Incomplete decomposition algorithms for discrete dynamic nonlinear models

N. Krejić * V. L. R. Lopes † J. M. Martínez ‡ R. Pérez §

May 28, 1998

Abstract

The problem of solving block triangular nonlinear systems of equations appears in several applications in industrial, social and environmental contexts. Those systems are given by

$$(F_1(x_1), F_2(x_1, x_2), \ldots, F_m(x_1, x_2, \ldots, x_m)) = 0.$$

The most usual way of solving such systems consists on solving sequentially the different $n_i \times n_i$ partial systems, $F_i(x_1, \cdots, x_i) = 0$. Some Newtonian approaches, where the system is considered as a whole and solved by Newton, quasi-Newton or inexact Newton methods have also been considered in the literature. In this paper it is proposed a "team model" scheme in which the system is also considered as a whole. Each iteration is constructed in m steps and for solving the partial problem $F_i(x_1, \cdots, x_i) = 0$ one is free to use any appropriate iterative method. A convergence theorem is proved and the algorithm is used to solve a dynamic model based on complementarity.

Keywords. Nonlinear systems, decomposition, convergence.

*Institute of Mathematics, University of Novi Sad, Trg Dositeja Obradovića 4, 21000 Novi Sad, Yugoslavia. This author was supported by FAPESP (Grant 96/8163-9). E-Mail: natasa@unsim.im.ns.ac.yu.

†Department of Applied Mathematics, IMECC-UNICAMP, University of Campinas, CP 6065, 13083-970 Campinas SP, Brazil. This author was supported by FAPESP (Grant 90-3724-6), CNPq and FAEP-UNICAMP. E-Mail: vlopes@ime.unicamp.br

‡Department of Applied Mathematics, IMECC-UNICAMP, University of Campinas, CP 6065, 13083-970 Campinas SP, Brazil. This author was supported by FAPESP (Grant 90-3724-6), CNPq and FAEP-UNICAMP. E-Mail: martinez@ime.unicamp.br

§Department of Applied Mathematics, IMECC-UNICAMP, University of Campinas, CP 6065, 13083-970 Campinas SP, Brazil. This author was supported by FAPESP (Grant 96-0607-5). E-Mail: rosana@ime.unicamp.br

1 Introduction

Many mathematical models used in industrial, social and environmental applications involve the solution of optimization, complementarity or variational inequality problems. When these models are considered at different periods of time it is usual that the solution of each of them provides boundary conditions for the following. Frequently, the original model can be expressed as a nonlinear system of equations, so that the complete dynamical model is

$$\begin{aligned} &F_1(x_1) = 0, \\ &F_2(x_1, x_2) = 0, \\ &\vdots \\ &F_m(x_1, x_2, \ldots, x_m) = 0, \end{aligned} \tag{1}$$

where $x = (x_1, x_2 \ldots, x_m)$, $x_i \in I\!R^{n_i}$ and $F_i : I\!R^{n_1} \times I\!R^{n_2} \times \ldots \times I\!R^{n_i} \to I\!R^{n_i}$, $i = 1, 2, \ldots, m$, $n_1 + n_2 + \ldots + n_m = n$.

The most usual way of solving (1) is the "straight decomposition scheme" (SDS) that consists on "solving" sequentially the different $n_i \times n_i$ systems. Alternative Newtonian procedures were suggested in [2, 3, 4]. In fact, "solving" is an ambiguous word in the case of a nonlinear system where to find exact solutions is not possible.

In order to understand the necessity of the analysis presented in this paper, let us consider the most simple situation, where we have two blocks of equations ($m = 2$). Suppose that one is going to "use SDS" and that, in practice, to obtain $\|F_1(x_1)\| \leq \varepsilon$, $\|F_2(x_1, x_2)\| \leq \varepsilon$ is acceptable. (The tolerance ε is not necessarily very small: when the system is originated on the constraints of a nonlinear programming problem the goal of an iteration can be to solve the system in quite a loose way. See, for example, [6].) The naive way to proceed consists on using some Newtonian algorithm applied to $F_1(x) = 0$ (starting with an arbitrary $x_1^0 \in I\!R^{n_1}$), repeating the iteration until $\|F_1(\tilde{x}_1)\| \leq \varepsilon$ is achieved. The second (and final) step would be to fix $x_1 = \tilde{x}_1$ and, again, to use some Newton-like method for the $n_2 \times n_2$ system $G(x_2) \equiv F_2(\tilde{x}_1, x_2) = 0$, stopping when $\|F_2(\tilde{x}_1, \tilde{x}_2)\| \leq \varepsilon$. However, it is possible that such $\tilde{x}_2$ does not exist at all. More frequently, in the course of the second calculation, we may realise that the computational effort that is necessary to find $\tilde{x}_2$ is not affordable. Certainly, in this case we should come back to the first equation and we should solve it with a tighter precision $\varepsilon' < \varepsilon$, expecting that to have a more accurate approximation $\tilde{x}_1$ will help us to find a reasonable $\tilde{x}_2$ in the second step.

The simple procedure described above suggests an iterative process,

which is essentially the one considered in this paper. The possibility of taking advantage of parallel computer architectures, led to the development of related methods in [2, 4]. The idea of these algorithms is to take the system (1) as a whole, though taking into account its block-angular structure, in such a way that each iteration can be considered an approximation to a Newton, quasi-Newton [2] or inexact-Newton iteration [4]. See also [1, 7, 9].

In this paper we preserve the philosophy of [2, 4] in the sense that at each iteration the system is considered as a single unity, but we do not use the Newtonian approach, being free to use any method to improve the approximation at each partial iteration. The method works as if the i-th problem were a real-life problem for which there exists a specific team dedicated to its solution, probably with different tools. In this way, each team takes advantage of the progress of the teams that are its predecesors, minimizing the inactive time. Of course, there exists a computer model of this situation, based on parallel processors, but we think that the "team-model" reflects reasonably many industrial situations.

2 Main results

The iterates of the Block-Inexact method introduced in this paper will be denoted $x^k, k = 0, 1, 2, \ldots$ According to the structure of the system, each iteration will consist of m steps. Given the current point at the $i-$th step, only its $i-$th component will be modified so that the new $i-$th component of the system will be a fraction of the $i-$th component of F computed at the current point. Each iterate $x^k \in I\!R^n$ can be partitioned into m components $x_i^k \in I\!R^{n_i}, i = 1, \ldots, m$. So, we will write

$$x^k = (x_1^k, \ldots, x_m^k) \in I\!R^{n_1} \times \ldots \times I\!R^{n_m} = I\!R^n.$$

We will use appropriate norms on the spaces $I\!R^{n_1} \times \ldots \times I\!R^{n_i}$. Given an arbitrary norm $|\cdot|$ on $I\!R^{n_i}$ we define

$$\|(x_1, \ldots, x_i)\| = \sum_{j=1}^{i} |x_j|.$$

for all $(x_1, \ldots, x_i) \in I\!R^{n_1} \times \ldots \times I\!R^{n_i}$.

The description of the main iteration of the algorithm is given below.

Algorithm 2.1

Given $x^k \in I\!R^n$ and $0 \leq t_k \leq t < 1$, $x^{k+1} = (x_1^{k+1}, \ldots, x_m^{k+1})$ is obtained in the following way:

For $i = 1, \ldots, m$, by means of an appropriate method, find $x_i^{k+1} \in I\!R^{n_i}$ such that

$$|F_i(x_1^{k+1}, \ldots, x_{i-1}^{k+1},\ x_i^{k+1})| \leq t_k |F_i(x_1^{k+1}, \ldots, x_{i-1}^{k+1},\ x_i^k)|. \tag{2}$$

The main assumption that implies convergence of the algorithm says that iterates can be always chosen in a set where the solution is "strongly unique" (see (3) below). In addition, a Lipschitz condition (see (4)) will be needed to bound the increase of $|F_i|$ between points where this component is not considered. No differentiability assumptions will be made. If the system has a strongly unique solution in the whole $I\!R^n$ and the Lipschitz condition takes place, Theorem 2.1 represents a global convergence result for Algorithm 2.1.

Assumption A1

There exist open sets $N_1 \subset I\!R^{n_1}, \ldots, N_m \subset I\!R^{n_m}$ such that

1. All the iterates x_i^k generated by Algorithm 2.1 belong to N_i.
2. There exist $x_1^* \in N_1, \ldots, x_m^* \in N_m$, $\beta > 0$, such that

$$\|(x_1, \ldots, x_i) - (x_1^*, \ldots, x_i^*)\| \leq \beta \|(F_1(x_1), \ldots, F_i(x_1, \ldots, x_i))\| \tag{3}$$

   for all $x_i \in I\!R^{n_i}$, $i = 1, \ldots, m$.

3. There exists $L > 0$ such that

$$|F_i(x_1, \ldots, x_i) - F_i(y_1, \ldots, y_i)| \leq L \|(x_1, \ldots, x_i) - (y_1, \ldots, y_i)\| \tag{4}$$

   for all $x_i, y_i \in I\!R^{n_i}$, $i = 1, \ldots, m$.

The following technical lemma will help us to prove the main convergence result of this section.

Lemma 2.1 *Let e_i^k, $i = 1, 2, \ldots, m$, $k = 0, 1, 2, \ldots$, $\rho \in [0, 1)$ and $C > 0$ be real numbers such that $e_i^k \geq 0$ for all $i = 1, 2, \ldots, m$, $k = 0, 1, 2, \ldots$ and*

$$e_i^{k+1} \leq \rho e_i^k + C \sum_{j=1}^{i-1} (e_j^{k+1} + e_j^k) \tag{5}$$

for all $i = 1, \ldots, m$, $k = 0, 1, 2, \ldots$ Then, the sequence $\left\{e^k\right\}$, with $e^k = \left(e_1^k, \ldots, e_m^k\right)$, converges to 0.

Proof. Let $B = (b_{ij})$ and $D = (d_{ij})$ be lower-triangular matrices such that for all $j = 1, \ldots, n$, $i = 1, \ldots, n$, $i > j$,

1. $b_{jj} = 1$ and $b_{ij} = -C$.
2. $d_{jj} = 1$ and $d_{ij} = C$.

Then, by (5),

$$Be^{k+1} \leq De^k.$$

Since all the entries of B^{-1} are nonnegative, this implies that

$$e^{k+1} \leq B^{-1}De^k.$$

Moreover, since the entries of $B^{-1}D$ are also nonnegative, we have that

$$e^k \leq [B^{-1}D]^k e^0 \tag{6}$$

for all $k = 0, 1, 2, \ldots$. But the matrix $B^{-1}D$ is lower triangular and all its diagonal elements are equal to ρ. Hence, its spectral radius is less than one, which implies that $lim_{k\to\infty}[B^{-1}D]^k = 0$. Therefore, since $e^k \geq 0$ for all k, (6) implies that $\lim_{k\to\infty} e^k = 0$, as we wanted to prove. ∎

Theorem 2.1 *Let F satisfy Assumption A1, and let $\{x^k\}$ be a sequence generated by (2). Then $\{x^k\}$ converges to $x^* = (x_1^*, \ldots, x_m^*)$ and $F(x^*) = 0$.*

Proof. Let us define

$$e_i^k = |F_i(x_1^k, x_2^k, \ldots, x_i^k)|,\ i = 1, \ldots, m,\ k = 0, 1, 2 \ldots.$$

Taking $i = 1$, from the definition of the iterative method we have that

$$e_1^{k+1} \leq te_1^k,$$

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

Now, for $i > 1$, by Assumption A1, we have:

$$e_i^{k+1} = |F_i(x_1^{k+1}, \ldots, x_{i-1}^{k+1},\ x_i^{k+1})| \leq t|F_i(x_1^{k+1}, \ldots, x_{i-1}^{k+1},\ x_i^k)|$$

$$\leq t|F_i(x_1^k, \ldots, x_{i-1}^k,\ x_i^k)| + t|F_i(x_1^{k+1}, \ldots, x_{i-1}^{k+1},\ x_i^k) - F_i(x_1^k, \ldots, x_{i-1}^k,\ x_i^k)|$$

$$\leq t|F_i(x_1^k, \ldots, x_{i-1}^k,\ x_i^k)| + tL\|(x_1^{k+1}, \ldots, x_{i-1}^{k+1}) - (x_1^k, \ldots, x_{i-1}^k)\|$$

$$\leq t|F_i(x_1^k, \ldots, x_{i-1}^k,\ x_i^k)|$$

$$+tL(\|(x_1^{k+1},\ldots,x_{i-1}^{k+1})-(x_1^*,\ldots,x_{i-1}^*)\|+\|(x_1^k,\ldots,x_{i-1}^k)-(x_1^*,\ldots,x_{i-1}^*)\|)$$

$$\leq t|F_i(x_1^k,\ldots,x_{i-1}^k,\ x_i^k)|-tL\beta\sum_{j=1}^{i-1}(|F_j(x_1^{k+1},\ldots,x_j^{k+1})|+|F_j(x_1^k,\ldots,x_j^k)|).$$

Therefore, by Lemma 1,

$$\lim_{k\to\infty}|F_i(x_1^k,\ldots,x_{i-1}^k,\ x_i^k)|=0.$$

Then, the second part of Assumption A1 implies the desired result. ∎

3 A practical application

We have applied Algorithm 2.1 to the problem described in this section.

Two different agencies allocate resources in five different departments of a university. Agency 1 has as objective the promotion of scientific research and Agency 2 aims the development of human resources. Let us call $x_1,\ldots,x_5$ and $y_1,\ldots,y_5$ the resources allocated by agencies 1 and 2, respectively, in the departments $1,2,\ldots,5$. For a given value of the investment x_i, the research production p_i of department i is an increasing concave function of x_i (as an application of the law of decreasing returns). However, this function also depends on y_i. In fact, when the investment y_i increases, the production p_i decreases. So,

$$p_i(x_i,y_i)=\beta_i(y_i)\alpha_i(x_i), \tag{7}$$

where α_i is increasing, concave and $\alpha_i(0)>0$, while β_i is decreasing, $\beta_i(0)=1$ and $\lim_{y_i\to\infty}\beta_i(y_i)=\theta_i\in(0,1)$. In other words, the function α_i is the production function without investment of Agency 2, and $\theta_i\alpha_i(x_i)$ is the minimum possible production when an infinite investment of Agency 2 is allocated. In our case, the formulae for $p_1,\ldots,p_5$ are given below:

$$p_1(x,y)=\frac{y+1.8}{1.9y+1.8}3.2(1.1x+0.97)^{0.61},\quad p_2(x,y)=\frac{y+2.1}{1.3y+2.1}4.1(1.3x+0.81)^{0.88},$$

$$p_3(x,y)=\frac{y+1.5}{3.1y+1.5}2.2(0.86x+0.53)^{0.42},\quad p_4(x,y)=\frac{y+1.9}{3.3y+1.9}4.2(1.5x+0.74)^{0.78},$$

$$p_5(x,y)=\frac{y+2.3}{1.1y+2.3}1.5(0.44x+0.51)^{0.29}$$

Analogously, calling $q_i(x_i,y_i)$ the function that gives the human resources production of department i, we have:

$$q_1(y,x)=\frac{x+1.2}{1.5x+1.2}2.3(0.74y+1.25)^{0.47},\quad q_2(y,x)=\frac{x+2.5}{2.4x+2.5}1.9(1.6y+0.44)^{0.93},$$

$$q_3(y,x)=\frac{x+1.8}{2.9x+1.8}3.6(1.4y+0.79)^{0.73},\quad q_4(y,x)=\frac{x+3.1}{1.8x+3.1}4.2(2.2y+0.69)^{0.45},$$

$$q_5(y,x) = \frac{x+2.1}{2.7x+2.1} 2.1(0.97y+2.4)^{0.25}.$$

Agency 1 has a limited amount of resources r_1 that are going to be distributed among the departments in order to maximize the research production. Therefore, for given investments $y_1, \ldots, y_5$ of Agency 2, the allocation of resources x_i of Agency 1 will be given in order to

$$\text{Maximize}_x \quad f(x,y) \equiv \sum_{i=1}^{5} p_i(x_i, y_i) \tag{8}$$

$$\text{subject to} \quad \sum_{i=1}^{5} x_i = r_1, \quad x_i \geq 0, \quad i = 1, \ldots, 5. \tag{9}$$

The optimality conditions of problem (8)-(9) are:

$$-\nabla_x f(x,y) + \lambda \underline{\mathbf{1}} \geq 0, \quad x \geq 0, \tag{10}$$

$$\langle -\nabla_x f(x,y) + \lambda \underline{\mathbf{1}}, \quad x \rangle = 0, \tag{11}$$

$$\sum_{i=1}^{5} x_i = r_1, \tag{12}$$

where $\underline{\mathbf{1}} = (1, \ldots, 1)$ and λ is the Lagrange multiplier associated to the constraint $\sum_{i=1}^{5} x_i = r_1$.

In a similar way, we see that the optimality conditions of the optimization problem corrresponding to (8)-(9) with the objective function $g(x,y) = \sum_{i=1}^{5} q_i(x_i, y_i)$ are:

$$-\nabla_y g(x,y) + \mu \underline{\mathbf{1}} \geq 0, \quad y \geq 0, \tag{13}$$

$$\langle -\nabla_y g(x,y) + \mu \underline{\mathbf{1}}, \quad y \rangle = 0, \tag{14}$$

$$\sum_{i=1}^{5} y_i = r_2. \tag{15}$$

The conditions (10)-(15) are the equilibrium conditions that determine the investments of both agencies. The set of all these conditions do not represent optimality conditions of any optimization problem, but a complementarity problem that can be written in many forms as a nonlinear system of equations. One of them is:

$$\min \{[-\nabla_x f(x,y) + \lambda \underline{\mathbf{1}}]_i, \quad x_i\} = 0, \quad i = 1, \ldots, 5; \tag{16}$$

$$\min \{[-\nabla_y g(x,y) + \mu \underline{\mathbf{1}}]_i, \quad y_i\} = 0, \quad i = 1, \ldots, 5; \tag{17}$$

$$\sum_{i=1}^{5} x_i = r_1, \quad \sum_{i=1}^{5} y_i = r_2. \tag{18}$$

The equilibrium conditions (16)-(18) form a 12×12 nonsmooth system of equations that can be solved, for example, using the method introduced in [5, 8]. Its solution represents the investments of the agencies under the limitation of resources r_1, r_2 in a definite period of time t (say, one year). We want to study the behavior of this system as time progreeds, when it is subject to changes in r_1 or r_2. However, we take into account that the real system is resistant to large changes in investments, so that the objective functions $f(x, y)$ and $g(x, y)$ are going to be replaced by $f(x, y) - \frac{\rho}{2}\|x - \bar{x}\|^2$ and $g(x, y) - \frac{\rho}{2}\|y - \bar{y}\|^2$, where $\bar{x}$ and $\bar{y}$ represent the investments at the previous period of time and $\rho > 0$ represents the resistance of the system to abrupt changes. So, the equilibrium conditions at period $t + 1$ turn out to be:

$$\min \{[-\nabla_x f(x, y) + \rho(x - \bar{x}) + \lambda \underline{\mathbf{1}}]_i, x_i\} = 0, \quad i = 1, \ldots, 5; \tag{19}$$

$$\min \{[-\nabla_y g(x, y) + \rho(y - \bar{y}) + \mu \underline{\mathbf{1}}]_i, y_i\} = 0, \quad i = 1, \ldots, 5; \tag{20}$$

$$\sum_{i=1}^{5} x_i = r_1, \quad \sum_{i=1}^{5} y_i = r_2. \tag{21}$$

Starting with $r_1 = 25$ and $r_2 = 11$, we want to simulate the behavior of the system during 10 periods of time, under the following boundary conditions:

(1) r_1 doubles at $t = 2$ and r_2 remains constant;

(2) r_2 doubles at $t = 2$ and r_1 remains constant;

(3) both r_1 and r_2 double at $t = 2$.

For each situation, we want to analyze different degrees of resistance of the system, taking $\rho = 0, 1$ and 10. Obviously, in the case $\rho = 0$ the ten problems are completely independent and the solution for $t = 2, \ldots, 10$ are the same.

An external iteration was finished when we achieved

$$\|F(x^k)\| \leq 10^{-5}. \tag{22}$$

We stopped the internal (inexact) iterations when

$$|F_i(x_1^{k+1}, \ldots, x_{i-1}^{k+1}, \ x_i^{k+1})| \leq 0.1 |F_i(x_1^{k+1}, \ldots, x_{i-1}^{k+1}, \ x_i^{k})|. \tag{23}$$

For the first period of time we run the algorithm of [5] with several different initial vectors and the same stopping criteria as above. The results are given in the following tables.

Case (1): r_1 doubles and r_2 does not change.

Time	*x(1)*	*x(2)*	*x(3)*	*x(4)*	*x(5)*	*y(1)*	*y(2)*	*y(3)*	*y(4)*	*y(5)*
1	0.000	25.000	0.000	0.000	0.000	0.000	0.000	9.258	1.742	0.000
2-10	0.000	50.000	0.000	0.000	0.000	0.000	0.000	9.258	1.742	0.000

Table 4.1: Solution for $\rho = 0$.

Time	*x(1)*	*x(2)*	*x(3)*	*x(4)*	*x(5)*	*y(1)*	*y(2)*	*y(3)*	*y(4)*	*y(5)*
1	0.000	25.000	0.000	0.000	0.000	0.000	0.000	9.258	1.742	0.000
2	4.916	31.797	4.001	5.305	3.981	0.000	0.183	8.975	1.842	0.000
3	4.848	33.537	3.032	5.592	2.990	0.000	0.341	8.743	1.916	0.000
4	4.787	35.229	2.095	5.860	2.028	0.000	0.468	8.573	1.959	0.000
5	4.722	36.879	1.193	6.108	1.099	0.000	0.543	8.502	1.955	0.000
6	4.624	38.487	0.350	6.328	0.212	0.000	0.494	8.658	1.849	0.000
7	4.132	39.703	0.000	6.165	0.000	0.000	0.299	9.059	1.643	0.000
8	3.457	40.726	0.000	5.817	0.000	0.000	0.110	9.413	1.477	0.000
9	2.799	41.723	0.000	5.477	0.000	0.000	0.000	9.683	1.317	0.000
10	2.179	42.658	0.000	5.163	0.000	0.000	0.000	9.847	1.153	0.000

Table 4.2: Solution for $\rho = 1$.

Time	*x(1)*	*x(2)*	*x(3)*	*x(4)*	*x(5)*	*y(1)*	*y(2)*	*y(3)*	*y(4)*	*y(5)*
1	0.000	25.000	0.000	0.000	0.000	0.000	0.000	9.258	1.742	0.000
2	4.989	30.186	4.897	5.032	4.895	0.000	0.021	9.225	1.754	0.000
3	4.979	30.371	4.795	5.064	4.791	0.000	0.042	9.192	1.766	0.000
4	4.969	30.555	4.692	5.096	4.687	0.000	0.062	9.161	1.777	0.000
5	4.959	30.739	4.591	5.128	4.583	0.000	0.082	9.129	1.789	0.000
6	4.950	30.922	4.489	5.159	4.479	0.000	0.101	9.099	1.800	0.000
7	4.941	31.104	4.388	5.191	4.376	0.000	0.121	9.068	1.811	0.000
8	4.932	31.285	4.287	5.222	4.274	0.000	0.140	9.039	1.821	0.000
9	4.923	31.466	4.187	5.253	4.171	0.000	0.159	9.010	1.832	0.000
10	4.915	31.642	4.087	5.283	4.069	0.000	0.177	8.981	1.842	0.000

Table 4.3: Solution for $\rho = 10$.

Case (2): r_1 does not change and r_2 doubles.

Time	*x(1)*	*x(2)*	*x(3)*	*x(4)*	*x(5)*	*y(1)*	*y(2)*	*y(3)*	*y(4)*	*y(5)*
1	0.000	25.000	0.000	0.000	0.000	0.000	0.000	9.258	1.742	0.000
2-10	0.000	50.000	0.000	0.000	0.000	0.000	0.000	19.401	2.599	0.000

Table 4.4: Solution for $\rho = 0$.

Time	*x(1)*	*x(2)*	*x(3)*	*x(4)*	*x(5)*	*y(1)*	*y(2)*	*y(3)*	*y(4)*	*y(5)*
1	0.000	25.000	0.000	0.000	0.000	0.000	0.000	9.258	1.742	0.000
2	0.000	25.000	0.000	0.000	0.000	1.754	2.391	12.214	4.187	1.454
3	0.000	25.000	0.000	0.000	0.000	1.342	2.576	12.943	4.402	0.737
4	0.000	25.000	0.000	0.000	0.000	0.962	2.750	13.645	4.587	0.057
5	0.000	25.000	0.000	0.000	0.000	0.478	2.757	14.163	4.602	0.000
6	0.000	25.000	0.000	0.000	0.000	0.042	2.733	14.636	4.589	0.000
7	0.000	25.000	0.000	0.000	0.000	0.000	2.578	14.965	4.456	0.000
8	0.000	25.000	0.000	0.000	0.000	0.000	2.410	15.267	4.323	0.000
9	0.000	25.000	0.000	0.000	0.000	0.000	2.241	15.556	4.203	0.000
10	0.000	25.000	0.000	0.000	0.000	0.000	2.074	15.833	4.093	0.000

Table 4.5: Solution for $\rho = 1$.

Time	*x(1)*	*x(2)*	*x(3)*	*x(4)*	*x(5)*	*y(1)*	*y(2)*	*y(3)*	*y(4)*	*y(5)*
1	0.000	25.000	0.000	0.000	0.000	0.000	0.000	9.258	1.742	0.000
2	0.000	25.000	0.000	0.000	0.000	2.152	2.219	11.536	3.970	2.123
3	0.000	25.000	0.000	0.000	0.000	2.105	2.239	11.613	3.997	2.046
4	0.000	25.000	0.000	0.000	0.000	2.058	2.258	11.690	4.024	1.970
5	0.000	25.000	0.000	0.000	0.000	2.012	2.277	11.767	4.050	1.894
6	0.000	25.000	0.000	0.000	0.000	1.966	2.296	11.844	4.076	1.818
7	0.000	25.000	0.000	0.000	0.000	1.920	2.316	11.921	4.102	1.742
8	0.000	25.000	0.000	0.000	0.000	1.874	2.335	11.997	4.127	1.667
9	0.000	25.000	0.000	0.000	0.000	1.829	2.354	12.073	4.152	1.592
10	0.000	25.000	0.000	0.000	0.000	1.784	2.373	12.149	4.177	1.517

Table 4.6: Solution for $\rho = 10$.

Case (3): r_1 and r_2 double.

Time	*x(1)*	*x(2)*	*x(3)*	*x(4)*	*x(5)*	*y(1)*	*y(2)*	*y(3)*	*y(4)*	*y(5)*
1	0.000	25.000	0.000	0.000	0.000	0.000	0.000	9.258	1.742	0.000
2-10	0.000	50.000	0.000	0.000	0.000	0.000	0.000	19.401	2.599	0.000

Table 4.7: Solution for $\rho = 0$.

Time	*x(1)*	*x(2)*	*x(3)*	*x(4)*	*x(5)*	*y(1)*	*y(2)*	*y(3)*	*y(4)*	*y(5)*
1	0.000	25.000	0.000	0.000	0.000	0.000	0.000	9.258	1.742	0.000
2	4.779	31.625	4.184	5.250	4.162	1.929	2.693	11.583	4.122	1.674
3	4.586	33.211	3.382	5.482	3.338	1.668	3.168	11.730	4.279	1.156
4	4.419	34.760	2.595	5.696	2.530	1.413	3.623	11.907	4.410	0.647
5	4.276	36.272	1.825	5.888	1.740	1.159	4.050	12.130	4.513	0.148
6	4.164	37.740	1.076	6.052	0.969	0.811	4.348	12.350	4.492	0.000
7	4.099	39.146	0.362	6.171	0.223	0.386	4.522	12.733	4.360	0.000
8	3.854	40.189	0.000	5.957	0.000	0.000	4.545	13.349	4.106	0.000
9	3.430	41.009	0.000	5.560	0.000	0.000	4.431	13.813	3.756	0.000
10	3.025	41.800	0.000	5.175	0.000	0.000	4.307	14.252	3.441	0.000

Table 4.8: Solution for $\rho = 1$.

Time	*x(1)*	*x(2)*	*x(3)*	*x(4)*	*x(5)*	*y(1)*	*y(2)*	*y(3)*	*y(4)*	*y(5)*
1	0.000	25.000	0.000	0.000	0.000	0.000	0.000	9.258	1.742	0.000
2	4.975	30.166	4.917	5.027	4.915	2.172	2.251	11.469	3.962	2.147
3	4.951	30.332	4.834	5.053	4.830	2.144	2.302	11.480	3.982	2.093
4	4.927	30.498	4.751	5.079	4.745	2.116	2.352	11.491	4.002	2.040
5	4.903	30.663	4.669	5.105	4.660	2.088	2.402	11.502	4.021	1.987
6	4.880	30.828	4.586	5.131	4.576	2.060	2.452	11.514	4.040	1.934
7	4.856	30.992	4.504	5.156	4.491	2.032	2.502	11.525	4.059	1.881
8	4.834	31.156	4.422	5.182	4.407	2.005	2.552	11.537	4.078	1.828
9	4.811	31.319	4.340	5.207	4.323	1.978	2.602	11.549	4.096	1.775
10	4.789	31.482	4.258	5.232	4.239	1.950	2.651	11.562	4.114	1.722

Table 4.9: Solution for $\rho = 10$.

These experiments were run using a modest computer environment (PC 486 with 66 MHz) and the code was written in Fortran. The results were completely satisfactory from the point of view of convergence, considering computer time and human cost. Moreover, the results of the simulations were consistent with independent qualitative analyses. Therefore, the algorithm introduced in this paper helped to validate the model considered in this section.

References

[1] Dembo, R.S., Eisenstat, S.C. and Steihaug, T., *Inexact Newton methods*. SIAM Journal on Numerical Analysis, 19 (1982) 400-408.

[2] Dennis, J.E., Martínez, J.M. and Zhang, X., *Triangular decomposition methods for solving reducible nonlinear systems of equations*. SIAM Journal on Optimization, 4 (1994) 358-382.

[3] Erickson, J., *A note on the decomposition of systems of sparse nonlinear equations*. BIT, 16 (1976) 462-465.

[4] Krejić, N. and Martínez, J.M., *A globally convergent Inexact - Newton method for solving reducible nonlinear systems of equations*. Technical Report 54/97, IMECC-UNICAMP, Campinas, Brazil, 1997.

[5] Lopes, V.L.R., Martínez, J.M. and Pérez, R., *On the local convergence of quasi-Newton methods for nonlinear complementarity problems.*To appear in Journal of Computational and Applied Mathematics.

[6] Martínez, J. M, *A Two-Phase Model Algorithm with global convergence for Nonlinear Programming*, to appear in Journal of Optimization Theory and Applications 96 (1998).

[7] Martínez, J.M. and Qi, L., *Inexact Newton methods for solving nonsmooth equations*. Journal of Computational and Applied Mathematics, 60 (1995) 127-145.

[8] Pérez, R., *Algoritmos para complementaridade não linear e problemas relacionados*. PhD. Thesis, IMECC-UNICAMP, Campinas, Brazil, 1997.

[9] Ypma, T.J., *Local convergence of inexact Newton methods*. SIAM Journal on Numerical Analysis, 21 (1984) 583-590.