Algorithms **2015**, *8*, 774-785; doi:10.3390/a8030774

OPEN ACCESS

algorithms

ISSN 1999-4893

www.mdpi.com/journal/algorithms

Article

Parallel Variants of Broyden's Method

Ioan Bistran, Stefan Maruster * and Liviu Octavian Mafteiu-Scai

West University of Timisoara, B-dul V. Parvan No.4, Timisoara 300223, Romania;
E-Mails: ioan.bistran@info.uvt.ro (I.B.); lscai@info.uvt.ro (L.O.M.-S.)

* Author to whom correspondence should be addressed; E-mail: maruster@info.uvt.ro;
Tel.: +40-748-585-220.

Academic Editors: Alicia Cordero, Juan R. Torregrosa and Francisco I. Chicharro

Received: 23 June 2015 / Accepted: 1 September 2015 / Published: 15 September 2015

Abstract: In this paper we investigate some parallel variants of Broyden's method and, for the basic variant, we present its convergence properties. The main result is that the behavior of the considered parallel Broyden's variants is comparable with the classical parallel Newton method, and significantly better than the parallel Cimmino method, both for linear and nonlinear cases. The considered variants are also compared with two more recently proposed parallel Broyden's method. Some numerical experiments are presented to illustrate the advantages and limits of the proposed algorithms.

Keywords: systems of equations; Broyden's method; parallel algorithms

---

1. Introduction

Let $F : \mathcal{R}^m \rightarrow \mathcal{R}^m$ be a (nonlinear) mapping and let $F(x) = 0$ be the corresponding system of equations. In [1], Xu proposed the following partially asynchronous block quasi-Newton method for such a nonlinear system. Let $\mathcal{P}$ be a partition of F and x in p partitions, $F = (F_1, ..., F_p)$, $x = (x_1, ..., x_p)$, where F_i and x_i are block functions and block variables, respectively, of the same dimensions m_i. The variable x is the common memory of the multiprocessor system, every processor i can read the content of x and can write its new update of x_i. Let B_i, $i = 1, ..., p$, be square matrices of dimension m_i. The main steps of the process i are :

Algorithm 1.

Input: $x,\ B_i$

Output: x

1. Compute $x_i^+ := x_i - B_i^{-1}F_i(x)$;
2. Get x^+ from x by replacing x_i with x_i^+;
3. Compute $s = x_i^+ - x_i$ and $y := F_i(x^+) - F_i(x)$;
4. Update $B_i,\ B_i := B_i + \frac{(y - B_i s)s^T}{s^T s}$;
5. $x_i := x_i^+,\ x := x^+$;

Applying the Sherman–Morrison lemma (Step 4, Algorithm 1) to inverse the matrix B_i, the polynomial complexity of order three is reduced to polynomial complexity of order two.

More recently, in [2,3] Jiang and Yang proposed several preconditioners for the block Broyden's method and discussed the implementation process. The main steps of parallel variant of Broyden's method considered in [2,3] (Algorithm 2 below) are (B_0 is a block diagonal matrix, B_k^i is the i^{th} diagonal block of B_k):

Algorithm 2.

Input: $x_0,\ B_0,\ kmax$

Output: x_k

1. **For** $k = 0, 1, ..$ **Until** $convergence$ **Or** $k > kmax$ **Do**
2. $x_{k+1} = x_k - B_k^{-1}F(x_k)$;
3. $s = x_{k+1} - x_k$;
4. **For** $i = 1, ..., p$
5. $B_k^i = B_k^i + \frac{F_i(x_{k+1})s_i^T}{s^T s}$.
6. **End**
7. **End**

In the case of a linear system, the sequential Broyden's method has global convergence provided that the system matrix is invertible [4], *i.e.*, the sequence generated by this method converges to the solution of the system for any starting point x_0 and for any starting matrix B_0 (or H_0). Moreover, in this case the convergence is finite [5], and the solution is obtained in at most $2n$ steps; conditions in which this method requires a full $2n$ steps to converge are also given. In the same paper [5], it is shown that Broyden's method has $2n$-step Q-quadratic local convergence for the nonlinear case (the initial starting point x_0 and the initial starting matrix B_0 must be close to the solution point of the system and to the Jacobian matrix in the solution point, respectively). Note that $2n$*-step Q-quadratic convergence* means that the subsequence $\{x_{2n}\}$ of $\{x_n\}$ has Q-quadratic convergence.

In this paper we propose some new variants of parallel Broyden's method that are suitable both for linear and nonlinear systems. We prove the convergence of our basic algorithm in the linear case. Numerical experiments are presented for linear and nonlinear cases, and some remarks are made concerning the behavior of the generated sequence.

2. The Parallel Variants and Preliminaries

In the case of a linear mapping, $F(x) = Ax - b$, the Broyden's "good formula" [6,7], $B^+ = B + \theta(y - Bs)s^T/s^Ts$ (B and B^+ are the current and the next iterates, respectively, and θ is a positive number chosen such that B^+ is invertible [4]) can be written in the following form: $B^+ = B - \theta(B - A)ss^T/s^Ts$. Using the notation $E = B - A$, the Broyden's method becomes $x^+ = x - (E + A)^{-1}F(x)$, $E^+ = E - \theta Ess^T/s^Ts$. The main idea of the proposed variants is to use instead of A the block diagonal matrix of A, denoted by D throughout the paper, partitioned according to $\mathcal{P}$. The computer routine $dg(A)$ will produce such a block diagonal matrix, *i.e.*, $D := dg(A)$.

Based on this idea we can consider several parallel variants of Broyden's algorithm. The first variant is just the above mentioned algorithm, to which we added a supplementary step to successively block-diagonalize the matrix E (the Step 5, Algorithm 3 below). We consider also the following slight modification of update formula

$$\widetilde{E}^+ := E - \theta\frac{(E + D)s_n s_n^T}{\|s_n\|^2}.$$

By adding D to both sides of this formula, the algorithm becomes more homogeneous (the same matrix is used in both Steps 2 and 4, Algorithm 3 below). Moreover (and not in the least) for this variant, a simple and elegant proof can be given for the convergence of the generated sequence (Theorem 1). Therefore the first our variant, which will be considered as the basic algorithm, is (E_0 is a block diagonal matrix):

Algorithm 3.

Input: $x_0,\ E_0,\ nmax$

Output: x_n

1: **For** $n = 0, 1, ..$ **Until** $convergence$ **Or** $n > nmax$ **Do**

2: $x_{n+1} := x_n - (E_n + D)^{-1}F(x_n)$;

3: $s_n := x_{n+1} - x_n$;

4: $\widetilde{E}_{n+1} := E_n - \theta\frac{(E_n+D)s_n s_n^T}{\|s_n\|^2}$;

5: $E_{n+1} := dg(\widetilde{E}_{n+1})$.

6. **End**

A variant of Algorithm 3 can be obtained by applying the Sherman–Morrison lemma to avoid the inversion of a matrix in Step 2. It results the following Algorithm 4 (B_0 is a block diagonal matrix):

Algorithm 4.

Input: $x_0,\ B_0,\ nmax$
Output: x_n

1: **For** $n = 0, 1, ..$ **Until** $convergence$ **Or** $n > nmax$ **Do**
2: $x_{n+1} := x_n - B_n F(x_n)$;
3: $s_n := x_{n+1} - x_n$;
4: $\widetilde{B}_{n+1} := (I + \frac{\theta}{1-\theta}\frac{s_n s_n^T}{\|s_n\|^2})B_n$:
5: $B_{n+1} := dg(\widetilde{B}_{n+1})$.
6. **End**

The simplest way to design a similar parallel algorithm for the nonlinear case is to replace D in the basic algorithm with the diagonal block of the Jacobian of F, $D = D(x) = dg(J(x))$. It results the following Algorithm 5 for the nonlinear case:

Algorithm 5.

Input: $x_0,\ E_0,\ nmax$
Output: x_n

1: **For** $n = 0, 1, ..$ **Until** $convergence$ **Or** $n > nmax$ **Do**
2: $x_{n+1} := x_n - (E_n + D(x_n))^{-1} F(x_n)$;
3: $s_n := x_{n+1} - x_n$;
4: $\widetilde{E}_{n+1} := E_n - \theta\frac{(E_n + D(x_n))s_n s_n^T}{\|s_n\|^2}$;
5: $E_{n+1} := dg(\widetilde{E}_{n+1})$.
6. **End**

Remark 1. *The formal replacement of D with $D(x_n)$ is based on the mean value theorem $F(x_{n+1}) \approx F(x_n) + J(x_n)(x_{n+1} - x_n)$. However, the convergence properties of Algorithm 5, including its specific conditions, rate of convergence,* etc.*, must be further analyzed as well. The numerical experiments (Section 5) performed so far for certain nonlinear systems show a comparable behavior of the two Algorithms (3 and 5) for the linear and nonlinear case.*

The main processing of the Algorithms 3, 5, is the computation of $(E_n + D)^{-1}$ or the solution of a corresponding linear system. Taking into account that $E_n + D$ is a block diagonal matrix, $E_n + D = diag((E_n + D)_1, ..., (E_n + D)_p)$, where $(E_n + D)_i$, $i = 1, ..., p$, are matrices of dimension m_i, we have to perform p subtasks, whose main processing is $(E_n + D)_i^{-1}$. It is obvious that these subtasks can be done in parallel. The successive updates of the matrix E, Steps 4 and 5, can also be executed in parallel. The concrete parallel implementation of this algorithm is an interesting problem and is left for a future research.

As usual $\mathcal{R}^{m\times m}$ denotes the set of square $m \times m$ real matrices. We use $\|\cdot\|$ and $\|\cdot\|_F$ to denote the spectral and Frobenius norm respectively. The properties of Algorithm 3 are based on the following two lemmas.

Lemma 1. *Let $D \in \mathcal{R}^{m\times m}$ be a block diagonal matrix (conforming with $\mathcal{P}$). Then, for any matrix $E \in \mathcal{R}^{m\times m}$, there holds*

$$\|dg(E) + D\|_p \leq \|E + D\|_p,$$

where $\|\cdot\|_p$ is the spectral or Frobenius norm. If $\theta \in [0, 2]$ then the sequence $\{\|E_n\|\}$ as defined in Algorithm 3 is bounded, $\|E_n\| \leq a$.

Proof. Using the inequality $\|dg(A)\|_p \leq \|A\|_p,\ A \in \mathcal{R}^{m\times m}$ (which is true for both spectral and Frobenius norm), we have

$$\|dg(E) + D\|_p = \|dg(E + D)\|_p \leq \|E + D\|_p.$$

To prove the second part of Lemma 1, observe that if $\theta \in [0, 2]$ then $\|I - \theta\frac{ss^T}{\|s\|^2}\| = 1$ for any $s \in \mathcal{R}^m$. Therefore

$$\|E_{n+1} + D\| \leq \|\tilde{E}_{n+1} + D\| = \|E_n + D - \theta\frac{(E_n + D)s_n s_n^T}{\|s_n\|^2}\| \leq \|E_n + D\|\|I - \theta\frac{s_n s_n^T}{\|s_n\|^2}\| = \|E_n + D\|.$$

Thus $\|E_n + D\| \leq \|E_0 + D\|,\ \forall n$ and $\{\|E_n\|\}$ is bounded. □

Lemma 2. *For any matrix $M \in \mathcal{R}^{m\times m}$, any $s \in \mathcal{R}^m$, any $\theta \in \mathcal{R}$, and $\widetilde{M} := M - \theta Mss^T/\|s\|^2$ we have*

$$\|\widetilde{M}\|_F^2 = \|M\|_F^2 - \theta(2 - \theta)\frac{\|Ms\|^2}{\|s\|^2}. \tag{1}$$

(The Formula (1) appears also in [4]; a short proof is given here for completeness).

Proof. By trivial computation we obtain

$$\begin{aligned}\|\widetilde{M}\|_F^2 &= \|M - \theta\frac{Mss^T}{\|s\|^2}\|_F^2 \\ &= \|M\|_F^2 - 2\frac{\theta}{\|s\|^2}\langle M, Mss^T\rangle_F + \frac{\theta^2}{\|s\|^4}\|Mss^T\|_F^2.\end{aligned}$$

Use $\langle M, Mxx^T\rangle_F = \|Mx\|^2$, $\|Mxx^T\|_F = \|Mx\|\|x\|$, and the desired equality results. □

3. The Convergence of Algorithm 3 in the Linear Case

In the following, we suppose that the sequence $\{x_n\}$ given by Algorithm 3, with starting point x_0, starting matrix E_0 and certain θ, satisfies the condition:

$$\|x_{n+1} - x_n\| \leq \beta,\ \forall n \in \mathcal{N}. \tag{2}$$

Algorithm 3 defines the next iteration as a function of the current iteration, $x_{n+1} = G(x_n)$, G being the iteration function or generation function. It is clear that the condition (2) is weaker than the condition of asymptotic regularity of G or asymptotic regularity of x_0 under G, ($\|G^{n+1}x_0 - G^n x_0\| \to 0,\ n \to \infty$).

The fulfillment of condition (2) depends not only on the characteristics of A (like the condition number) but also on x_0, E_0 and θ. Note also that usually, the sequence $\{E_n\}$ is bounded, and if F is also bounded on $\mathcal{R}^m$ then the condition (2) is satisfied; in Section 5 an example of mapping is given which is bounded on $\mathcal{R}^m$ and $\|D(x_n)^{-1}\|\|E_n\| < 1$.

Theorem 1. *Let $A \in \mathcal{R}^{m\times m}$ be a nonsingular matrix, $D = dg(A)$, and let θ be a real number, $0 < \theta < 2$. Suppose that D is invertible and that $a\|D^{-1}\| < 1$, where a is defined in Lemma 1. Suppose also that condition (2) is satisfied and that $E_0 + D$ is invertible. Then the parallel variant of Broyden method (Algorithm 3) converges to the solution of the equation $Ax - b = 0$.*

Proof. Since $\|E_n\|\|D^{-1}\| \leq \|D^{-1}\|a < 1$, from perturbation Lemma, it results that $(E_n + D)^{-1}$ exists for all n and the sequence $\{x_n\}$ is well defined by Algorithm 3. By taking $M = E_k + D,\ s = s_k$ and applying Lemmas 1 and 2, we obtain

$$\|E_{k+1} + D\|_F^2 \leq \|E_k + D\|_F^2 - \theta(2-\theta)\frac{\|(E_k + D)s_k\|^2}{\|s_k\|^2}.$$

By summing on k, $k = 0, 1, ..., n$, we have

$$0 \leq \|E_{n+1} + D\|_F^2 \leq \|E_0 + D\|_F^2 - \theta(2-\theta)\sum_{k=0}^{\infty}\frac{\|(E_k + D)s_k\|^2}{\|s_k\|^2}.$$

This implies that

$$\frac{\|(E_n + D)s_n\|}{\|s_n\|} \to 0,\ n \to \infty.$$

Now, because $(E_n + D)s_n = -F(x_n)$, we have

$$\frac{\|F(x_n)\|}{\beta} \leq \frac{\|F(x_n)\|}{\|s_n\|} = \frac{\|(E_n + D)s_n\|}{\|s_n\|} \to 0,$$

and

$$\|A^{-1}\|^{-1}\|x_n - x^*\| \leq \|A(x_n - x^*)\| = \|F(x_n)\| \to 0.$$

□

4. Remarks on the Parallel Implementation

The parallelization of the product $(E_n + D)^{-1}F(x_n)$ (Step 2 in Algorithm 3) is obvious, because $E_n + D$ is block diagonal. Concerning the product $(E_n + D)s_n s_n^T$, observe first that the element p_{ij} of the product ss^T is $p_{ij} = s_i s_j$ and $s_n s_n^T$ can be computed by a very simple algorithm. Then, because $E_n + D$ is block diagonal, $E_n + D = diag((E_n + D)_1, ..., (E_n + D)_p)$ with dimensions $m_1, ..., m_p$, the product $(E_n + D)_i s_n s_n^T$ gives the m_i lines of $(E_n + D)s_n s_n^T$. These products are independent from each other and therefore can be computed in parallel. The same observations can be made for Algorithms 4 and 5.

In order to implement this method, let us suppose that the multiprocessor system has p processors. A simple idea is to set the system into p partitions and assign a partition to every processor. In this

case the following issue arises: How large should the partitions $m_i,\ i = 1, ..., p$ be in order to obtain a low cumulative computation cost per iteration? In the case of Algorithms 3 and 5, the main processing of processor i is to inverse a matrix of dimension m_i. Therefore every processor has to perform a computation that has a polynomial complexity of order m_i^3. The cumulative computation cost for all p processors is $m_1^3 + m_2^3 + ... + m_p^3$, and, if there is not overlapping, $m_1 + m_2 + ... + m_p = m$. Further, we use the following elementary inequality

$$\sum_{i=1}^{p} x_i^q \leq (p-1)\mu^q + (\sum_{i=1}^{p} x_i - (p-1)\mu)^q,$$

where x_i are positive numbers and $\mu = \min\{x_1, ..., x_p\}$. If $q = 3$ and $x_i = m_i$, then

$$\sum_{i=1}^{p} m_i^3 \leq (p-1)\mu^3 + (m-(p-1)\mu)^3.$$

We will prove now that if

$$\frac{m}{(p^2-p+1)^{1/3}+p-1} \leq \mu \leq \frac{m}{p}, \tag{3}$$

then

$$(p-1)\mu^3 + (m-(p-1)\mu)^3 \leq \frac{m^3}{p}.$$

Because $p\mu \leq m$, it is sufficient to show that $(p-1)\mu^3 + (m-(p-1)\mu)^3 \leq p^2\mu^3$, and this is equivalent with the first part of (3). We obtain the following

Propositon 1. *Suppose that the smallest size of partitions, $\mu = \min\{m_1, ..., m_p\}$, satisfies the condition (3). Then the cumulative computation cost in Algorithms 3 and 5 is less than m^3/p.*

A similar result can be established for Algorithm 4. The condition (3) becomes

$$\frac{m}{(p+1)^{1/2}+p-1} \leq \mu \leq \frac{m}{p},$$

and the cumulative computation cost satisfies $\sum_{i=1}^{p} m_i^2 \leq 2m^2/p$.

5. Numerical Experiments and Conclusions

The main purpose of this section is to highlight the convergence properties of the proposed parallel Broyden's algorithms in the case of a synchronous model. The communication costs between processors, scheduling and loads balancing, optimal processors assignment, and so on, both for synchronous and asynchronous models, are important issues in themselves. However they are not the subject of the present study.

The behavior of the sequence $\{x_n\}$ obtained by the proposed algorithms and for various linear or nonlinear cases is shown in our experiments by the graph of $ln\varepsilon_n$, where ε_n is the error at step n, $\varepsilon_n = \|F(x_n)\|$ or $\varepsilon_n = \|x_n - x^*\|$ (n on horizontal axis and $ln\varepsilon_n$ on vertical axis). The convergence of the process entails a decreasing graph until negative values (for example, $ln\varepsilon_n = -30$ means $\varepsilon_n \approx 10^{-15}$). If the convergence is linear ($r = 1$) then $ln\varepsilon_n \approx nln\varrho + ln\varepsilon_0$; $ln\varepsilon_n$ depends linearly on n and the graph

of $ln\varepsilon_n$ is close to a straight line. For comparison reasons, the proposed parallel Broyden's method is compared with the parallel Newton method and the parallel Cimmino method. We consider the following parallel Newton method [8]:

$$x_{n+1} = x_n - D(x_n)^{-1}F(x_n),\ D(x_n) = dg(J(x_n)),$$

where $J(x_n)$ is the Jacobian of F. Note that in the case of a linear system, by applying this form of Newton method, the direct computation of x^* (in one iteration) is lost and the convergence is linear. Consequently, the graph of $ln(\varepsilon_n)$ is a straight line.

The block Cimmino method is considered only for the linear case, $F(x) = Ax + b$, and it can be described as follows (see, for example, [9]). The system is partitioned into p strips of rows, A^i and b^i, $1 \leq i \leq p$. Supposing that A^i has full row rank, the Moore–Penrose pseudo inverse of A^i is defined by $A^{i^+} = A^{i^T}(A^i A^{i^T})^{-1}$. The block Cimmino algorithm is

Step 1: Compute in parallel $Q_n^i = A^{i^+}(A^i x_n + b^i)$;
Step 2: $x_{n+1} = x_n + \omega \sum_{i=1}^{p} Q_n^i$,
where ω is a relaxation parameter.

Experiment 1.

We applied Algorithm 3 to linear systems of medium dimensions, $m = 20, 50, 100, 500, 600$, with different values of condition numbers. A program "genA" generates a square, sparse matrix A of dimension m with random elements $a_{ij} \in (-1, 1)$, $i \neq j$ and $a_{ii} \in (d1, d2)$, where $d1, d2$ are given (positive) numbers. The percent of nonzero elements is an entry data of "genA" and in this experiment we considered it to be about 50%. The free term b and the starting point x_0 were also randomly generated with elements between some limits. Thus the function F is randomly defined by $F(x) = Ax - b$.

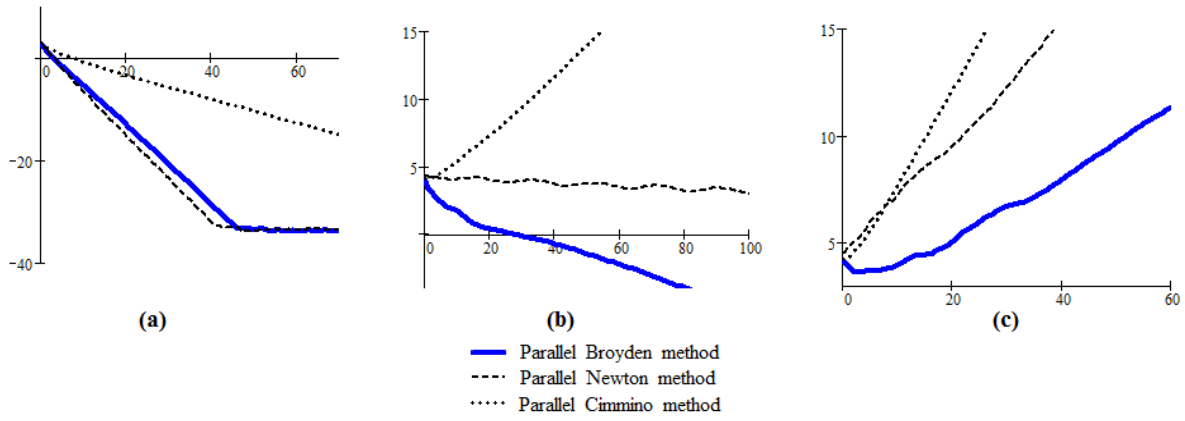

Figure 1. The graphs of $ln(\varepsilon_n)$ generated by Algorithm 3, parallel Newton method and parallel Cimmino method in the linear case; (**a**) system well conditioned; (**b**) system medium conditioned; (**c**) system poorly conditioned.

For every considered system, we applied Algorithm 3, the parallel Newton method and the parallel Cimmino method, and we drew the resulting graphs in the same plot; for the graph of Algorithm 3 we used a continuous line, for the parallel Newton method a dashed line, and for the parallel Cimmino

method a dotted line. The number of partitions and the dimensions of every partition are entry data of the routine "diag(A)". The test systems were automatically generated by routine "genA"; the three graphs of Figure 1 are examples from a large number of cases, corresponding to certain systems of dimensions 50, 100, 200 respectively; other parameters like c (the condition number), θ (the factor in Step 4), $p(n_1, ..., n_p)$ (the number of partitions and their dimensions), were chosen as follows:

(a) $m = 50,\ c = 3.569,\ \theta = 0.02,\ p(...) = 5(11, 9, 13, 11, 6),\ E_0 = I,$
(b) $m = 100,\ c = 71.69,\ \theta = 0.02,\ p(...) = 5(21, 19, 23, 21, 16),\ E_0 = I,$
(c) $m = 200,\ c = 785,\ \theta = 0.02,\ p(...) = 5(41, 39, 43, 41, 36),\ E_0 = I.$

The following conclusions can be drawn. (1) The proposed parallel Broyden's algorithm works well if the system is relatively well conditioned (the condition number should be around ten), and in this case the behavior of Algorithm 3 is very similar to the parallel Newton method and significantly better than the parallel Cimmino method. (Figure 1a); (2) For medium conditioned systems, the behavior of Algorithm 3 is unpredictable, sometimes it works better than the Newton method (Figure 1b); (3) If the system is poorly conditioned (the condition number is greater than 300) the proposed parallel Broyden's algorithm fails to converge (the Newton method has the same behavior) (Figure 1c).

Figure 2a presents the behavior of the sequence generated by the Algorithm 3 for a linear system of dimension $m = 500,\ c = 4.366,\ p(...) = 5(101, 51, 151, 101, 96),\ \theta = 0.2,\ E_0 = I.$ The condition (2) of Theorem 1 is not satisfied and the sequence generated by this algorithm (in this case) does not converge. However, the behavior is interesting, the generated sequence becomes very close to the solution, the error decreases until a very small value (in our example until 10^{-12}), and then the good behavior is broken. The graph corresponding to the sequence generated by Algorithm 4 for a system of dimension $m = 600$ and $c = 3.942,\ p(...) = 6(101, 51, 151, 101, 101, 95),\ \theta = 0.03,\ E_0 = (I + D)^{-1}$ is presented in Figure 2b. We can observe the similarities between this sequence and the sequence obtained by the parallel Newton method.

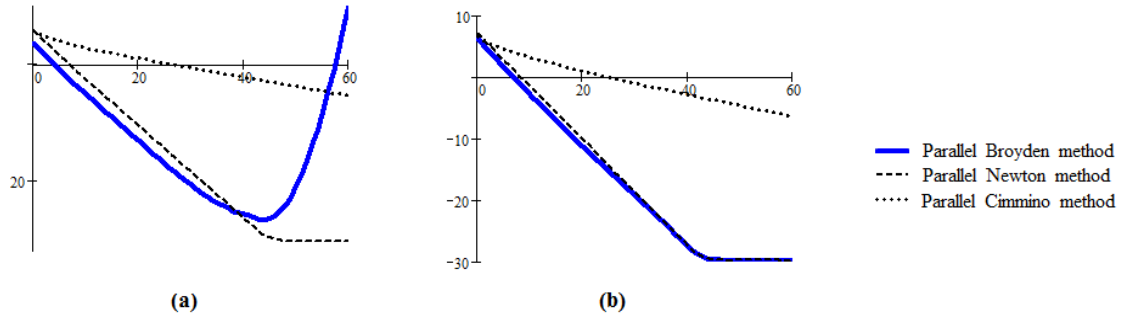

Figure 2. The behavior of the sequence generated by Algorithm 3, Parallel Newton method and Parallel Cimmino method (the graphs (a)) and by Algorithm 4, Parallel Newton method and Parallel Cimmino method (the graphs (b)); (**a**) system which does not satisfy the condition (2); (**b**) system well conditioned.

Remark 2. *The numerical experiments presented here, for the most part, have been performed with the basic Algorithm 3; for Algorithm 4 similar conclusions can be drawn. A special characteristics of Algorithm 4, in comparison with Algorithm 3, is that Algorithm 4 is less sensitive to θ; this parameter can take its values on a much larger interval, $\theta \in (0, 2)$.*

Experiment 2.

This experiment is devoted to show the behavior of the sequences generated by Algorithm 5 in the case of nonlinear systems. We considered several nonlinear systems of low dimensions and of certain sparsity (every nonlinear equation depends on a few numbers of unknowns). Figure 3 presents the results for the following three nonlinear systems:

$$(1)\begin{cases} 4x_1^2 - x_2x_6 - 3 = 0, \\ x_1 - 5x_2 + x_3^2 + x_7 = 0, \\ -x_1 + 5x_3 - x_4^2 - 3 = 0, \\ x_1x_2 - 4x_4 + x_5^2 + 2 = 0, \\ x_4 - 5x_5 + 2x_6 = 0, \\ x_1 - 6x_6 + 4 = 0, \\ x_2 + x_3^2 - 4x_7 + 2 = 0. \end{cases} \quad (2)\begin{cases} 3x_1 + x_2^2x_3 = 0, \\ 2x_2 + x_3 - 2x_7^2 = 0, \\ x_1 - 0.2x_2x_4 + 3x_3 = 0, \\ 3x_4 - x_6^2 = 0, \\ x_2 + 2x_5 = 0, \\ x_3x_7 + 5x_6 = 0, \\ x_1^2 + x_2 + x_5 + 4x_7 = 0. \end{cases} \quad (3)\begin{cases} 5\frac{x_1^2+x_2^2}{x_1^2+x_2^2+1} - \frac{10}{3} = 0, \\ 4\frac{x_3^2}{x_3^2+1} - 2 = 0, \\ 3\frac{x_2^2+x_3^2}{x_2^2+x_3^2+1} = 0. \end{cases}$$

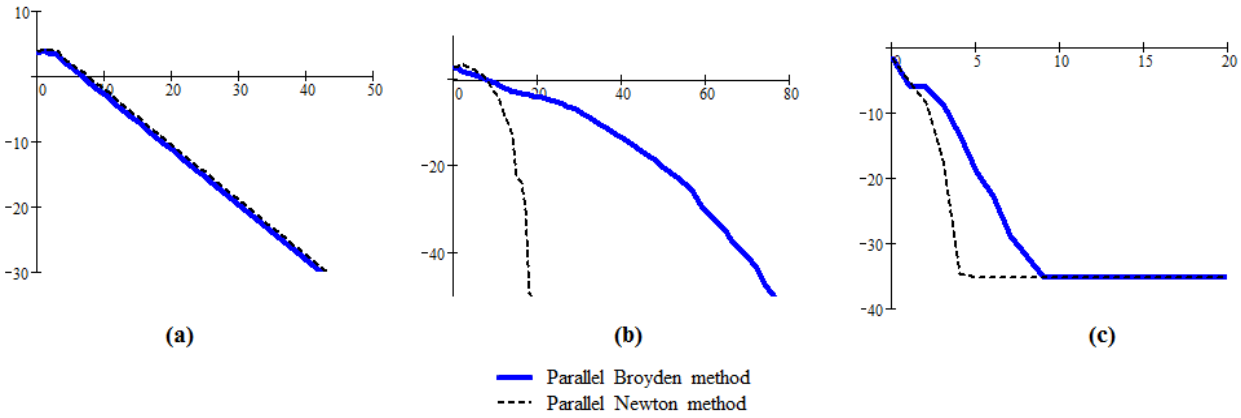

Figure 3. The graphs of $ln(\varepsilon_n)$ generated by Algorithm 5 and parallel Newton method; (**a**) the graph corresponding to systems (1); (**b**) the graph corresponding to systems (2) given by Algorithm 5 modified in accordance with Sherman-Morrison lemma; (**c**) example of a system verifying the condition $\|D(x_n)^{-1}\|\|E_n\| < 1$.

The proposed parallel Broyden's method is compared with the parallel Newton method described above. The conclusion is that, generally, the parallel Broyden's method has similar characteristics with the considered parallel Newton method, convergence properties, attraction basin, *etc.* This similarity can

be seen in the case of the first system (Figure 3a). The third system is an example for which F is bounded on $\mathcal{R}^3$, as it is required in Section 3. Note that the condition $\|D(x_n)^{-1}\|\|E_n\| < 1$ is also satisfied.

Experiment 3.

This experiment is devoted to compare Algorithm 3 with the parallel variants proposed in [1] and [2,3]. Three linear systems of low dimensions (m = 5) with condition numbers equal to $8.598,\ 4.624,\ 3.701$ were considered as test systems. Every system was set into two partitions of dimensions 3 and 2 respectively.

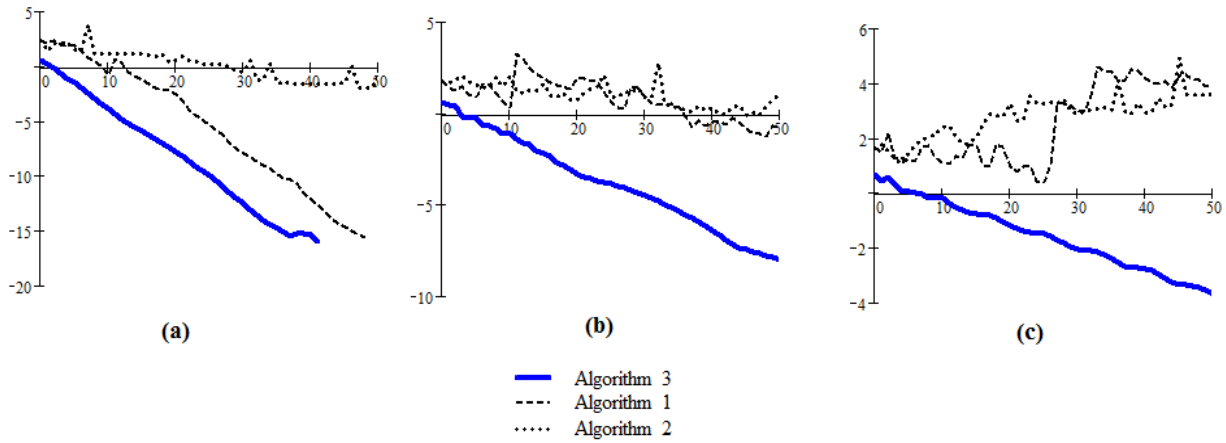

Figure 4. The comparison of Algorithms 3, 1 and 2 in the care of linear systems of low dimensions (m = 5); (**a**) condition number = 8.598; (**b**) condition number=4.624; (**c**) condition number = 3.624.

The results are presented in Figure 4. We can conclude that the behavior of Algorithm 3 is satisfactory, as in all cases the generated sequence converges to the solution of the system and the convergence is linear. It is worthwhile to note the very good behavior of Algorithm 3 in comparison with the other two variants for cases (b) and (c): in case (b) the Algorithms 1 and 2 appear to have a very slow convergence, and in case (c) these algorithms fail to converge.

Acknowledgment

The authors are very grateful to the anonymous referees for valuable remarks and comments, which significantly contributed to the quality of the paper.

This research was supported of West University of Timisoara, Romania, Agreement of Academic and Research Activities, No. 12910/21.05.2015.

Author Contributions

Stefan Maruster theoretical approach; Ioan Bistran and Liviu Octavian Mafteiu-Scai designed and performed the experiments.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Xu, J.J. Convergence of partially asynchronous block quasi-Newton methods for nonlinear systems of equations. *J. Comput. Appl. Math.* **1999**, *103*, 307–321.
2. Jiang, P.; Yang, G. Performance analysis of preconditioners based on Broyden method. *Appl. Math. Comput.* **2006**, *178*, 295–308.
3. Yang, G.; Jiang, P. SSOR and ASSOR preconditioners for block Broyden method. *Appl. Math. Comput.* **2007**, *188*, 194–205.
4. More, J.J.; Trangenstein, J.A. On the global convergence of Broyden's method. *Math. Comput.* **1976**, *30*, 523–540.
5. Gay, D.M. Some convergence properties of Broyden's method. *SIAM J. Numer. Anal.* **1979**, *16*, 623–630.
6. Dennis, J.E., Jr.; Schnabel, R.B. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*; Prentice-Hall, Inc.: Upper Saddle River, NJ, USA, 1983.
7. Martinez, J.M. *Algorithms for Solving Nonlinear Systems of Equations*; Springer: Heidelberg, Germany, 1994.
8. Lazar, I. On the convergence of asynchronous block Newton method for nonlinear systems of equations. *Informatica* **2002**, *47*, 75–84.
9. Balsa, C.; Guiverch, R.; Raimundo, J.; Ruiz, D. MUMPS Based Approach to Parallelize the Block Cimmino Algorithm. In Proceedings of the 8th International Meeting High Performance Computing for Computational Science, Toulouse, France, 1 Junuary 2008.