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

Maria A. Diniz-Ehrhardt † Márcia A. Gomes-Ruggiero ‡ Véra 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 [11].

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 [6], and the local variations method [1, 12, 13]. The way in which we combine these ideas is what characterizes our algorithm. Roughly speaking, a typical iteration of the new algorithm starts from a current "base point" (the initial approximation or the previous iterate) and is defined by the following features:

(i) The groups used are CPR-valid groups in the sense of [3, 6];

*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

(ii) A new trial point is obtained each time a group is incorporated for updating the Jacobian matrix;

(iii) This trial point is used as "base point" for updating the next group of Jacobian columns if its residual norm is smaller than the residual norm at the previous base point of the discretization; otherwise, the base point remains the same;

(iv) The discretization process described in (i)-(iii) continues until the last group is incorporated;

(v) A Newtonian iteration with line search is performed when the discretization finishes.

Global convergence is obtained using classical backtracking in which we introduce a tolerant strategy of Li-Fukushima type [8].

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 [13] and Goldfarb and Toint [6], we apply our algorithm to solve some problems from the classical literature [9]. The numerical results show a good performance of this approach.

The paper is organized as follows. In Section 2 we describe the general algorithm and we present a convergence result. In Section 3 we give a more detailed description of the evaluation of the Jacobian matrix by groups as proposed by Goldfarb and Toint and of the local variations method. Section 4 presents our numerical experiments. Finally, in Section 5, we make some comments and present some conclusions about this work.

2 Model algorithm and convergence

Assume that $F : I\!R^n \to I\!R^n$, $F \in C^1(I\!R^n)$, and $\|\cdot\|$ an arbitrary norm. We denote $J(x)$, the Jacobian matrix of $F(x)$. Assume also 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 $\sigma \in (0, 1)$, $0 < \tau_{min} < \tau_{max} < 1$, $\alpha_{-1} = 1$, $c > 0, c' > 0$ and that $\{\eta_k\}$ is a sequence such that $\eta_k > 0$ for all $k = 0, 1, 2, \ldots$ and $\sum_{k=0}^{\infty} \eta_k = \eta < \infty$.

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

$$\|B_k - J(x_k)\| \leq c\ \alpha_{k-1}. \tag{4}$$

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

$$B_k d_k + F(x_k) = 0. \tag{5}$$

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

$$\|F(x_k + \alpha d_k)\| \leq (1 - \alpha\sigma)\|F(x_k)\| + \eta_k, \tag{6}$$

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

If (6) 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

$$\|x_{k+1} - x_k\| \leq c'\alpha_k\|d_k\| \tag{7}$$

and

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

Remark. Since $\eta_k > 0$ the condition (6) necessarily holds if α is small enough. Therefore, the iteration is well defined whenever the linear system (5) has a solution.

Theorem 1. *Assume that, for some sequence of indices $K_1 \subset I\!N$, we have that $J(x_k)$ 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 $k \in K \subset I\!N$, we have that $\alpha_k \geq \bar{\alpha} > 0$.

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

Let us analyze first Case (i). From (6) and (8)

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

for all $k = 0, 1, 2, \ldots$.

Adding all these inequalities, we get

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

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

Therefore,

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

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

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

Therefore,

$$\lim_{k\to\infty} \|F(x_k)\| = 0. \tag{9}$$

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

$$\lim_{k\to\infty} \alpha'_k = 0 \tag{10}$$

and

$$\|F(x_k + \alpha'_k d_k\| > (1 - \alpha'_k\sigma)\|F(x_k)\|. \tag{11}$$

Define

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

Then, by (11),

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

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 (5),

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

So, by (4)

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

Then, by (3),

$$\frac{L}{2}\,\alpha_k'\|d_k\|^2 + \alpha_{k-1}' c\,\|d_k\| > (1-\sigma)\|F(x_k)\|. \tag{12}$$

Now, for $k \in K_1$, $\|J(x_k)^{-1}\| \leq M$. But, by (4),

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

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 (10) and (12),

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

As in the deduction of (9), 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.

Theorem 2. *Assume that $\|J(x_k)^{-1}\| \leq M$ for all $k \in \mathbb{N}$ and that*

$$\|B_k - J(x_k)\| \leq c \min\{\alpha_0, \ldots, \alpha_{k-1}\} \tag{13}$$

for all $k \in \mathbb{N}$. Then, there exists $\bar{\alpha} > 0$ such that $\alpha_k \geq \bar{\alpha}$ for all $k \in \mathbb{N}$. Moreover, for all $r \in (1 - \sigma\bar{\alpha}, 1)$, there exists $k_0 \in \mathbb{N}$ such that

$$\|F(x_{k+1})\| \leq \|F(x_k)\| \quad \textit{for all} \quad k \geq k_0. \tag{14}$$

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 (13),

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

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

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

Then, by (5),

$$\|d_k\| \leq 2M\|F(x_k)\| \tag{15}$$

By Theorem 1, $\lim_{k\to\infty} \|F(x_k)\| = 0$. So, by (15),

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

Now, by (3) and (15),

$$\begin{aligned}
&\|F(x_k + d_k)\| \\
&= \|F(x_k + d_k) - F(x_k) - B_k d_k + (F(x_k) + B_k d_k) - (F(x_k) + J(x_k)d_k) + F(x_k) + J(x_k)d_k\| \\
&\leq \|F(x_k + d_k) - F(x_k) - J(x_k)d_k\| + \|(B_k - J(x_k))d_k\| \\
&\leq L\|d_k\|^2 + \|B_k - J(x_k)\|\,\|d_k\| \\
&\leq [4M^2 L\|F(x_k)\| + 2M\|B_k - J(x_k)\|]\,\|F(x_k)\|.
\end{aligned}$$

Since $\|F(x_k)\| \to 0$ and $\|B_k - J(x_k)\| \to 0$, the previous inequality implies that (6) 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. The second part of the prove follows straightforwardly from (6). □

Counterexample. It is interesting to show that α_k might not be bounded away from zero under the rule (4), even in situations in which the sequence converges to an isolated solution of (1). Define $F : \mathbb{R} \to \mathbb{R}$ 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, $\sigma = 1/2$. Clearly,

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

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

but

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

This shows that the rule (13) is necessary for proving the existence of a lower bound of α_k.

Theorems 1 and 2 did not use the assumption (7). This assumption, however, is necessary to prove local convergence of the algorithm, as we show in Theorem 3 below.

Theorem 3. *Assume the hypotheses of Theorem 1 and, 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 $\{\|B_k^{-1}\|\}_{k\in\mathbb{N}}$ is bounded. Then, x_k converges to x_*. If, in addition, (13) holds, there exists $\bar{\alpha} > 0$ such that $\alpha_k \geq \bar{\alpha}$ for all $k \in \mathbb{N}$ 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 $k_1 \in \mathbb{N}$ such that

$$\|J(x_*)(x_{k+1} - x_*)\| \leq r\|J(x_*)(x_k - x_*)\| \tag{16}$$

for all $k \geq k_1$.

Proof. By Theorem 1 we have that

$$\lim_{k\to\infty} \|F(x_k)\| = 0 \tag{17}$$

and

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

By the boundedness of $\{\|B_k^{-1}\|\}_{k\in I\!N}$, (5), (6), (7) and (17), 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 $0 < \|x - x_*\| \leq \varepsilon$. Let k_0 be such that

$$\|x_{k+1} - x_k\| \leq \frac{\varepsilon}{2} \tag{18}$$

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 (18) 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 (13) 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,

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

Then, by (3),

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

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

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

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

3 Discretization and local variations

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, $j = 1, 2, \ldots, n$, the jth column, of the approximation of $J(x)$ is given by:

$$(\hat{J})_j = \frac{F(x + he_j) - F(x)}{h}, \tag{19}$$

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 was to update groups of columns of $\hat{J}$ that could be evaluated together, using only one function evaluation for each group. The 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 the same group must have all its nonzero elements on different rows. Those groups are called CPR-valid.

The greedy choice of CPR groups suggested in [3] does not always give the least possible number of groups (see, for instance [10, 2, 6]). 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, it is not always that this strategy uses the minimum number of groups.

Newsan and Ramsdell [10] proved that it is always possible to estimate the sparse Jacobian matrix in a number of function evaluations which is equal to the maximum number of nonzero elements on a single row.

Goldfarb and Toint [6] also used the CPR idea to solve nonlinear sparse systems coming from the discretization of boundary value problems in partial differential equations, by finite diferences. With their strategy, they perform a number of function evaluations that is equal to the number of CPR-valid 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 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-valid groups can easily be identified [6].

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

$$|h_{k+1}| \leq \beta \alpha_k \tag{20}$$

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

Assume that $\{v_1, \ldots, v_q\} \subset I\!R^n$ 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

$$\|F(z)\| < \|F(y_j)\| \tag{21}$$

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

`Step 3.` 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 3, we redefine $x_0 \leftarrow y_{q+1}$.

Clearly, by (20) and (21) the conditions (7) and (8) are satisfied.

Now we explain how to compute the matrix B_{k+1} and we prove that, in this way, the condition (4) is satisfied so that the new algorithm is a particular case of Algorithm 1.

To each CPR direction v_j it is associated a set $I_j \subset \{1, \ldots, n\}$ such that

$$I_1 \cup \ldots \cup I_q = \{1, \ldots, n\},$$

$$I_j \cap I_\ell = \emptyset \quad \text{if} \quad i \neq j$$

and

$$[v_j]_i = 1 \ \text{ if } \ i \in I_j, \ \ 0 \quad \text{otherwise .}$$

The sets I_j are related to the sparsity of $J(x)$ in the following way: if $i_1, i_2 \in I_j$ and $i_1 \neq i_2$, then

$$\frac{\partial f_\ell}{\partial x_{i_1}}(x) \ \frac{\partial f_\ell}{\partial x_{i_2}}(x) = 0$$

for all $\ell = 1, \ldots, n$, $x \in I\!R^n$.

Let us write, to simplify the notation, $h = h_{k+1}$. The vector $[F(y_j + hv_j) - F(y_j)]/h$ contributes to the construction of the matrix B_{k+1} as follows: if $\nu \in I_j$ and the ith coordinate of $[F(y_j + hv_j) - F(y_j)]/h$ is nonnull, then the (i, ν) entry of B_{k+1} is set to be equal to this coordinate.

Then, for all $j = 1, \ldots, q$,

$$B_{k+1}v_j = [F(y_j + hv_j) - F(y_j)]/h. \tag{22}$$

Now, by (3),

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

Therefore,

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

So, by (22),

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

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

$$\|B_{k+1} - J(y_j)\| \leq c_1|h|. \tag{23}$$

But, by the definition of y_j we have that

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

therefore, by (2),

$$\|J(y_j) - J(x_{k+1})\| \leq c_2L|h|.$$

Then, by (23),

$$\|B_{k+1} - J(x_{k+1})\| \leq (c_2L + c_1)|h|.$$

So, by the choice (20), the condition (4) holds. Clearly, if we choose, instead of (20),

$$|h_{k+1}| \leq \beta \min\{\alpha_0, \ldots, \alpha_k\} \tag{24}$$

the assumption (13) 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 (20) and (24) 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,

$$|h_{k+1}| = \min\{\alpha_0, \ldots, \alpha_k\} s_k, \quad k = 0, 1, \ldots.$$

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

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 (6), then we compute the new step size as

$$\alpha_{new} = min\{\tau_{max}\alpha \ , \ max\{\tau_{min}\alpha \ , \ \frac{\alpha}{2}\}\}.$$

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 Algorithmic parameters

For the parameter σ used in the criterion (6), we took $\sigma = 10^{-4}$.
The values chosen for the limits of the interval for taking the new value of the parameter α were $\tau_{min} = 10^{-6}$ and $\tau_{max} = 1.0$.

4.5 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 direction vector v_j is defined as in
algorithm 2.

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

`Step 4:` Set: $x_{k+1} = x_k + s$ 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 [9] 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 were used. For example, using $\eta_k = 10^{39}$ at the three first iterations, the performance of `DNLV` method 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 numbers 2, 4, 7, 8, and 11) both algorithms had the same performance and in four problems (problems numbers 3, 6, 9 and 10) the two algorithms had a similar performance in terms of number of iterations: algorithm `DNLV` performed just one more iteration than algorithm `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. Finally, in Problems 1 and 5 the `DNLV` algorithm had its worst performance.

(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

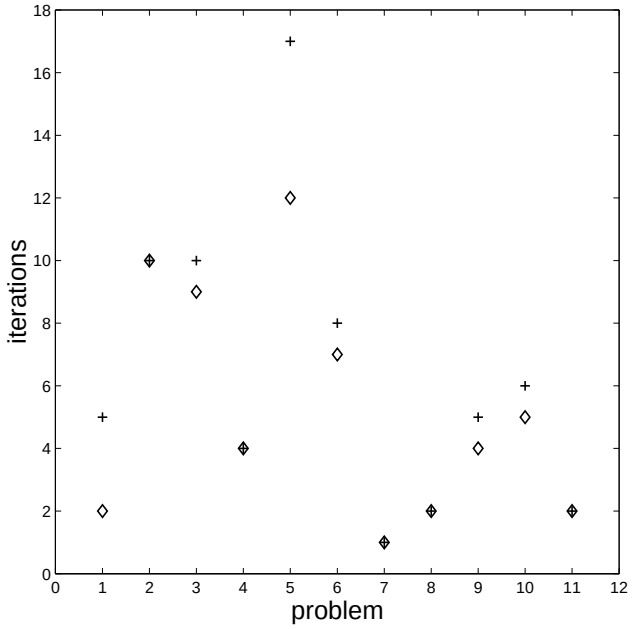

Figure 1: Comparison between algorithms DN and DNLV: number of iterations.

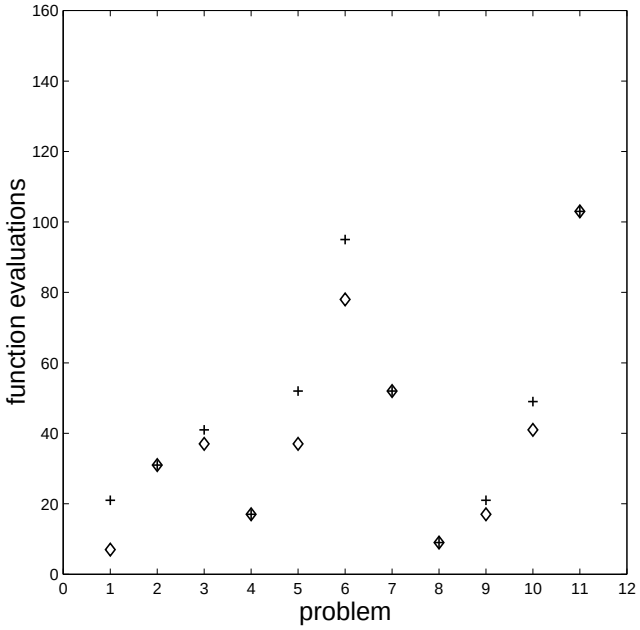

Figure 2: Comparison between algorithms DN and DNLV: number of function evaluations.

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

$$G_\lambda(u) \;=\; f(s,t), \tag{25}$$

with boundary conditions, where G is an operator that involves second-order partial derivatives of u. As in [7], we assumed the following known solution for the problem:

$$u_*(s,t) \;=\; 10st(1-s)(1-t)e^{s^{4.5}}. \tag{26}$$

In our tests 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 (25) the function and the derivatives by their approximations, and using the boundary conditions, we obtain a nonlinear system of equations like (1), whose dimension is 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 8 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 tool "performance profile" indroduced by Dolan and Moré, [5]. In a few words, 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 $\overline{t}$, such that, $\rho_s(\overline{t}) = 1$. 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 $m_{s,t}$. And 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 $\rho_s(\overline{t}) = 1$, if there exists such value for $t < r_M$. So, the winner in terms of robustness will be the solver $\hat{s}$ for which $\overline{t_{\hat{s}}} = \min\{\overline{t}_s, \forall s \in S\}$.

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 $\overline{t} \sim 14$. 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.

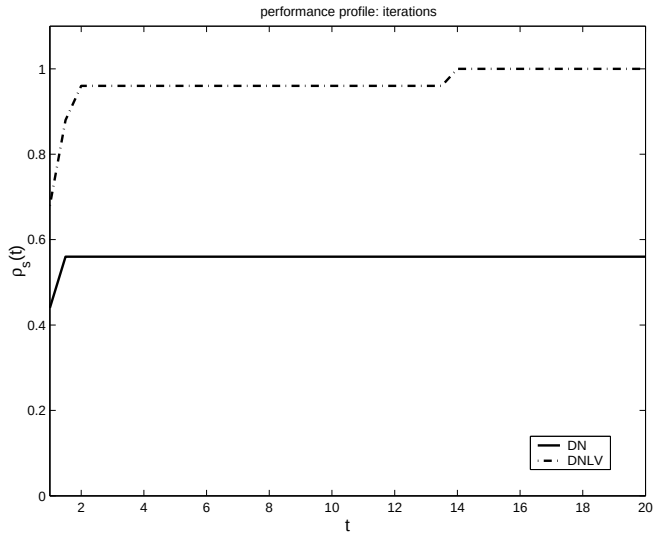

Figure 3: Performance profile with the number of iteratios as measure.

6 Conclusions

Analyzing the Tables 1–3, we can conclude that the new algorithm `DNLV` is competitive and it seems to be more robust than the discrete Newton's method implemented in `DN` algorithm. We believe that this fact is due to the different strategies that we used in our method: local variations, that allows 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.

Finally, for the boundary value problems tested, the performance of `DNLV` may be considered much better than that of `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 the problems were solved with the minimum number of iterations.

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*, Ž. 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] 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).

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

[8] 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).

[9] 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).

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

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

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

[13] 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).