Discrete Newton's method with local variations for solving large-scale nonlinear systems *

Maria A. Diniz-Ehrhardt † Márcia A. Gomes-Ruggiero ‡
Vera L. Rocha Lopes § José Mario Martínez ¶

Abstract

A globally convergent discrete Newton method is proposed for solving large-scale nonlinear systems of equations. Advantage is taken from discretization steps so that the residual norm can be reduced while the Jacobian is approximated, besides the reduction at Newtonian iterations. The Curtis-Powell-Reid (CPR) scheme for discretization is used for dealing with sparse Jacobians. Global convergence is proved and numerical experiments are presented.

Key words: Nonlinear systems, discrete Newton's method, local variations method.

1 Introduction

The problem of solving nonlinear systems of equations

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

where

$$F : I\!R^n \to I\!R^n, \quad F \in C^1(I\!R^n),$$

$F = (f_1, \ldots, f_n)$, appears frequently in applications to Physics, Chemistry and Engineering [6, 8, 13].

In this paper we introduce a new method to solve problem (1) which is based on two ideas: the evaluation of the discrete approximation of large scale sparse Jacobian matrices by groups, with just one function evaluation per group [3, 7], and the local variations

*All the authors are from: DMA-IMECC-UNICAMP, 13083-970 Campinas SP, Brazil. They were supported by PRONEX-Optimization 76.79.1008-00, FAPESP (Grant 2001-04597-4) and CNPq.

†e-mail: cheti@ime.unicamp.br

‡e-mail: marcia@ime.unicamp.br

§Also supported by FAPESP (Grant 2001-07987-8); e-mail: vlopes@ime.unicamp.br

¶e-mail: martinez@ime.unicamp.br

method [1, 14, 15]. What characterizes our algorithm is the way in which we combine these ideas. Roughly speaking, an iteration of the new algorithm starts from a current "base point" and this point can be changed according to the local variations concepts. The groups used in the evaluation of the discrete Jacobian matrices are CPR–valid groups in the sense of [7]. Global convergence is obtained using classical backtracking with a tolerant strategy [10], performed on the Newtonian direction.

We present a model algorithm, a particular case of which is the implemented method.

Besides testing the new method with boundary value problems, as done by Polak [15] and Goldfarb and Toint [7], we apply our algorithm to solve some problems from the classical literature [11]. The numerical results show a good performance of this approach.

The paper is organized as follows. In Section 2 we introduce the general algorithm and we present a convergence result. In Section 3 we describe the implementation of the local variations method proposed. Section 4 presents our numerical experiments. Finally, in Section 5, we make some comments and present some conclusions about this work.

¿From now on, $\|\cdot\|$ denotes an arbitrary norm and $I\!N$ is the set $\{0, 1, 2, \ldots\}$.

2 Model algorithm and convergence

Assume that $F : I\!R^n \to I\!R^n$, $F \in C^1(I\!R^n)$. We denote $J(x)$, the Jacobian matrix of $F(x)$. Assume that J satisfies the Lipschitz condition

$$\|J(x) - J(y)\| \leq L\|x - y\| \ \ \forall x,\ y \in I\!R^n. \tag{2}$$

Then,

$$\|F(y) - F(x) - J(x)(y - x)\| \leq \frac{L}{2}\|x - y\|^2 \ \ \forall x,\ y \in I\!R^n. \tag{3}$$

Assume that $\sigma \in (0, 1)$, $0 < \tau_{min} < \tau_{max} < 1$, $\alpha_{-1} = 1$, $c > 0$ and that $\{\eta_k\}$ is a sequence such that

$$\eta_k > 0 \text{ for all } k \in I\!N \text{ and } \sum_{k=0}^{\infty} \eta_k = \eta < \infty. \tag{4}$$

Let $x_0 \in I\!R^n$ be an arbitrary initial point.

Given $x_k \in I\!R^n$, the kth iterate of the algorithm, the steps for obtaining x_{k+1} are given in Algorithm 1.

Algorithm 1. (Model Algorithm)

`Step 1:` Compute $B_k \in I\!R^{n\times n}$ such that

$$\|B_k - J(x_k)\| \leq c\ \alpha_{k-1}. \tag{5}$$

`Step 2:` Compute (if possible) $d_k \in I\!R^n$ such that

$$B_k d_k + F(x_k) = 0. \tag{6}$$

`Step 3:`
`step 3.1:` Set $\alpha \leftarrow 1$.
`step 3.2:` If

$$\|F(x_k + \alpha d_k)\| \leq (1 - \alpha\sigma)\|F(x_k)\| + \eta_k, \tag{7}$$

set $\alpha_k = \alpha$ and go to Step 4.

If (7) does not hold, compute $\alpha_{new} \in [\tau_{min}\alpha, \tau_{max}\alpha]$, set $\alpha \leftarrow \alpha_{new}$ and repeat step 3.2.
`Step 4:` Compute $x_{k+1} \in I\!R^n$ such that

$$\|F(x_{k+1})\| \leq \|F(x_k + \alpha_k d_k)\|. \tag{8}$$

Remarks.

1. Since $\eta_k > 0$ the condition (7) necessarily holds if α is small enough. Therefore, the iteration is well defined whenever the linear system (6) has a solution.

2. Algorithm 1 may be viewed as a Newton-like method for solving the nonlinear system $F(x) = 0$. At Step 2 the typical Newtonian linear system $B_k d = -F(x_k)$ is solved and Step 3 represents a nonmonotone line search procedure, inspired in [10]. An additional improvement of the system norm is obtained at Step 4.

3. In specific implementations, B_k will be a discretization of the Jacobian matrix $J(x_k)$. Therefore, the "error" $\|B_k - J(x_k)\|$ will be proportional to a discretization step h_k. In (5) we impose that the discretization step at iteration k must be proportional to the previous steplength α_{k-1}. The motivation for this is that "α_{k-1} small" implies that, at iteration $k-1$, the test (7) failed several times probably because B_{k-1} was not close enough to $J(x_{k-1})$. So, at iteration k we choose a small discretization step that makes B_k closer to $J(x_k)$.

4. The presence of $\eta_k > 0$ in the line search (7) makes it possible that, sometimes, $\|F(x_{k+1})\| > \|F(x_k)\|$. The direction d_k obtained in (6) might not be a descent direction for $\|F(x)\|$, therefore the iteration might not be well defined without the incorporation of η_k. Therefore, nonmonotone line searches seem to be important in the implementation of this type of algorithms.

5. After the computation of α_k, we collect information for computing a new Jacobian approximation. We do this by means of the evaluation of $F(x)$ at auxiliary points. Each time we evaluate the system at a new auxiliary point, we verify whether the system norm at the new point is smaller than at the previous one. When the cycle

of auxiliary points is completed, we possibly have a new point with a system norm smaller than $\|F(x_k + \alpha_k d_k)\|$. This new point will be called x_{k+1}. Step 4 will be the "local variations" part of the implemented algorithm. Observe that the choice of auxiliary points must obey the proportionality constraint (5).

6. The local variations procedure may help to improve the quality of the approximate solution, especially when we are far from the solution of the system. In this way, additional evaluations are more than an auxiliary device to estimate Jacobians.

In the following theorem we prove that, assuming boundedness of the inverse Jacobian, Algorithm 1 finds solutions of $F(x) = 0$ with an arbitrary precision.

Theorem 1. *Assume that $\{x_k\}$ is generated by Algorithm 1 and, for some sequence of indices $K_1 \subset \mathbb{N}$, $J(x_k)$ is nonsingular and $\|J(x_k)^{-1}\| \leq M$. Then*

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

and every limit point of $\{x_k\}$ is a solution of the system (1).

Proof. We consider two disjoint possibilities:
(i) For infinitely many indices $k \in K \subset \mathbb{N}$, we have that $\alpha_k \geq \bar{\alpha} > 0$.
(ii) $\lim_{k\to\infty} \alpha_k = 0$.
Let us analyze first Case (i). From (7) and (8),

$$\|F(x_{k+1})\| \leq (1 - \alpha_k\sigma)\|F(x_k)\| + \eta_k \tag{9}$$

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

Adding all these inequalities, we get

$$0 \leq \|F(x_0)\| + \sum_{k=0}^{\infty} \eta_k - \sigma \sum_{k=0}^{\infty} \alpha_k \|F(x_k)\|.$$

Then,

$$\sigma \sum_{k\in K} \bar{\alpha}\|F(x_k)\| \leq \sigma \sum_{k\in K} \alpha_k\|F(x_k)\| \leq \sigma \sum_{k=0}^{\infty} \alpha_k\|F(x_k)\| \leq \|F(x_0)\| + \eta.$$

Therefore,

$$\lim_{k\in K} \|F(x_k)\| = 0.$$

Given $\varepsilon > 0$, let k_0 be such that
(a) For all $k \in K$, $k \geq k_0$, $\|F(x_k)\| \leq \varepsilon/2$;
(b) $\sum_{\ell=k_0}^{\infty} \eta_\ell \leq \varepsilon/2$.

Then, if $k \geq k_0$,

$$\|F(x_k)\| \leq \|F(x_{k_0})\| + \sum_{\ell=k_0}^{k-1} \eta_\ell \leq \|F(x_{k_0})\| + \sum_{\ell=k_0}^{\infty} \eta_\ell \leq \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon.$$

Therefore,

$$\lim_{k\to\infty} \|F(x_k)\| = 0. \tag{10}$$

Consider now Case (ii). Then

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

Thus, for $k \in I\!N$ large enough, there exists

$$\alpha'_k \in [\frac{\alpha_k}{\tau_{max}} \ , \ \frac{\alpha_k}{\tau_{min}}]$$

such that

$$\lim_{k\to\infty} \alpha'_k = 0 \tag{11}$$

and

$$\|F(x_k + \alpha'_k d_k\| > (1 - \alpha'_k\sigma)\|F(x_k)\|. \tag{12}$$

Define

$$T_k = \|F(x_k + \alpha'_k d_k) - F(x_k) - J(x_k)\alpha'_k d_k\|.$$

Then, by (12),

$$T_k + \|F(x_k) + J(x_k)\alpha'_k d_k\| > \|F(x_k)\| - \alpha'_k\sigma\|F(x_k)\|.$$

So,

$$T_k + \|\alpha'_k[F(x_k) + J(x_k)d_k]\| + (1 - \alpha'_k)\|F(x_k)\| > \|F(x_k)\| - \alpha'_k\sigma\|F(x_k)\|.$$

Then by (6),

$$T_k + \alpha'_k\|B_k - J(x_k)\| \ \|d_k\| > \alpha'_k(1 - \sigma)\|F(x_k)\|.$$

So, by (5)

$$\frac{T_k}{\alpha'_k} + \alpha_{k-1}c \, \|d_k\| > (1 - \sigma)\|F(x_k)\|.$$

Then, by (3),

$$\frac{L}{2} \, \alpha'_k\|d_k\|^2 + \alpha_{k-1}c \, \|d_k\| > (1 - \sigma)\|F(x_k)\|. \tag{13}$$

Now, for $k \in K_1$, $\|J(x_k)^{-1}\| \leq M$. But, by (5),

$$\lim_{k\to\infty} \|B_k - J(x_k)\| = 0;$$

So, for $k \in K_1$ large enough,

$$\|B_k^{-1}\| \leq 2M.$$

So, since

$$\|F(x_k)\| \leq \|F(x_0)\| + \eta,$$

we have that $\|d_k\|$ is bounded for $k \in K_1$, large enough.

Then, by (11) and (13),

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

As in the deduction of (10), this implies that

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

So, every limit point must be a solution. □

Remark.

Theorem 1 shows that the only reason why the algorithm can fail is when it is not well defined (perhaps because of singularity of the Hessian) or unboundedness of $\|J(x_k)^{-1}\|$. In particular, Theorem 1 implies that, if x_* is a limit point but not a solution, the Jacobian $J(x_*)$ is necessarily singular.

Now we introduce three variations of the original algorithm related to the choice of x_{k+1} at Step 4, to the approximation of B_k with respect to $J(x_k)$ and to the nonmonotonicity sequence η_k. We present these variations as assumptions on the algorithmic sequence.

Assumption 1. There exists $c' > 0$ such that, for all $k \in I\!N$,

$$\|x_{k+1} - x_k\| \leq c' \alpha_k \|d_k\|. \tag{14}$$

Assumption 2. For all $k \in I\!N$,

$$\|B_k - J(x_k)\| \leq c \min\{\alpha_0, \ldots, \alpha_{k-1}\}. \tag{15}$$

Assumption 3. There exists a sequence $\{\gamma_k\}$ such that $\lim_{k \to \infty} \gamma_k = 0$ and, for all $k \in I\!N$,

$$\eta_k \leq \gamma_k \|F(x_k)\|. \tag{16}$$

(Observe that (4) must also be satisfied by $\{\eta_k\}$.)

When we implement Algorithm 1 as a discrete Newton method with local variations, the interpretation Assumption 1 is that the distance between successive points is proportional to the damped Newton step $\|\alpha_k d_k\|$. In other words, changes due to Newton steps are of the same order as those related to local variations. This assumption has strong

consequences in local convergence. If Assumption 1 does not hold, oscillation between different solutions is possible.

Assumption 3 is easy to satisfy in practice with a convenient choice of η_k. This assumption guarantees monotone behavior of $\|F(x_k)\|$ near a regular solution, as will be shown later.

The motivation of Assumption 2 is not so simple. In practical computations it is not desirable to compute discrete derivatives using very small discretization steps because of the possibility of severe cancellation errors. According to (5), however, a very small discretization step might appear if α_{k-1} is very small. Therefore, it is interesting to investigate under which assumptions we can guarantee that the steplengths α_k are bounded away from zero. We will see that this happens when we choose the discretization steps according to the rule (15).

Theorem 2. *Assume that $\{x_k\}$ is generated by Algorithm 1, $\|J(x_k)^{-1}\| \leq M$ for all $k \in \mathbb{N}$ and Assumption 2 holds. Then, there exists $\bar{\alpha} > 0$ such that $\alpha_k \geq \bar{\alpha}$ for all $k \in \mathbb{N}$. Moreover, if Assumption 3 is satisfied, then for all $r \in (1 - \sigma\bar{\alpha}, 1)$, there exists $k_0 \in \mathbb{N}$ such that*

$$\|F(x_{k+1})\| \leq r\|F(x_k)\| \quad \textit{for all} \quad k \geq k_0. \tag{17}$$

Proof. Assume, by contradiction, that K_2 is an infinite sequence of indices such that

$$\lim_{k \in K_2} \alpha_k = 0.$$

Then, by (15),

$$\lim_{k \to \infty} \|B_k - J(x_k)\| = 0.$$

Therefore, for k large enough, B_k^{-1} exists and

$$\|B_k^{-1}\| \leq 2M.$$

Then, by (6),

$$\|d_k\| \leq 2M\|F(x_k)\| \tag{18}$$

By Theorem 1, $\lim_{k \to \infty} \|F(x_k)\| = 0$. So, by (18),

$$\lim_{k \to \infty} \|d_k\| = 0.$$

Now, by (3), (6) and (18),

$$\begin{aligned} &\|F(x_k + d_k)\| \\ &= \|F(x_k + d_k) - F(x_k) - B_k d_k + (F(x_k) + B_k d_k) - J(x_k)d_k + J(x_k)d_k\| \\ &\leq \|F(x_k + d_k) - F(x_k) - J(x_k)d_k\| + \|(B_k - J(x_k))d_k\| \\ &\leq \frac{L}{2}\|d_k\|^2 + \|B_k - J(x_k)\|\,\|d_k\| \end{aligned}$$

$$\leq [2M^2L\|F(x_k)\| + 2M\|B_k - J(x_k)\|]\,\|F(x_k)\|.$$

Since $\|F(x_k)\| \to 0$ and $\|B_k - J(x_k)\| \to 0$, the previous inequality implies that (7) holds with $\alpha = 1$ for k large enough. Therefore, for k large enough, $\alpha_k = 1$. This contradicts the initial assumption. Therefore, α_k is bounded away from zero, as we wanted to prove.

Let us assume now that Assumption 3 holds. Then, by (7),

$$\begin{aligned}\|F(x_{k+1})\| &\leq (1 - \bar{\alpha}\sigma)\|F(x_k)\| + \eta_k \\ &\leq (1 - \bar{\alpha}\sigma)\|F(x_k)\| + \gamma_k\|F(x_k)\| \\ &= (1 - \bar{\alpha}\sigma + \gamma_k)\|F(x_k)\|\end{aligned}$$

for all $k \in I\!N$. Since $\lim_{k\to\infty} \gamma_k = 0$, (17) holds for k large enough. □

Counterexample. It is interesting to show that α_k might not be bounded away from zero under the rule (5), even in situations in which the sequence converges to an isolated solution of (1). Define $F : I\!R \to I\!R$ by $F(x) = x^2 - 1$, $x_0 = -2$, $B_k = J(x_k) = 2x_k$ if k is even, $B_k = 1$ if k is odd. Take, for example, $c = 10$, $\sigma = 1/2$ and $\eta_k = 1/(k+1)^2$. When (7) fails we divide α by 2.

Clearly,

$$\lim_{k\to\infty} x_k = -1,$$

which is a solution of the equation.

The first iterates of the algorithm are:

$$x_0 = -2.000000000000000, F(x_0) = 3.000000000000000,$$

$$\begin{aligned}x_1 &= -1.250000000000000, F(x_1) = 5.625000000000000 \times 10^{-1}, \\ x_2 &= -1.320312500000000, F(x_2) = 7.432250976562500 \times 10^{-1}, \\ x_3 &= -1.038854474852071, F(x_3) = 7.921861992017210 \times 10^{-2}, \\ x_4 &= -1.058659129832114, F(x_4) = 1.207591531768887 \times 10^{-1}, \\ x_5 &= -1.001625118707098, F(x_5) = 3.252878425008438 \times 10^{-3}, \\ x_6 &= -1.004877997132107, F(x_6) = 9.779789120234205 \times 10^{-3}, \\ x_7 &= -1.000011839674114, F(x_7) = 2.367948840588374 \times 10^{-5}, \\ x_8 &= -1.000035519162520, F(x_8) = 7.103958665064674 \times 10^{-5}, \\ x_9 &= -1.000000000630783, F(x_9) = 1.261565963240480 \times 10^{-9}, \\ x_{10} &= -1.000000001892349, F(x_{10}) = 3.784697891998264 \times 10^{-9}.\end{aligned}$$

The even iterations are Newton iterations for which (7) holds with $\alpha_k = 1$. Therefore, the inequality (5) is

$$\|B_k - J(x_k)\| \leq 10$$

when k is odd. Therefore, (5) is satisfied when we choose $B_k = 1$ at odd iterations. However, the direction chosen at odd iterations is an ascent direction, which makes that the steplengths at odd iterations tend to zero. Namely,

$$\alpha_{2k} = 1 \ \ \forall \ \ k \in I\!N,$$

but

$$\lim_{k\to\infty} \alpha_{2k+1} = 0.$$

This shows that the rule (15) is necessary for proving the existence of a lower bound of α_k. With this rule, the choice $B_k = 1$ at odd iterations would not be admissible.

Theorems 1 and 2 did not use Assumption 1. This assumption, however, is necessary to prove local convergence of the algorithm, as we show in Theorem 3 below.

Theorem 3. *Assume that $\{x_k\}$ is generated by Algorithm 1 and Assumptions 1, 2, 3 are satisfied. In addition, suppose that x_* is a limit point of $\{x_k\}$,*

$$\lim_{k\in K_3} x_k = x_*,$$

$J(x_*)$ is nonsingular and $\{\|B_k^{-1}\|\}_{k\in I\!N}$ is bounded. Then, x_k converges to x_* and $F(x_*) = 0$. Moreover, there exists $\bar{\alpha} > 0$ such that $\alpha_k \geq \bar{\alpha}$ for all $k \in I\!N$ and $\{x_k\}$ converges at a linear rate to x_* in the sense that for all $r \in (1 - \sigma\bar{\alpha}, 1)$ there exists $k_1 \in I\!N$ such that

$$\|J(x_*)(x_{k+1} - x_*)\| \leq r\|J(x_*)(x_k - x_*)\| \tag{19}$$

for all $k \geq k_1$.

Proof. By Theorem 1 we have that

$$\lim_{k\to\infty} \|F(x_k)\| = 0 \tag{20}$$

and

$$F(x_*) = 0.$$

By the boundedness of $\{\|B_k^{-1}\|\}_{k\in I\!N}$, (6), (14) and (20), we have that

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

Since $J(x_*)$ is nonsingular, by the Inverse Function Theorem, there exists $\varepsilon > 0$ such that $\|F(x)\| > 0$ whenever $x \neq x_*$ and $\|x - x_*\| \leq \varepsilon$. Let k_0 be such that

$$\|x_{k+1} - x_k\| \leq \frac{\varepsilon}{2} \tag{21}$$

for all $k \geq k_0$. The set $\{x \in I\!R^n \mid \frac{\varepsilon}{2} \leq \|x - x_*\| \leq \varepsilon\}$ can contain only a finite number of iterates x_k. Otherwise, it should contain a limit point that, by Theorem 1, should be a solution of (1). Therefore, there exists $k_1 \geq k_0$ such that, for all $k \geq k_1$, either $\|x_k - x_*\| < \varepsilon/2$ or $\|x_k - x_*\| > \varepsilon$. But, since there exist infinitely many iterates such that $\|x_k - x_*\| < \varepsilon/2$ and (21) holds for all $k \geq k_1$, it follows that $\|x_k - x_*\| < \varepsilon/2$ must hold for all k large enough. Since x_* is the only possible limit point in this ball, it follows that x_k converges to x_*.

By Theorem 2, if (15) holds, there exists $\bar{\alpha} > 0$ such that $\alpha_k \geq \bar{\alpha}$ for all $k \in I\!N$. Let $r \in (1 - \sigma\bar{\alpha}, 1)$, $r' \in (1 - \sigma\bar{\alpha}, r)$. By Theorem 2, for k large enough,

$$\|F(x_{k+1}\| \leq r'\|F(x_k)\|. \tag{22}$$

By (3), since $F(x_*) = 0$,

$$\|F(x_{k+1}) - J(x_*)(x_{k+1} - x_*)\| \leq \frac{L}{2}\|x_{k+1} - x_*\|^2.$$

So,

$$\|J(x_*)(x_{k+1} - x_*)\| - \|F(x_{k+1})\| \leq \frac{L}{2}\|x_{k+1} - x_*\|^2$$

and, by (22),

$$\|J(x_*)(x_{k+1} - x_*)\| - \frac{L}{2}\|x_{k+1} - x_*\|^2 \leq \|F(x_{k+1})\| \leq r'\|F(x_k)\|. \tag{23}$$

But, also by (3),

$$\|F(x_k)\| \leq \|J(x_*)(x_k - x_*)\| + \frac{L}{2}\|x_k - x_*\|^2,$$

so, by (23),

$$\|J(x_*)(x_{k+1} - x_*)\| - \frac{L}{2}\|x_{k+1} - x_*\|^2 \leq r'\|J(x_*)(x_k - x_*)\| + \frac{r'L}{2}\|x_k - x_*\|^2.$$

So,

$$\|J(x_*)(x_{k+1} - x_*)\| - \frac{L}{2}\|J(x_*)^{-1}\|\,\|J(x_*)(x_{k+1} - x_*)\|\,\|x_{k+1} - x_*\|$$

$$\leq r'\|J(x_*)(x_k - x_*)\| + \frac{Lr'}{2}\|J(x_*)^{-1}\|\,\|J(x_*)(x_k - x_*)\|\|x_k - x_*\|.$$

Therefore,

$$(1 - \frac{L}{2}\|J(x_*)^{-1}\|\,\|x_{k+1} - x_*\|)\|J(x_*)(x_{k+1} - x_*)\|$$

$$\leq (r' + \frac{Lr'}{2}\|J(x_*)^{-1}\|\,\|x_k - x_*\|)\|J(x_*)(x_k - x_*)\|.$$

Thus,

$$\|J(x_*)(x_{k+1} - x_*)\| \leq Q_k r'\|J(x_*)(x_k - x_*)\|$$

where

$$Q_k = \frac{1 + \frac{L}{2}\|J(x_*)^{-1}\|\,\|x_k - x_*\|}{1 - \frac{L}{2}\|J(x_*)^{-1}\|\,\|x_{k+1} - x_*\|}.$$

Since $\lim_{k\to\infty} Q_k = 1$, the desired result is proved. □

Remark.

A simple adaptation of classical results (see, for example, Theorem 8.2.4 of [4]) shows that, in Algorithm 1 with the hypothesis of Theorem 3, superlinear convergence is obtained if $\|B_k - J(x_k)\| \to 0$. One only needs to prove that (7) holds with $\alpha = 1$ for k large enough, which is straightforward. In our context, and having in mind the discrete Newton application, the hypothesis $\|B_k - J(x_k)\| \to 0$ would be linked to small discretization steps, which are not practical because of cancellation.

3 Discretization and local variations

Our aim will be to define the matrices B_k as Jacobian discretizations. We do not want to use very small discretization steps in order to avoid cancellation due rounding errors as much as possible. On the other hand, the discretization step must be small enough ($\|B_k - J(x_k)\|$ must be small enough) for guaranteing convergence. The theorems in the previous section give a theoretical basis to the choice $h = O(\alpha_{k-1})$ of the discretization step. A second reason for choosing not very small (and proportional to the steplength) discretization steps is that we want to take advantage of auxiliary points for further decrease of $\|F(x)\|$. This is the "local variations" instance of our algorithm.

Assume that $J(x)$, the Jacobian matrix of $F(x)$, has a sparse structure. The standard discretization of a Jacobian considers $J(x)$ as a full matrix and has to perform $n + 1$ function evaluations for computing the approximation. For each j, $\quad j = 1, 2, \ldots, n$, the jth column of the approximation of $J(x)$ is given by:

$$(\hat{J})_j = \frac{F(x + he_j) - F(x)}{h}, \tag{24}$$

where $h \in I\!R,\ h \neq 0$, is the step size and e_j is the jth vector of the canonical basis of $I\!R^n$.

For the case in which $J(x)$ is sparse, Curtis, Powell and Reid [3] proposed an algorithm to evaluate the approximate Jacobian matrix with a reduced number of function evaluations (see [3]). The idea is to update groups of columns of $\hat{J}$ that could be evaluated together, using only one function evaluation for each group. The greedy CPR strategy to deal with problems where the Jacobian matrix has a known sparse structure is based on generating the groups, the first one always starting with the first column, and then going in the natural order of the columns. The only requirement is that two different columns belonging to a CPR-group must have all its nonzero elements on different rows. Those groups are also called CPR-valid in the literature.

The greedy choice of CPR-groups suggested in [3] does not always provide the least possible number of groups (see, for instance [12, 2, 7]). Coleman and Moré [2] showed that this problem, for a general sparse pattern, is equivalent to a certain coloring problem on a suitable graph, and proposed the use of graph coloring algorithms to obtain less groups then CPR method. But, again, not always that this strategy provides the minimum number of groups.

Newsan and Ramsdell [12] proved that it is always possible to estimate the sparse Jacobian matrix using a number of function evaluations equal to the maximum number of nonzero elements on a single row.

Goldfarb and Toint [7] also used the CPR idea to solve nonlinear sparse systems coming from the discretization of boundary value problems in partial differential equations, by finite differences. With their strategy, they perform a number of function evaluations that is equal to the number of CPR-groups plus one; and the number of groups that they need to use is the maximum number of nonzero elements on a single row.

In this work, we use the strategy of Goldfarb and Toint for boundary value problems and the greedy CPR strategy for general problems. It is not hard to see that, if we use the information about how the grid was constructed, the CPR-groups can easily be identified [7].

Our algorithm is a particular case of Algorithm 1, described in the previous section. In this section we describe how to compute the iterate x_{k+1}, starting from $x_k + \alpha_k d_k$ and how to compute, at the same time, the new matrix B_{k+1}. The first procedure is the local variations method and the second is the discretization scheme. Details (including the way to compute the first iteration) are left to the following section.

To simplify the notation, let us define $y_1 = x_k + \alpha_k d_k$. Initially, y_1 will be called "base point" of the discretization. Let $h_{k+1} \neq 0$ be a discretization step, such that

$$|h_{k+1}| \leq \beta \alpha_k \tag{25}$$

where $\beta > 0$ is a parameter independent of k.

Assume that we have q CPR-groups. To each of them it is associated a $0-1$ vector v_j in such a way that the evaluation $F(x + hv_j)$, together with $F(x)$, allows one to estimate all the columns of the Jacobian belonging to the $j-$th CPR-group. The $i-$th entry of v_j is equal to 1 if the $i-$th column belongs to the $j-$th group and 0 otherwise. For example, in the classical discrete Newton method, without sparsity, there are n CPR groups and each one contains only one column. In this case the vectors v_j are those of the canonical basis of $\mathbb{R}^n$. The vectors v_j are called "CPR directions".

Assume that $\{v_1, \ldots, v_q\} \subset \mathbb{R}^n$ is the set of (nonnull) CPR directions, which is also independent of the iteration index k.

The following algorithm describes how to obtain x_{k+1}.

Algorithm 2.

For $j = 1, \ldots, q$, execute Steps 1 to 3.

`Step 1.` If $\langle d_k\,,\, v_j \rangle \leq 0$, set $w_j = -v_j$; else, set $w_j = v_j$.

`Step 2.` Compute $z \leftarrow y_j + h_{k+1} w_j$.

`Step 3.` If

$$\|F(z)\| < \|F(y_j)\| \tag{26}$$

set $y_{j+1} = z$; else set $y_{j+1} = y_j$.

`Step 4.` Define $x_{k+1} = y_{q+1}$.

At the beginning of the algorithm, when we have an initial x_0 given externally, we apply Steps 1 and 2 of Algorithm 2 starting from $y_1 = x_0$ and, at Step 4, we redefine $x_0 \leftarrow y_{q+1}$.

Clearly, by (25) and (26) the conditions (14) and (8) are satisfied.

Let us write, to simplify the notation, $h = h_{k+1}$. In the CPR scheme, for all $j = 1, \ldots, q$, we have that:

$$B_{k+1} w_j = [F(y_j + h w_j) - F(y_j)]/h. \tag{27}$$

These equations characterize B_{k+1} in the sense that B_{k+1} is the only matrix with the given structure that satisfies (27) for all $j = 1, \ldots, q$.

Now, by (3),

$$\|F(y_j + h w_j) - F(y_j) - J(y_j) h w_j\| \leq \frac{L}{2} h^2 \|w_j\|^2.$$

Therefore,

$$\|\frac{F(y_j + h w_j) - F(y_j)}{h} - J(y_j) w_j\| \leq \frac{L}{2} |h| \, \|w_j\|^2.$$

So, by (27),

$$\|[B_{k+1} - J(y_j)] w_j\| \leq \frac{L}{2} |h| \, \|w_j\|^2.$$

By the definition of w_j this implies that there exists a constant c_1, independent of the iteration index k, such that

$$\|B_{k+1} - J(y_j)\| \leq c_1 |h|. \tag{28}$$

But, by the definition of y_j we have that

$$\|y_j - x_{k+1}\| \leq c_2 |h|,$$

therefore, by (2),

$$\|J(y_j) - J(x_{k+1})\| \leq c_2 L |h|.$$

Then, by (28),

$$\|B_{k+1} - J(x_{k+1})\| \leq (c_2 L + c_1) |h|.$$

So, by the choice (25), the condition (5) holds. Clearly, if we choose, instead of (25),

$$|h_{k+1}| \leq \beta \min\{\alpha_0, \ldots, \alpha_k\} \tag{29}$$

the assumption (15) is satisfied as well.

4 Implementation features

Algorithm 1 was implemented with the discrete Newton definition of B_k described in the previous section and the local variations procedure given by Algorithm 2 for defining x_{k+1}, after the computation of $x_k + \alpha_k d_k$. The theorems proved in Section 2 give the theoretical properties of the algorithm for the choices (25) and (29) of the discretization step h_{k+1}.

In this section we give more details about the implementation of the algorithm.

From now on, $||\,.\,||$ means the Euclidean norm.

4.1 The choice of the discretization step

Given $smin$ and $smax$ such that $0 < smin < smax < \infty$ we defined

$$s_0 = smax$$

and

$$s_k = \min\{smax, \max\{smin, \|d_k\|\}\}, \text{ if } k > 0.$$

Finally,

$$|h_{k+1}| = \min\{\alpha_0, \ldots, \alpha_k\} s_k, \quad k \in I\!N.$$

We used $smin = \sqrt{\varepsilon_{mach}}$, where ε_{mach} is the machine precision; $smax$ will be defined in the next section.

According to this choice, observe that Assumption 2 is satisfied.

4.2 Line search procedure

If the vector $x_k + \alpha_k d_k$ does not give an acceptable decrease in the value of the function, in the sense of (7), then we compute the new step size as $\alpha_{new} = \alpha/2$. For the parameter σ used in the criterion (7), we took $\sigma = 10^{-4}$.

4.3 The sequence η_k

We define:

- $ftip(0) = \|F(x_0)\|$,
- $ftip(k) = \min\{\|F(x_k)\|, ftip(k-1)\}$, if k is a multiple of 10 and
- $ftip(k) = ftip(k-1)$, otherwise.

Then, we set:

$$\eta_k = \frac{ftip}{(k+1)^{1.1}}.$$

4.4 Stopping criteria

The process is finished successfully if

$$\|F(x_k)\| \leq 10^{-6} \text{ and } k < 500.$$

5 Numerical Experiments

In order to test the new algorithm proposed in this work we implemented also the discrete Newton algorithm, where the approximation of the Jacobian matrix is obtained by groups. Let us describe now this algorithm.

Given:

- q, the number of CPR-valid groups for the Jacobian matrix;
- $I_j,\ j = 1, \ldots q$, the vectors of the indices of the columns at the group q;
- str, the array which contains the sparse structure of Jacobian matrix: $(i, j) \in str$ if the (i, j) entry of Jacobian is nonzero;
- $\varepsilon > 0$, the tolerance for the stopping criterion;
- $x_0 \in I\!R^n$, the initial approximation for the solution of (1);
- $h_\varepsilon > 0$, the finite-difference step size.

Let $x_k \in I\!R^n$ be the kth iterate of the algorithm. Then the steps for obtaining x_{k+1} are given as follows:

Algorithm 3.

`Step 1:` While $\|F(x_k)\| > \varepsilon$ perform Steps 2 to 4.

`Step 2:` Evaluate the approximation of the Jacobian matrix:

For $gcol = 1, \ldots, q$

For all $j \in I_{gcol}$ and i such that $(i, j) \in str$:
compute: $\hat{J}_{i,j} = ((F(x_k + h_\varepsilon v_j))_i - (F(x_k))_i)/h_\varepsilon$,
where the vector v_j is the CPR direction.

`Step 3:` Compute the direction d, solution of: $\hat{J}d = -F(x_k)$.

`Step 4:` Set: $x_{k+1} = x_k + d$ and $k = k + 1$.

We ran both algorithms with the same parameters as described in the last section. All the tests were performed in an Pentium III - 1.0GHz computer, using the software MatLab 6.0.

5.1 Academic Tests

The first set of numerical experiments consists of 12 problems selected from Moré, Garbow and Hillstrom [11] collection.

The results are presented in Table 1 with the following notation:

- `(Problem, n, q)` denotes the name of the nonlinear system, its dimension and number of CPR-valid groups of the Jacobian matrix, respectively;
- `Algorithm`: `DN` indicates the discrete Newton method (algorithm 3) and `DNLV` indicates the discrete Newton method with local variations (algorithms 1 and 2);
- δ denotes the initial value for step size for discretization of matrix $\hat{J}$: for `DN` algorithm, this value is fixed for all iterations and $\delta = h_\varepsilon = \sqrt{\varepsilon_{mach}}\,\|x_0\|_\infty$ (if $\|x_0\|_\infty = 0$ then we chose $\sqrt{\varepsilon_{mach}}$); for `DNLV`, $\delta = smax$;
- `Conv`: `C` indicates that the stopping criterion was satisfied for one approximation x_k, and `NC1` indicates that the maximum number of iterations was exceeded and `NC2` means non convergence with `normf = Nan (non numeric value)`;
- `(Iter, Feval)` denotes the number of iterations and the number of function evaluations performed by the algorithm; according to the `DN` algorithm the number of function evaluations is given by the formula:

  ```
  ((q+1)*iter + 1)
  ```

  and for `DNLV` algorithm, this number will be given by same formula plus the number of function evaluations performed at line search steps and
- $\|F(x)\|_2$ indicates the norm-2 of the function at the solution obtained by the algorithm.

We observe that, for some problems, the performance of `DNLV` could be better if a more tolerant line search process had been used. For example, using $\eta_k = 10^{39}$ at the three first iterations, the performance of `DNLV` is the same of `DN` method for Rosenbrock problem. But this tolerant line search resulted in a worse perfomance for `DNLV` at other tests. So we fixed the strategy indicated in algorithm 3 for all the tests performed in this work.

Comparing the performance of `DNLV` using different choices for the parameter δ we concluded that the best choice is $\delta = 0.02$. This choice was made because despite of better results were obtained with $\delta = 0.2$ or $\delta = 0.7$ for a few problems, the algorithm with $\delta = 0.02$ had a more robust performance.

To illustrate a comparison between `DN` and `DNLV` (with $\delta = 0.02$) algorithms, we plotted, in the same figure, the number of iterations performed by these methods at each problem numbered from 1 to 11 according to the order that they appear in Table 1. A similar comparison was done using the number of function evaluations. These results are showed in Figure 1, where the symbols $+$ and $\diamondsuit$ represents `DNLV` and `DN` methods respectively.

In five problems (problems 2, 4, 7, 8, and 11) both algorithms had the same performance and in four problems (3, 6, 9 and 10) the two algorithms had a similar performance in terms of number of iterations: `DNLV` performed just one more iteration than `DN`, but the difference between the number of function evaluations is more significant, because at each iteration this number is equal to the the number of CPR-valid groups plus one.

(Problem,n,ngroup)	Algorithm	δ	Conv.	(Iter, Evalf)	$\|F(x)\|_2$
(Rosenbrock,2,2)	DN	$\sim$ 1.5D−08	C	(2,7)	0.155D−13
	DNLV	0.02	C	(5,21)	0.133D−13
		0.2	C	(5,21)	0.222D−14
		0.7	C	(7,28)	0.000D+00
(Powell badly scaled,2,2)	DN	$\sim$ 1.5D−08	C	(10,31)	0.358D−06
	DNLV	0.02	C	(10,31)	0.594D−06
		0.2	C	(11,34)	0.561D−06
		0.7	C	(13,40)	0.405D−07
(Helical valley,3,3)	DN	$\sim$ 1.5D−08	C	(9,37)	0.616D−07
	DNLV	0.02	C	(10,41)	0.653D−12
		0.2	C	(9,37)	0.419D−07
		0.7	C	(7,29)	0.343D−06
(Box three-dimensional,3,3)	DN	$\sim$ 2.9D−07	C	(4,17)	0.317D−08
	DNLV	0.02	C	(4,17)	0.445D−06
		0.2	C	(5,21)	0.811D−08
		0.7	C	(5,21)	0.618D−09
(Powell singular,4,2)	DN	$\sim$ 4.5D−08	C	(12,37)	0.756D−06
	DNLV	0.02	C	(17,52)	0.869D−06
		0.2	C	(20,61)	0.479D−06
		0.7	C	(19,58)	0.989D−06
(Trigonometric,10,10)	DN	1.5D−09	C	(7,78)	0.801D−11
	DNLV	0.02	C	(8,95)	0.506D−07
		0.2	NC1	(500,10057)	0.113D−00
		0.7	NC1	(500,9386)	0.580D−02
(Brown almost-linear,50,50)	DN	$\sim$ 7.5D−09	C	(1,52)	0.123D−13
	DNLV	0.02	C	(1,52)	0.286D−11
		0.2	C	(1,52)	0.502D−13
		0.7	C	(1,52)	0.100D−12
(Discrete boundary value, 100,3)	DN	$\sim$ 1.1D−09	C	(2,9)	0.108D−08
	DNLV	0.02	C	(2,9)	0.196D−07
		0.2	C	(2,9)	0.463D−06
		0.7	C	(3,13)	0.136D−07
(Broyden tridiagonal,100,3)	DN	$\sim$ 1.5D−08	C	(4,17)	0.106D−08
	DNLV	0.02	C	(5,21)	0.583D−09
		0.2	C	(5,21)	0.255D−07
		0.7	C	(6,25)	0.475D−06
(Broyden banded,100,7)	DN	$\sim$ 1.5D−08	C	(5,41)	0.154D−07
	DNLV	0.02	C	(6,49)	0.144D−07
		0.2	C	(7,57)	0.344D−07
		0.7	NC2	(13,16)	NaN
(Discrete integral equation,50,50)	DN	$\sim$ 3.7D−09	C	(2,103)	0.768D−06
	DNLV	0.02	C	(2,103)	0.937D−06
	DNLV	0.2	C	(3,154)	0.527D−06
	DNLV	0.7	C	(3,154)	0.457D−07

Table 1: First set of numerical tests

5.2 Bratu and Convection-Diffusion Problems

The problems considered in this section consist on finding $u : [0,1] \times [0,1] \to I\!R$ such that

$$G_\lambda(u) \;=\; f(s,t), \tag{30}$$

with boundary conditions, where G is an operator that involves second-order partial derivatives of u. The real parameter λ and the two-variable function f define different instances of this problem. As in [9], we assumed the following known solution for the problem:

$$u_*(s,t) \;=\; 10st(1-s)(1-t)e^{s^{4.5}}. \tag{31}$$

and we computed f in such a way that u_* is a solution.

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 (30) the function and the derivatives by their approximations, and using the boundary conditions, we obtain a nonlinear system of equations like (1), with dimension 3969 (the total number of grid points).

We ran the `DN` and `DNLV` algorithms with the initial approximation $x_0 = 0$, and we took $\delta = 0.02$ for `DNLV`. This value was the one with the best performance among all the tests showed in Table 1 (the best in 10 problems). The other parameters were the same used for the academic tests.

In what follows, we define the operators considered in this work. In all the cases, the boundary condition is $u = 0$ and Δ is the Laplacian operator. The number of groups is always $q = 5$. In Tables 2 and 3, we used the same notation as the one used in Table 1 and the last column was introduced shown the CPU time, in seconds.

1. Bratu Problem

$$G_\lambda(u) \;=\; -\Delta u + \lambda e^u.$$

In Table 2, we show the results obtained when the algorithms `DN` and `DNLV` were applied for Bratu problem with several values of λ. For this formulation of the problems, only negative values of λ have physical meaning; for these values the performance of both algorithms were the same as showed in Table 2 for $\lambda = -100$ and $\lambda = -50$.

Positive values of λ make the problems mathematically more difficult and we used some of them to compare the performance of the algorithms for solving harder problems. For $\lambda = 20\,,\ 50\,,\ 60\,,\ 100$ and 500 the algorithm `DN` did not converge, while the `DNLV` obtained the solution of the system for all the values of λ.

When both methods converged, the best performance of `DN` was for $\lambda = 400$ and the best performance of `DNLV`, for $\lambda = 25$. For the other problems `DN` was slightly better than `DNLV`.

λ	Algorithm	Conv.	(Iter, Evalf)	$\|F(x)\|_2$	Time(s)
-100	DNVL	C	(6,37)	0.444D-09	5.66
	DN	C	(5,31)	0.352D-10	2.80
-50	DNLV	C	(6,37)	0.415D-10	5.60
	DN	C	(5,31)	0.342D-10	2.86
0	DNLV	C	(1,7)	0.883D-10	0.93
	DN	C	(1,7)	0.661D-10	0.61
20	DNLV	C	(7,46)	0.556D-08	6.54
	DN	NC2	(5,31)	NaN	4.78
25	DNLV	C	(6,38)	0.304D-06	5.49
	DN	C	(7,43)	0.256D-06	6.87
50	DNLV	C	(10,65)	0.172D-06	9.45
	DN	NC2	(11,67)	NaN	10.16
60	DNLV	C	(13,81)	0.306D-07	12.08
	DN	NC2	(18,109)	NaN	17.25
75	DNLV	C	(8,49)	0.195D-06	7.31
	DN	C	(6,37)	0.425D-10	5.82
100	DNVL	C	(10,63)	0.125D-09	9.45
	DN	NC2	(8,49)	NaN	7.36
150	DNLV	C	(8,49)	0.183D-07	7.58
	DN	C	(6,37)	0.149D-06	5.82
200	DNLV	C	(11,67)	0.256D-07	10.38
	DN	C	(6,37)	0.176D-06	5.82
300	DNLV	C	(9,55)	0.120D-06	8.57
	DN	C	(6,37)	0.255D-08	5.88
400	DNLV	C	(97,779)	0.918D-07	95.68
	DN	C	(7,43)	0.228D-09	6.82
500	DNLV	C	(60,554)	0.804D-06	60.86
	DN	NC2	(18,109)	NaN	17.31

Table 2: Bratu Problem.

2. Convection-Diffusion Problem

$$G_\lambda(u) \;=\; -\Delta u + \lambda u(u_s + u_t)$$

It is shown, in Table 3, the results obtained for the convection-diffusion problem for both `DN` and `DNLV` methods. Again, we worked with different values for the parameter λ.

About these results, we can observe that the `DN` method did not converge when we took λ equal to ± 200, ± 150, ± 100 , and the new algorithm, `DNLV`, was always successful.

In the tests where both methods converged, we can say that they presented almost the same performance.

λ	Algorithm	Conv.	(Iter, Evalf)	$\|F(x)\|_2$	Time(s)
-200	DNLV	C	(52,571)	0.751D-06	64.20
	DN	NC2	(17,103)	NaN	17.85
-150	DNLV	C	(57,623)	0.350D-10	70.30
	DN	NC2	(24,145)	NaN	26.37
-100	DNLV	C	(23,216)	0.927D-07	26.91
	DN	NC2	(24,145)	NaN	26.42
-75	DNLV	C	(19,161)	0.350D-10	21.48
	DN	C	(11,67)	0.363D-10	10.98
-50	DNLV	C	(10,71)	0.343D-10	10.76
	DN	C	(9,55)	0.347D-10	8.95
-25	DNLV	C	(6,37)	0.123D-08	6.15
	DN	C	(6,37)	0.123D-08	5.82
25	DNLV	C	(5,31)	0.447D-06	5.16
	DN	C	(5,31)	0.445D-06	4.89
50	DNLV	C	(8,53)	0.331D-10	8.41
	DN	C	(9,66)	0.957D-06	9.72
75	DNLV	C	(9,66)	0.957D-06	9.72
	DN	C	(10,61)	0.465D-09	10.05
100	DNLV	C	(14,116)	0.638D-07	15.71
	DN	NC2	(28,169)	NaN	30.32
150	DNLV	C	(19,176)	0.305D-06	22.13
	DN	NC2	(27,163)	NaN	28.89
200	DNLV	C	(35,366)	0.344D-10	42.63
	DN	NC2	(19,115)	NaN	20.16

Table 3: Convection-Diffusion Problem

With the objective of comparing and analyzing the performance of the solvers `DN` and `DNLV` we applied the "performance profile" tool indroduced by Dolan and Moré, [5]. This tool compares the performance of n_s solvers of a set S for the resolution of n_p problems

of a set P using a measure like the number of iterations, the number of function evaluations or the computing time. $m_{s,p}$ denotes the total of the measure chosen required to solve problem p by solver s. For each problem p and solver s the *performance ratio* $r_{s,p}$ is computed:

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

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, two points give us very important information, which are: $\rho_s(1)$ and $\bar{t}$, such that, $\rho_s(\bar{t}) = 1$. The value of $\rho_s(1)$ indicates the probability of solver s be the best solver in terms of set S and using the measure $m_{s,t}$. The efficiency of 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 $\rho_s(\bar{t}) = 1$, if there exists such value for $t < r_M$. So, the winner in terms of robustness will be the solver $\hat{s}$ for which $\overline{t_{\hat{s}}} = \min\{\bar{t}_s, \forall s \in S\}$.

We performed this analysis, considering the 25 problems listed at Tables 2 and 3 (13 Bratu problems and 12 convection-diffusion problems), the two algorithms `DN` and `DNLV` and the number of iterations as the measure of performance. We plotted at Figure 3 the function $\rho_s : [1, 20] \to [0, 1]$ for both solvers. From this figure we observe that `DNLV` solves approximately 70% of the problems with the minimum number of iterations and this solver get $\rho_s(t) = 1$ for $\bar{t} \sim 14$. The algorithm `DN` solves approximately 45% of the problems with the minimum number of iterations, and only 55% of the problems can be solved by this software. So, for this set of problems, the solver `DNLV` has the best "performance profile" in terms of minimum number of iterations and robustness.

6 Conclusions

Analyzing the Tables 1–3, we can conclude that the new algorithm `DNLV` is competitive. Specially for the boundary value problems tested, the performance of `DNLV` is much better than that of the discrete Newton's method implemented in `DN`, taking into account the number of problems solved by `DNLV` which `DN` could not solve. This can be seen in Tables 2 and 3. We observe that this conclusion can also be taken from the analysis made of the performance of both methods, considering their performance profiles (using the number

of iterations as measure, see Figure 3). `DNLV` solved all the problems, 70% of them with the minimum number of iterations, while `DN` solved only 55% of the problems and only 45% of them were solved with the minimum number of iterations.

One of the most interesting conclusions of these experiments is related to the Bratu problems with $\lambda > 0$. These problems are harder to solve than those with $\lambda < 0$. It is interesting to observe, however, that the discrete Newton method with local variations solved all of them, whereas the ordinary discrete Newton method failed. This seems to confirm that the method with local variations is less prone to convergence to undesirable local minimizers of $\|F(x)\|$.

Concluding, the discrete Newton method with local variations seems to be more robust than the ordinary discrete Newton algorithm. We believe that this fact is due to the different strategies that we used in our method: the local variations, that allow us to change the base points; the reduction of the step size of the discretization and the line search process, that produces the global convergence results.

Acknowledgement. We are indebted to an anonymous referee whose comments helped us to improve the first version of this paper.

References

[1] Banitchouk, N. V., Petrov, V. M. and Chernousko, R. L., *Numerical Solution of Problems with Variational Limits by the Method of Local Variations*, Žurnal Vyčisl. Mat. i Mat. Fiz., Vol. 6, pp 947-961, (1966).

[2] Coleman, T. F. and Moré, J. J., *Estimation of Sparse Jacobian Matrices and Graph Coloring Problems*, SIAM J. Numer. Anal., Vol. 20, pp 187-209, (1983).

[3] 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).

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

[5] Dolan, E. D. and Moré, J. J., *Benchmarking Optimization Software with Performance Profiles*, Math. Program. Series A91, pp 201-213, (2002)

[6] Friedlander, A., Gomes–Ruggiero, M. A., Kozakevich, D. N., Martínez, J. M. and Santos, S. A., *Solving Nonlinear Systems of Equations by Means of Quasi–Newton Methods with a Nonmonotone Strategy*, Optimization Methods and Software, Vol.8, pp 25-51, (1997).

[7] Goldfarb, D. and Toint, Ph. L., *Optimal Estimation of Jacobian and Hessian Matrices that Arise in Finite Difference Calculations*, Mathematics of Computation, Vol. 43, **167**, pp 69-88, (1984).

[8] Gomes–Ruggiero, M. A., Kozakevich, D. N. and Martínez, J. M., *Numerical Study on Large–Scale Nonlinear Solvers*, Computers and Mathematics with Applications, Vol.32, **3**, pp 1-13, (1996).

[9] Kelley, C. T., Iterative Methods for Linear and Nonlinear Equations, SIAM, (1995).

[10] Li, Dong-Hui and Fukushima, M., *Derivative-Free Line Search and Global Convergence of Broyden-Like Method for Nonlinear Equations*, Optimization Methods and Software **13**, pp 181–201, (2000).

[11] Moré, J. J., Garbow, B. S. and Hillstrom, K. E., *Testing Unconstrained Optimization Software*, ACM Transactions on Mathematical Software, Vol. 7, **1**, pp 17-41, (1981).

[12] Newsam, G. N. and Ramsdell, J.D., *Estimation on Sparse Jacobian Matrices*, SIAM J. Algebraic Discrete Methods, Vol. 4, pp 404-418, (1983).

[13] Pérez, R., Lopes, V. L. R., *Solving Recent Applications by Quasi-Newton Methods*, to appear in Applied Numerical Mathematics.

[14] Polak, E., *Computational Methods in Optimization: A Unified Approach*, Academic Press, New York, (1970).

[15] Polak, E., *A Globally Convergent Secant Method with Applications to Boundary Value Problems*, SIAM Journal of Numerical Analysis, Vol. 11 , **3**, pp 529-537, (1974).