2.6 Convergence

In this subsection we state and prove convergence results for Algorithm 3. We will not present the proof for Lemma 2.6.1, because it is basically the same as that of Lemma 1 in [2].

Lemma 2.6.1. *Let x_k be a sequence generated by Algorithm 3. If, for some sequence of indices $K_0 \subset \{0, 1, 2, \ldots\}$, $\lim_{k \in K_0} F(x_k) = 0$, then*

$$\lim_{k \to \infty} F(x_k) = 0.$$

In particular, if x_* is a limit point of x_k such that $F(x_*) = 0$, then every limit point of the sequence x_k is a solution of (1).

Proof. (See [2])

The proof of the next Lemma is identical to the one of Lemma 2 in [2]; so, we will only state it, too.

Lemma 2.6.2. *Let x_k be a sequence generated by Algorithm 3 and assume that all the limit points of the sequence x_k are solutions of (1). Assume also that x_* is a limit point of x_k such that $J(x_*)$ is nonsingular and*

$$\lim_{k\to\infty} \|x_{k+1} - x_k\| = 0.$$

Then, the whole sequence converges to x_*.

Proof. (See [2])

We finish this section with the convergence Theorems.

Theorem 2.6.1. *Assume that the sequence $\{x_k\}$ is generated by Algorithm 3 and that there exists $M > 0$ such that, for an infinite sequence of indices $K_1 \subset \{0, 1, 2, \ldots\}$,*

$$\|J(x_k)s_k + F(x_k)\|_2 \leq \eta_k \|F(x_k)\|_2 \tag{17}$$

and $\|s_k\|_2 < M$. Then any limit point of the subsequence $\{x_k\}_{k\in K_1}$ is a solution of the system 1. Moreover, if a limit point of $\{x_k\}$ exists, then $F(x_k) \to 0$ and every limit point of $\{x_k\}$ is a solution to 1.

Proof.

Let $K_2 \subset K_1$ be a sequence of indices such that $lim_{k\in K_2} = x_*$. The proof will be done, considering two cases. Firstly, let us consider that $\{\xi_k\}_{k\in K_2}$ does not tend to 0. In this case, there will exist a sequence K_3 of indices, $K_3 \subset K_2$ and $\bar{\xi} > 0$ such that $\xi_k \geq \bar{\xi} > 0$, $\forall\, k \in K_3$.

So, by (16),

$$\|F(x_{k+1})\| \leq \|F(x_k)\| + \bar{\xi}\sigma\|F(x_k)\| + \mu_k, \;\; \forall k \in K_3.$$

But for all k, including $k \notin K_3$,

$$\|F(x_{k+1})\| \leq \|F(x_k)\| + \mu_k.$$

Then, adding all these inequalities, we have:

$$\sigma\bar{\xi} \sum_{k\in K_3} \|F(x_k)\| \leq \|F(x_0)\| + \sum_{k=0}^{\infty} \mu_k = \|F(x_0)\| + \mu.$$

Therefore, $\lim_{k\in K_3} = 0$ and then, $F(x_*) = 0$.

Let us consider now the second case, in which we assume that $\lim_{k\in K_2}\xi_k = 0$. Taking into account the way in which ξ_{new} is chosen, for $k \in K_2$, k large enough, there exists $\xi_k^{'} > \xi$, $\xi_k^{'} \in [\xi_k/\rho_{max}\,,\, \xi_k/\rho_{min}]$ such that $\lim_{k\in K_2}\xi_k^{'} = 0$ and

$$\|F(x_k + \xi_k^{'}s_k)\| > \|F(x_k)\| - \xi_k^{'}\sigma\|F(x_k\| + \mu_k.$$

Then,

$$\|F(x_k + \xi_k^{'}s_k\| > (1 - \xi_k^{'}\sigma)\|F(x_k)\|.$$

So, $\|F(x_k + \xi_k^{'})s_k\| - \|F(x_k) + J(x_k)\xi_k^{'}s_k\| + \|F(x_k) + J(x_k)\xi_k^{'}s_k\| > (1 - \xi_k^{'}\sigma)\|F(x_k)\|$.

Thus,

$$\|F(x_k + \xi_k^{'}s_k) - F(x_k) - J(x_k)\xi_k^{'}s_k\| + \|\xi_k^{'}[F(x_k) + J(x_k)s_k]\| - \xi_k^{'}\|F(x_k\| > (1 - \sigma_k s_k)\|F(x_k)\|.$$

Now, by (17),
$\|F(x_k + \xi_k^{'}s_k) - F(x_k) - J(x_k)\xi_k^{'}s_k\| + \xi_k^{'}\|F(x_k)\| + (1 - \xi_k^{'})\|F(x_k\| > (1 - \sigma_k s_k)\|F(x_k)\|$.

After some algebraic manipulation, we have

$$\xi_k^{'}\|F(x_k\|(1 - \sigma) < \|F(x_k + \xi_k^{'}s_k) - F(x_k) - J(x_k)\xi_k^{'}s_k)\|.$$

Then,

$$\|F(x_k\| < \frac{\|F(x_k + \xi_k^{'}s_k) - F(x_k) - J(x_k)\xi_k^{'}s_k)\|}{\xi_k^{'}}. \tag{18}$$

Now, $\|s_k\|$ is bounded, $\{\xi_k^{'}\}$ tends to 0 and $F^{'}$ is continuous. These things together imply that the right-hand side of (18) tends to 0 when $k \in K_2$. Therefore, $lim_{k\in K_2}\|F(x_k\| = 0$ and then $F(x_*) = 0$. The rest of the proof follows from Lemma 2.6.1. ∎

Theorem 2.6.2. *Let all the assumptions of Theorem 2.6.1 hold. If (17) is valid for k large enough, with* $\lim_{k\to\infty}\eta_k = 0$*, then the convergence of* x_k *to* x_* *is superlinear.*

Proof.

By Theorem 2.6.1, $\lim_{k\to\infty}\{x_k\} = x_*$. Since $\|s_k\| \leq M$ and by the uniform continuity of $J(x)$, we have that

$$\|F(x_k + s_k) - [F(x_k) + J(x_k)s_k]\| \leq o(\|s_k\|),\ \forall\, k = 0, 1, 2, \ldots.$$

So,

$$\|F(x_k + s_k)\| - \|F(x_k) + J(x_k)s_k\| \leq o(\|s_k\|),$$

that is,

$$\|F(x_k + s_k)\| \leq \|F(x_k) + J(x_k)s_k\| + o(\|s_k\|) \leq \eta_k\|F(x_k\| + o(\|F(x_k)\|.$$

Since $\|F(x_k\|$ tends to 0, $\dfrac{\|F(x_k + s_k)\|}{\|F(x_k)\|} \leq \eta_k$ also tends to 0.

Thus, for k large enough, $\|F(x_k + s_k)\| \leq (1 - \sigma)\|F(x_k)\|$, which means that

$$\|F(x_{aux})\| \leq (1 - \xi\sigma)\|F(x_k)\| + \mu_k$$

is true for $\xi = 1$. Then for k large enough, $x_{k+1} = x_k + s_k$ and so,

$$\lim_{k\to\infty} \frac{\|F(x_{k+1})\|}{\|F(x_k\|} = 0. \tag{19}$$

Since $J(x_*)$ is nonsingular, there exist $c > 0$ and $C > 0$ such that

$$c\|x - x_*\| \leq \|F(x_k\| \leq C\|x - x_*\|$$

for all x close enough to x_*. Then, by (19),

$$\lim_{k\to\infty} \frac{\|x_{k+1} - x_*\|}{x_k - x_*\|} = 0.$$

∎

3 Numerical Experiments

3.1 Introduction

In order to test the new algorithm proposed in this work we implemented it and the inexact Newton algorithms with η_k constant ($\eta_k = 0.01$) and with the two choices proposed by Eisenstat and Walker in [11]; the same global line search strategy described previously was used in all the tests. We used as test problems three kinds of two-dimensional boundary-value problems, whose general formulation is:
Find $u : \Omega = [0, 1] \times [0, 1] \to I\!R$ such that, for $\lambda \in I\!R$,

$$-\Delta u + h(\lambda\,,\, u) \;=\; f(x, y), \;\; \text{in } \Omega \tag{20}$$

$$u(x, y) \;=\; 0 \;\; \text{on } \partial\Omega. \tag{21}$$

The real valued function $h(\lambda, u)$, the values of the parameter λ and the two-variables function f define different instances of this problem.

We used a grid with 63 interior points in each axis. The unknowns of the discretized system are the values of u at these grid points. All the derivatives were approximated using central differences. Replacing in (20) the function and the derivatives by their approximations, and using the boundary conditions, we obtain a nonlinear system of equations like (20-21), with dimension 3969 (the total number of grid points).

In sections 3.3 to 3.5, we define the operators and make a comparative analysis, using the performance profiles of the inexact Newton methods described in Section 2, applied to different instances of the system in (20) and (21). In all the cases, the boundary condition is $u = 0$ and Δ is the Laplacian operator.

3.2 Implementation Features

In this section we give more details about the implementation of the algorithms. All the tests were performed in an Pentium III - 1.0GHz computer, using the software MatLab 6.1. In what follows, we give details of the implementation of some parts of the method.

- Line search procedure
  If the vector $x_k + \xi_k s_k$ does not give an acceptable decrease in the value of the function, in the sense of (15), then we compute the new step size as $\xi_{new} = 0.5\xi_k$. For the parameter σ used in the criterion (15), we took $\sigma = 10^{-4}$.

- The sequence μ_k
  We define:
  $$ftip(0) = \|F(x_0)\|,$$
  $ftip(k) = \min\{\|F(x_k)\|, ftip(k-1)\}$, if k is a multiple of 3 and
  $ftip(k) = ftip(k-1)$, otherwise.
  Then, we set:
  $$\mu_k = \frac{ftip}{(k+1)^{1.1}}.$$

- The initial value and safeguards for η.
  For all the choices for η_k we set the initial value: $\eta_0 = 0.1$. For the choices 1 and 2 of [11] and for the new choice, $\eta_k = \min\{\eta_k, 0.1\}$ if $k \leq 3$, and $\eta_k = \min\{\eta_k, 0.01\}$ if $k > 3$. At the final iterations we have adopted the safeguard introduced in [22] which can be described as: since the linear model is $F(x) \sim F(x_k) + J(x_k)s$, at the final iterations, we can have: $||F(x_{k+1})|| \sim ||F(x_k) + J(x_k)s_k|| \leq \eta_k||F(x_k)||$. In this case it is important to set η_k such that $\eta_k||F(x_k)|| \sim \varepsilon$ where ε is the precision required for the nonlinear system. A safeguard which represents these ideas is: if $\eta_k \leq 2\varepsilon$ then we set $\eta_k = 0.8\varepsilon||F(x_k)||$.

- Stopping criterion
  The process is finished successfully if $\|F(x_k)\| \leq 10^{-6}$ and $k < 100$.

3.3 Performance profile

Introduced by Dolan and Moré, ([10]) in 2002, the "performance profile" is a powerful tool to compare the performance of n_s solvers of a set S when applied to solve n_p problems of a set P, using some measure such as the number of function evaluations or the computing time, for instance. It was applied here to compare the inexact Newton method with the new choice that we introduced, with the three other choices: the choice with η_k constant ($\eta_k = 0.01$) and choices 1 and 2, proposed in [11], which were described in subsection 2.1. We used the performance profile to compare all the problems tested with four options of measure: the number of inner iterations, the number of outer iterations, the number of function evaluations and also the elapsed `cpu` time.

Let $m_{s,p}$ denote the performance measurement , such as cpu time required to solve problem p by solver s. For each problem p and solver s the *performance ratio* $r_{s,p}$ is computed as

$$r_{s,p} = \frac{m_{s,p}}{\min\{m_{s,p}\ \forall s \in S\}}$$

if the problem p is solved by solver s; otherwise,

$$r_{s,p} = r_M,$$

where r_M is a large enough fixed parameter.

Then, for each $s \in S$, the cumulative distribution function $\rho_s : I\!R \to [0,1]$, for performance ratio $r_{s,t}$, is built:

$$\rho_s(t) = \frac{1}{n_p}\text{size}\{p \in P \ \mid \ r_{s,p} \leq t\}.$$

This function represents the performance of the solver s, it is nondecreasing and piecewise constant. At the analysis of solver s, the points $\rho_s(1)$ and $\overline{t}$, such that, $\rho_s(\overline{t}) = 1$, give us important information, Let $\overline{s}$ be the solver s which gives the maximum value for the function $\rho_s(1)$. This solver can solve the maximum number of problems using the minimum number of the measure m. The efficiency of the solver s in terms of the number of problems that can be solved is evaluated by the minimum value of t, denoted by $\overline{t_s}$, such that $\rho_s(\overline{t}) = 1$, if there exists such value for $t < r_M$. Therefore, the best solver in terms of efficiency will be the solver $\hat{s}$ for which $\overline{t_{\hat{s}}} = \min\{\overline{t}_s, \forall s \in S\}$.

3.4 Bratu problem

For the Bratu problem, $h(\lambda, u) = -\lambda e^u$ so our boundary-value problem is:

$$-\Delta u - \lambda e^u = f(x,y) \ \text{ in } \Omega, \quad u(x,y) = 0 \ \text{ on } \partial\Omega. \tag{22}$$

If $f(x,y) \equiv 0$ we have the homogeneous problem. In this case, there is a critical value λ_{crit} for which the problem has two solutions if $\lambda \in (0, \lambda_{crit})$ and no solution if $\lambda > \lambda_{crit}$, $\lambda_{crit} \approx 6.81$, [1]. Based on the formulation of Bratu problem we generated a set of problems by computing the function $f(x,y)$ in such a way that the exact solution is $u_*(x,y) = 10xy(1-x)(1-y)e^{x^{4.5}}$ (see [16],[9]). In this set of problems, positive values of λ result in a nonlinear system that is hard to solve [15].

In Figure 2, we show the results obtained when the four algorithms were applied for this problem with the following values for λ: -1000, -500, -250, -100, -50, -10, 1, 3, 5, 7 and 10. The initial approximations were: $x_0 = (0,\ 0, \ldots, 0)$ and a vector whose components were randomly generated in the interval: $[-5,\ 5]$, for a total of $n_p = 22$ problems. In Figure 2 the continuous line represents the performance of the new choice for η_k and this choice is indicated by `NC` in the legends. The choices 1 and 2 of [11] are indicated by `EW1` and `EW2` in the legends and represented at the figures by lines (`---`) and (`.-.-.`), respectively. The constant choice ($\eta_k = 0.01$) is represented by `C = 0.01` in the legends and by a line (...) at the figures. The measures used for the performance profile analysis are denoted by: `minner` for the number of inner iteration; `mouter` for the number of external iterations; `mevalf` for the number of function evaluations and `mcpu` for the `cpu` time.

Let us analyze now the important values: $\rho_s(1)$ and $\overline{t}$, such that, $\rho_s(\overline{t}) = 1$ from Figure 2. The new choice solved approximately 20% of the problems with the minimum value when

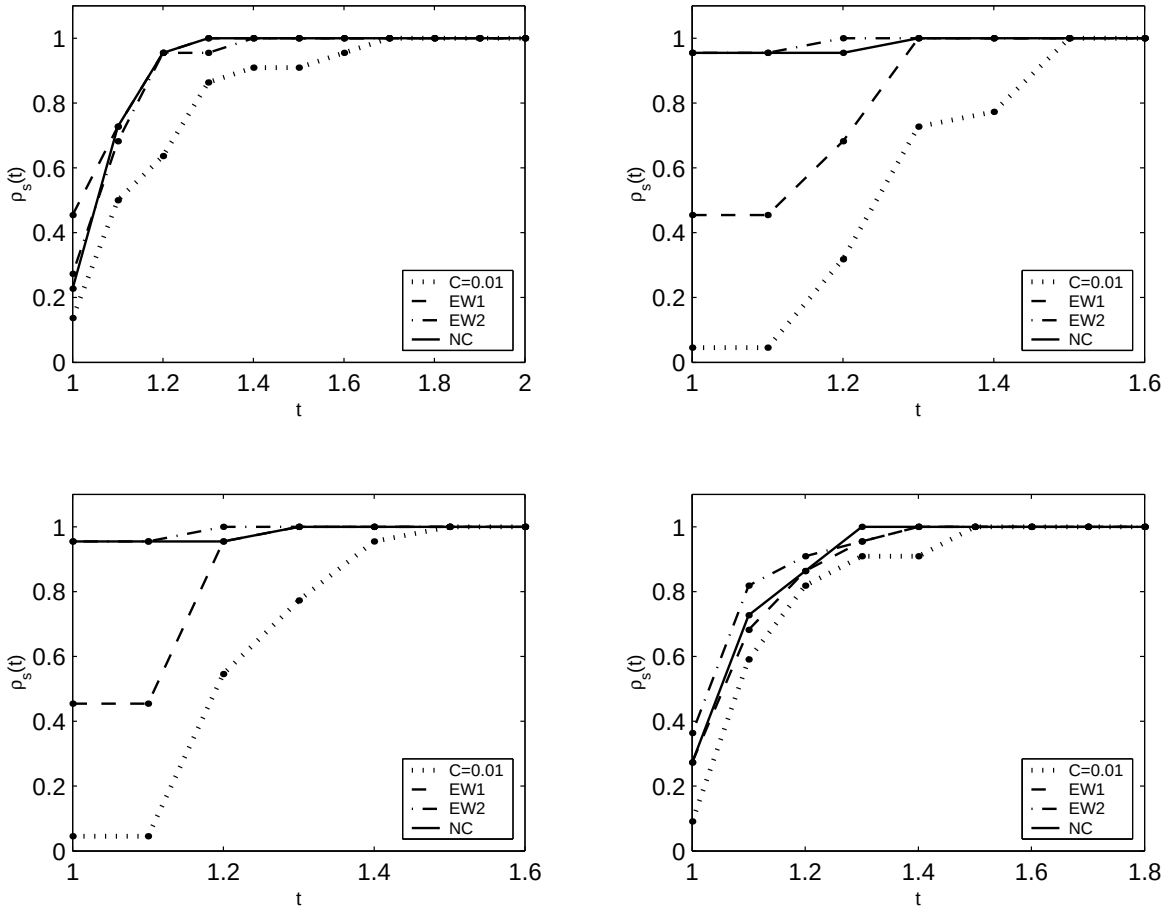

Figure 2: Performance profile using the inner and outer iterations, number of function evaluations and `cpu` time as measures, in clockwise order.

the number of inner iterations and the `cpu` time where used as measure, and it solved almost the total set of problems with the minimum number of outer iterations and number of function evaluations. For these three measures, the new choice achieved $\rho_s(\overline{t}) = 1$ when $\overline{t} = 1.3$. Also, the new choice is the best method in terms of the efficiency when the measures are the number of inner iterations and the `cpu` time, because it is the choice that achieves the value 1 for $\rho_s(t)$ with the minimum value for t. Even though using the other two measures (the number of outer iterations and the number of function evaluations), the new choice was not the best, it is still competitive. The choice with constant value for η_k had a very poor performance for this set of problems. The choice `EW2` had a similar overall performance as that of the choice `NC`. However, in the examples that follow, we will see that `NC` may outperform `EW2`. The choice `EW1` is also similar to `NC` in what concerns to the measures inner iterations and `cpu` time.

3.5 Convection-Diffusion problem

The Convection-Diffusion problems have the formulation:

$$-\Delta u + \lambda u(u_x + u_y) = f(x,y), \quad \text{in } \partial\Omega, \quad u(x,y) \;=\; 0 \text{ on } \partial\Omega, \; \lambda \in I\!R. \tag{23}$$

As in 3.3, $f(x,y)$ was computed in such a way that $u_*(x,y) = 10xy(1-x)(1-y)e^{x^{4.5}}$ is the solution.

In Figure 3 we show the results obtained when the four algorithms were applied for this problem with the following values for λ: 5, 10, 25, 50, 75, 100, 110, 125 and 150. In all the cases the initial approximation was $x_0 = (0, 0, \ldots, 0)$. In this figure we adopted the same symbols used for Bratu problem. We observed that the new choice was the first one to achieve the value 1 for the function $\rho_s(t)$ for all measures. This value was achieved at $\overline{t} = 1.1$ when the measures were outer iterations and number of function evaluations, $\overline{t} = 1.2$ when the measure was inner iterations and $\overline{t} = 1.3$ when the measure was `cpu` time. These results indicate, again, the robustness of the proposed choice for η_k. For all the measures, the new choice solved the maximum number of problems with the minimum value of the respectively measure. For inner iterations as measure, `NC` and `EW1` solved approximately 40% of the problems using the minimum value. However, `EW1` had a poor performance in terms of efficiency because it solved all the problems only for $\overline{t} = 1.9$! With the measures outer iterations and number of function evaluations, `NC` had a superior performance because it solved approximately 80% of the problems with the minimum value and achieved the value $\rho_s(t) = 1$ for $\overline{t} = 1.1$. `EW1` and `EW2` had similar performance and `C` had the worse performance: it was not the best in terms of the value of $\rho_s(1)$ for any measure and achieved the value $\rho_s(t) = 1$ only for high values for $\overline{t}$.

3.6 Another nonlinear problem

In this case, we solved the problem (20) proposed by Briggs, Henson and McCormick [3]:

$$-\Delta u + \lambda u e^u = f(x,y), \quad \text{in } \Omega, \quad u(x,y) = 0 \;\text{ on } \partial\Omega, \text{ where} \tag{24}$$

$$f(x,y) = ((9\pi^2 + \gamma e^{(x^2-x^3)sin(3\pi y)})(x^2 - x^3) + 6x - 2)sin(3\pi y)$$

In Figure 4, we show the results obtained when the four algorithms were applied for this problem with the following values for λ: 10, 100 and 1000. The initial approximations were vectors with these constant components: -2, -1, 0, 1, 2 and 10, and a vector whose components were randomly generated in the interval $[-2\ 2]$. Again, for all the measures, the choice `NC` solved the maximum number of problems with the minimum value of the measure, which indicates the efficiency of our strategy. For this set of problems, the choices `EW1`, `EW2` and `NC` had a similar robustness, that is, all of them solved all the problems with a value of $\overline{t} \in [1.2; 1.3]$. Also, the constant choice, `C`, had a poor performance: this choice was the last one to achieve the value 1 for the function $\rho_s(t)$ for all the measures, and this choice solved a low number of problems with the minimum number (only almost 5% of the problems) when the measures were the number of outer iterations and number of function evaluations.

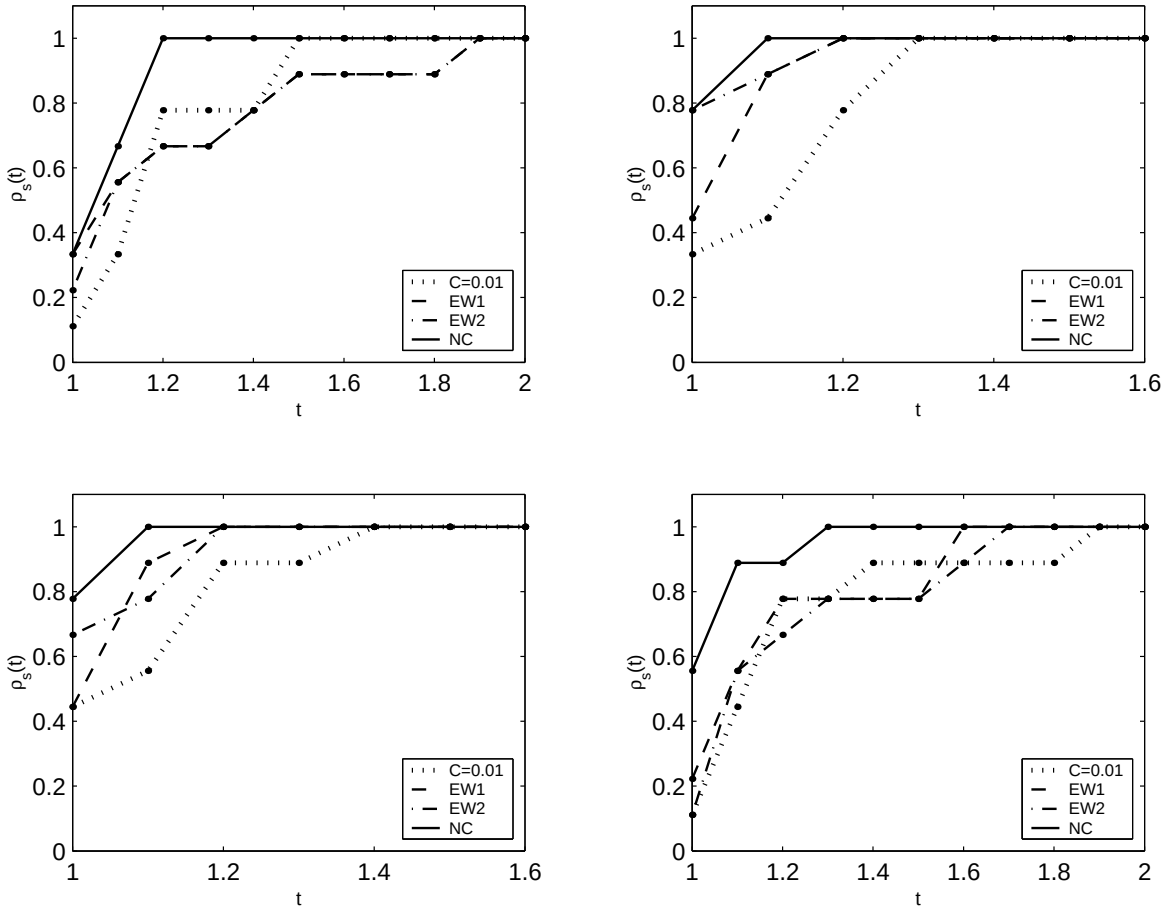

Figure 3: Performance profile using the inner and outer iterations, the number of function evaluations and `cpu` time as measures, in clockwise order.

4 Conclusions

In this work we proposed a new Inexact–Newton method with a different choice for the forcing term η_k and with a robust line search strategy, which resulted in an algorithm with a global convergence result. We think that the robustness of our algorithm is due to both strategies: the new choice of the forcing term and the inclusion of a good backtracking strategy. However, the new choice for η_k is what had the decisive influence in the performance, because, for comparing the different choices, the same backtracking strategy was introduced in all of them. The numerical experiments showed that this new algorithm is competitive in terms of number of inner and outer iterations performed, which allow us to conclude that our objectives were obtained: to built an inexact Newton algorithm avoiding the high number of inner iterations at linear system solver without performing a very large number of outer iterations. Besides, it is important to observe that these objectives were achieved without an increase in the number of function evaluations.

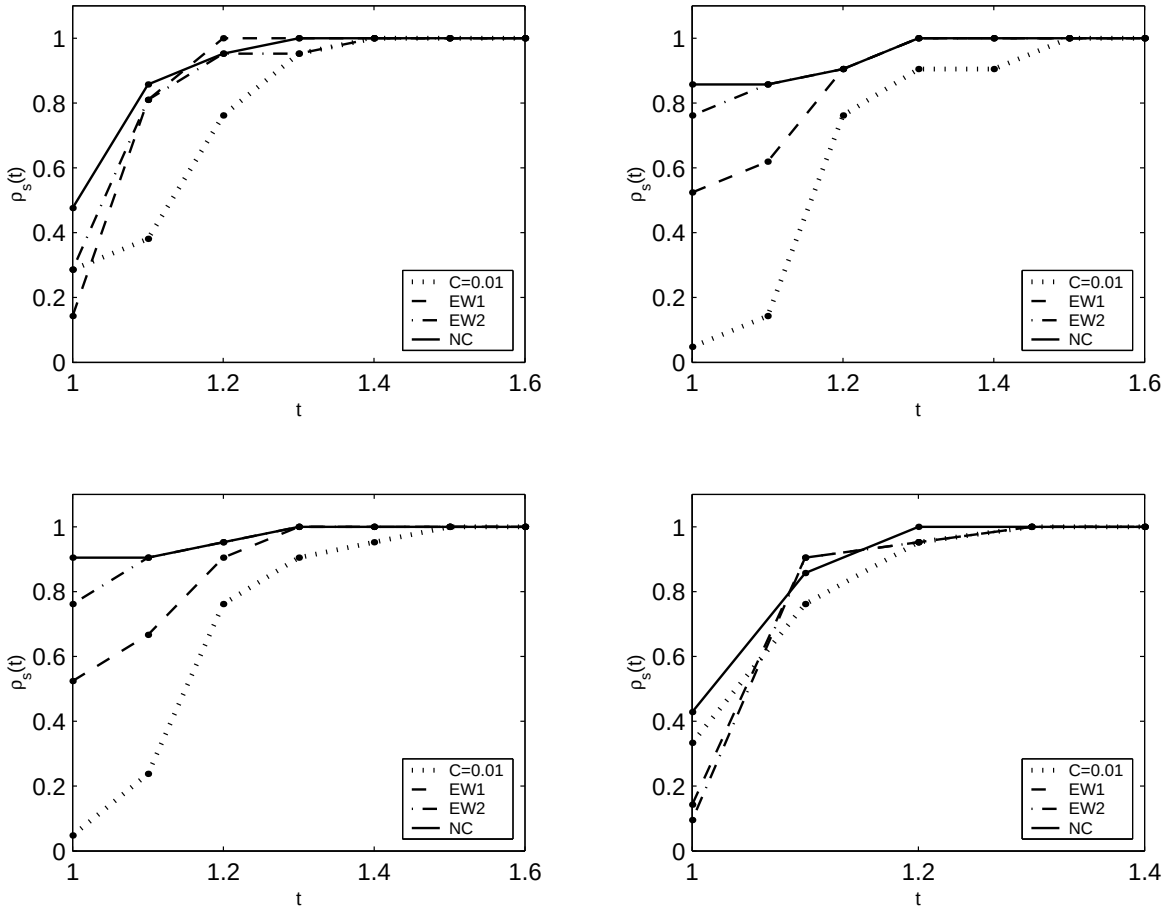

Figure 4: Performance profile using the inner and outer iterations, number of function evaluations and `cpu` time as measures, in clockwise order.

The constant choice has the only advantage of being easy to implement and `EW1` and `EW2` had a very similar and efficient performance. However, `NC` seems to be superior to the other choices because it is faster in terms of number of inner and outer iterations and requires fewer function evaluations, and it can solve the whole set of problems with the minimum value of t for almost all the measures.

References

[1] Averick, B. M., Carter, R. G., Moré, J. J. and Xue, G.-L., *The Minpack-2 test problem collection* Preprint MCS-P153-0692, Mathematics and Computer Science Division, Argonne National Laboratory, 1992.

[2] Birgin, E. G., Krejić, N. and Martínez, J. M., *Globally convergent inexact quasi-Newton methods for solving nonlinear systems*, Numerical Algorithms, Vol. 32, pp. 249–260, 2003.

[3] Briggs, W. L., Henson, V. E. and McCormick, S. F., A multigrid tutorial, 2nd. edition, SIAM, 2000.

[4] Broyden, C. G. , *A class of methods for solving sparse nonlinear systems*, Math. Comput., Vol. 25, pp. 285–294, 1965.

[5] Broyden, C. G., Dennis Jr., J. E. and Moré, J. J., *On the local and superlinear convergence of quasi-Newton methods*. J. Inst. Math. Appl. Vol. 12, pp. 223-245, 1973.

[6] Curtis, A., Powell, M. J. D. and Reid, J., *On the estimation of sparse Jacobian matrices*. J. Inst. Math Appl., Vol. 13, pp. 117-119, 1974.

[7] Dembo, R. S., Eisenstat, S. C. and Steihaug, T., *Inexact Newton methods*. SIAM J. Numer. Anal., Vol. 19, No. 2, pp. 401-408, 1982.

[8] Dennis, Jr., J. E. and Schnabel, R. B., Numerical Methods for Unconstrained Optimization and Nonlinear Equations, SIAM Classics in Applied Mathematics, 1996.

[9] Diniz-Ehrhardt, M. A., Gomes-Ruggiero, M. A., Lopes, V. L. R. and Martínez, J. M., *Discrete Newton's method with local variations for solving large-scale nonlinear systems*, to appear in Optimization.

[10] Dolan, E. D. and Moré, J. J., *Benchmarking optimization software with performance profiles*. Math. Program. Ser. A 91, pp. 291-213, 2002.

[11] Eisenstat, S. C. and Walker, H. F., *Choosing the forcing terms in inexact Newton method*. SIAM J. Sci. Comput., Vol. 17, No. 1, pp. 16-32, 1996.

[12] Goldfarb, D. and Toint, Ph. L., *Optimal estimation Jacobian and Hessian matrices that arise in finite difference calculations*. Mathematics of Computation, Vol. 43, No. 167, pp. 69-88, 1984.

[13] Gomes–Ruggiero, M. A. and Martínez, J. M., *The Column-Updating Method for Solving Nonlinear Equations in Hilbert Space*. M^2AN Mathematical Modeling and Numerical Analysis, Vol. 26, No. 2, pp. 309–330, 1992.

[14] Gomes–Ruggiero, M. A., Martínez, J. M. and Moretti, A. C., *Comparing Algorithms for Solving Sparse Nonlinear Systems of Equations*. SIAM Journal Scientific and Statistical Computing, Vol. 13, No.2, pp. 459-483, 1992.

[15] Gomes–Ruggiero, M. A., Kozakevich, D. N. and Martínez, *A Numerical Study on Large–Scale Nonlinear Solver*. Computers and Mathematics with Applications, Vol. 32, No. 3, pp. 1–13, 1996.

[16] Kelley, C. T.,Iterative methods for linear and nonlinear equations. SIAM, 1995.

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

[18] Lopes, V. L. R. and Martínez, J. M., *Convergence properties of the inverse column-updating method*. Optimization Methods and Software Vol. 6, pp. 127-144, 1995.

[19] Martínez, J. M., *A quasi-Newton method with modification of one column per iteration*. Computing, Vol. 33, pp. 353-362, 1984.

[20] Martínez, J. M. and Zambaldi, M. C., *An inverse column-updating method for solving large-scale nonlinear systems of equations*. Optimization Methods and Software Vol. 1, pp. 129-140, 1992.

[21] Newsam, G. N. and Ramsdell, J. D., *Estimation on sparse Jacobian matrices*. SIAM J. Algebraic Discrete Methods, Vol. 4, pp. 404-418, 1983.

[22] Pernice, M., and Walker, H., *NITSOL: a Newton iterative solver for nonlinear systems*. SIAM J. Sci. Comput., Vol. 19, No. 1, pp. 302-318, 1998.

[23] Saad, Y. and Schultz, M. H., *GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems*. SIAM J. Sci. Stat. Comput., Vol. 7, No. 3, 1986.