Asynchronous Block-Iterative Methods for Almost Linear Equations

Amit Bhaya, Eugenius Kaszkurewicz, and Francisco Mota*
COPPE / UFRJ
Federal University of Rio de Janeiro
Department of Electrical Engineering
C.P. 68504
Rio de Janeiro, RJ 21945, Brazil

Dedicated to Gene Golub, Richard Varga, and David Young

Submitted by Michael Neumann

ABSTRACT

This paper gives convergence conditions for asynchronous block-iterative methods for the solution of the almost linear equation \( Ax = F(x) \), where \( A \) is a linear operator, \( F \) a block-diagonal Lipschitz-continuous operator, and \( x \) a vector, in terms of a splitting of \( A \) and the Lipschitz constant of \( F \). The methods used are a combination of the contraction-mapping approach using a vectorial norm and a large-scale systems approach using vector difference inequalities. The load-flow equations for a power system are almost linear in the above sense, and considerable speedup can be obtained on a four transputer machine.

1. INTRODUCTION

A great deal of research is currently being focused on the efficient implementation of iterative algorithms on parallel computers (Ortega, 1988,

*E-mail: na.bhaya@na-net.orl.gov or COE10027@UFRJ.BITNET.
and references therein). In particular, asynchronous implementations, in which computation and communication are performed independently in each processor, are attracting a lot of attention on account of several potential advantages such as reduction of processor idle times, shorter time to convergence, simpler programming, and so on (Bertsekas and Tsitsiklis, 1989).

In this paper, we consider asynchronous block-iterative methods to find the fixed point of the almost linear equation $Ax = F(x)$, where $x$ is an $n$-vector, $A$ a nonsingular linear operator, and $F$ a diagonal or block-diagonal Lipschitz-continuous operator. For this commonly occurring class of fixed-point problems it is natural to consider the synchronous and asynchronous versions of the classical block-iterative methods such as Jacobi and SOR (Varga, 1962; Young, 1971; Golub and Van Loan, 1989) which are based on a splitting of the matrix associated to the linear operator $A$. Sufficient conditions for the convergence of an asynchronous iteration have been given by several authors and can be broadly classified as follows: (a) the iteration operator is required to be a contraction in a vectorial norm (Baudet, 1978; Miellou, 1974; Robert, 1976); (b) the iteration operator is required to be a contraction in a weighted maximum norm (El Tarazi, 1982; Bertsekas and Tsitsiklis, 1989, and references therein); and (c) a Liapunov function of the weighted maximum-norm type is used to prove asymptotic stability of the iteration (Kaszkurewicz, Bhaya, and Šiljak, 1990). Using a result of Miellou (1974), El Tarazi (1982) pointed out that conditions of type (b) generalize (i.e. are weaker than) conditions of type (a), while the Liapunov approach (c) is essentially equivalent to approach (b).

In Section 2, we present a hybrid approach to the problem of finding conditions for convergence of an asynchronous block iteration used to solve an almost linear equation: a vectorial norm is used to set up an asynchronous difference inequality, and then the Liapunov approach is used to prove stability of the associated comparison vector difference equation. Although results similar to those derived by the above approach can be derived (under slightly different hypotheses) using an approach of the type (b) above, we believe that the simplicity of our approach and the connections that it makes between “large-scale” techniques on the one hand and numerical analysis techniques on the other justify its presentation. Finally, we note that related ideas are used in a different context by Mitra (1987) to prove convergence of asynchronous relaxations for ordinary differential equations.

In Section 3 we apply the asynchronous method proposed in this paper to an example of an almost linear equation arising in a real power system. The power system is only described briefly, since the main concern is to show the speedup achievable with asynchronism.

Finally, in Section 4, we make some concluding remarks and give some suggestions for further research.
2. ASYNCHRONOUS BLOCK ITERATION FOR ALMOST LINEAR EQUATIONS

In this section we consider block-iterative methods for large systems of equations of the form

\[ Ax = F(x), \quad A \in \mathbb{R}^{n \times n}, \quad F : \mathbb{R}^n \to \mathbb{R}^n, \]  

in the special case where \( F \) is a block-diagonal Lipschitz-continuous operator. The class of methods we consider is suitable for asynchronous implementation on a distributed-memory multiprocessor. Our objective is to find conditions on a block splitting of the matrix \( A \) and conditions on \( F \) such that the corresponding asynchronous block-iterative method converges.

The special case of a block-diagonal, Lipschitz-continuous nonlinear operator \( F \) is quite common in practical applications such as the load-flow problem for electrical networks and is also susceptible to an analysis inspired by large-scale systems techniques. The main ideas of our analysis are: (i) to manipulate the asynchronous error equation so that it has the form of a standard vector difference inequality; (ii) to use the Liapunov stability theorem of Kaszkurewicz, Bhaya, and Šiljak (1990) (Appendix A below) to prove exponential stability of the associated comparison vector difference equation; and (iii) to use the comparison lemma (Appendix B) to show that a suitable norm of the error tends to zero. To formalize these ideas, we will introduce some notation and assumptions below.

Let a Cartesian product decomposition of \( \mathbb{R}^n \) be given:

\[ \mathbb{R}^n = \mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_m} = n_1 + \cdots + n_m - n. \]  

A block splitting of the matrix \( A \) conformal with the decomposition (2) is given:

\[ A = M - N, \]  

where \( M \) is block-diagonal, and if \( m = \{1, 2, \ldots, m\} \),

\[ M = \text{diag}(M_1, \ldots, M_m), \quad M_i \in \mathbb{R}^{n_i \times n_i}, \quad \det M_i \neq 0 \quad \forall i \in m, \]  

\[ N_{ij} \in \mathbb{R}^{n_i \times n_j}, \quad N := (N_{ij}). \]

Note that (4) implies that \( M \) is nonsingular.
Let $D \subset \mathbb{R}^n$ and $D = D_1 \times \cdots \times D_m$ with $D_i \subset \mathbb{R}^{n_i} \forall i \in m$. We assume that $F : D \to F(D)$ is block-diagonal, i.e., if $x = (x_1^T, \ldots, x_m^T)^T \in D$, $x_i \in D_i \forall i \in m$, and $F = (F_1, \ldots, F_m)^T$, then $F_i$ depends only on $x_i$. More precisely, we have

**Assumption 1 (Block-diagonal nonlinearity).**

$$D \subset \mathbb{R}^n, \quad D = D_1 \times \cdots \times D_m, \quad D_i \subset \mathbb{R}^{n_i} \forall i \in m,$$

and

$$F_i : D_i \to F_i(D_i) : x_i \mapsto F_i(x_i) \quad \forall i \in m,$$

$$F : (x_1^T, \ldots, x_m^T)^T \to \left( F_1(x_1)^T, \ldots, F_m(x_m)^T \right)^T.$$

We also assume that a single norm $\| \cdot \|$ is used on all spaces $\mathbb{R}^{n_i}$, $i = 1, \ldots, m$, and on $\mathbb{R}^n$ as well, and that each $F_i$ is Lipschitz-continuous with constant $L_{F_i}$ with respect to this norm. In other words, we have

**Assumption 2 (Lipschitz continuity).**

$$\forall i \in m, \quad \forall x_i, y_i \in D_i, \quad \| F_i(x_i) - F_i(y_i) \| \leq L_{F_i} \| x_i - y_i \|.$$

Substituting the splitting (3) in (1) yields

$$x = M^{-1}Nx + M^{-1}F(x) = G(x),$$

where $G = M^{-1}N + M^{-1}F$. The equation (1) has a solution in a set $D$ iff there exists a fixed point $x^* \in D$ for $G$, i.e., $x^* = G(x^*)$. Throughout this paper we will make the following important assumption.

**Assumption 3 (Uniqueness of fixed point).** There exists a closed set $D \subset \mathbb{R}^n$ such that $G(D) \subset D$ [where $G$ is defined in (8) above], which contains exactly one fixed point $x^*$ of the operator $G$. 
Let $D$ admit a Cartesian decomposition as in Assumption 1. Then using (4)–(6) and denoting the fixed point by $x^* = (x_1^*, \ldots, x_n^*)^T$ gives

$$\forall i \in m, \quad x_i^* = \sum_{j=1}^{n} M^{-1}_{i,j} N_{i,j} x_j^* + M^{-1}_{i} F_i(x_i^*),$$

(9)

which is the fixed-point equation that must be satisfied by all solutions of (1).

Asynchronism refers to the possibility of using delayed or "old" variables in iterative methods based on (8). To be more specific, we will assume that, at time instant $k$, the $i$th processor (which updates $x_i$) receives information from the $j$th processor with a time-varying delay of $k - d_{ij}(k)$ units. We make the important assumption that the delays are uniformly bounded in time (over all processors) by a positive integer $d$. This is stated formally as a restriction on the range of the positive integer-valued functions $d_{ij}(\cdot)$ in Assumption 4 below.

**Assumption 4 (Uniform bound on delays).**

$$\exists d \in \mathbb{N}, \quad \forall k \in \mathbb{N}, \quad \forall i, j \in m \quad d_{ij}(k) \in \{k, k-1, \ldots, k-d\}.$$  (10)

**Comment 1.** We emphasize that the $d_{ij}(k)$'s are functions of three variables: time ($k$), sending processor ($j$), and receiving processor ($i$). We will return to these points below. This uniformly-bounded-delay asynchronism is referred to by Bertsekas and Tsitsiklis (1989) as "partial asynchronism" to distinguish it from the case of possibly unbounded delays. We also point out that this bound is enforceable in practice, by the use of the so-called synchronization barriers, at which all processors are forced to stop and communicate before proceeding with their individual computations.

Using this notation, we write the general bounded-delay asynchronous block-iterative method based on (8) as

$$\forall i \in m, \quad x_i(k+1) = \sum_{j=1}^{m} M_{i,j}^{-1} N_{i,j} x_j(d_{ij}(k)) + M_{i}^{-1} F_i(x_i(d_{ij}(k))).$$  (11)

Subtracting (9) from (11) yields the following asynchronous error equation
for the $i$th component:

$$\forall i \in \mathbf{m}, \quad x_i(k+1) - x_i^* = \sum_{j=1}^{m} M_i^{-1} N_{ij} \left( x_j(d_j^i(k)) - x_j^* \right)$$

$$+ M_i^{-1} \left[ F_i(x_i(d_i^i(k))) - F_i(x_i^*) \right]. \quad (12)$$

We now define two error vectors:

$$\forall i, j \in \mathbf{m}, \quad e_i(k+1) := x_i(k+1) - x_i^*, \quad e_j(d_j^i(k)) := x_j(d_j^i(k)) - x_j^*. \quad (13)$$

Then (12) can be rewritten as

$$\forall i \in \mathbf{m}, \quad e_i(k+1) = \sum_{j=1}^{m} M_i^{-1} N_{ij} e_j(d_j^i(k))$$

$$+ M_i^{-1} \left[ F_i(x_i(d_i^i(k))) - F_i(x_i^*) \right]. \quad (14)$$

Taking norms in (14) and using (7) gives

$$\forall i \in \mathbf{m}, \quad \|e_i(k+1)\| \leq \sum_{j=1}^{m} \|M_i^{-1} N_{ij}\| \cdot \|e_j(d_j^i(k))\|$$

$$+ M_i^{-1} \|e_i(d_i^i(k))\|. \quad (15)$$

**Comment 2.** The use of Assumption 2 above (Lipschitz continuity of the $F_i$'s in the domains $D_i$) implicitly uses the further hypothesis that \( \forall i \in \mathbf{m}, \forall k \in \mathbb{N}, x_i(d_i^i(k)) \in D_i \) and \( x_i^* \in D_i \). This is guaranteed by Assumption 3.
We introduce the following notation:

\[ z_i(k+1) := \| e_i(k+1) \|, \quad z_j(d_j^i(k)) := \| e_j(d_j^i(k)) \|, \quad (16) \]

\[ z_{i,p}(k) := \| e_i(k-p) \|, \quad p = 0, 1, \ldots, d, \quad (17) \]

\[ h_{ij} := \begin{cases} \| M_i^{-1}N_{ii} \| + \| M_i^{-1} \|_{F_1}, & i = j, \\ \| M_i^{-1}N_{ij} \|, & i \neq j; \end{cases} \quad (18) \]

\[ H := (h_{ij}). \quad (19) \]

**COMMENT 3.** All \( z \) variables are nonnegative scalar variables. The \( z_{i,p}(k) \)'s introduced in (17) are norms of delayed error variables, and by (10) we only need to consider \( p = 0, 1, \ldots, d \).

We now introduce some notation for vectorial norms. Let \( x \in \mathbb{R}^n \), with \( x^T = (x_1^T, \ldots, x_m^T) \) and \( x_i \in \mathbb{R}^{n_i} \). Then \( \| x \| := (\| x_1 \|, \ldots, \| x_m \|)^T \in \mathbb{R}^m \) and \( \| C \| = (\| C_{ij} \|) \in \mathbb{R}^{m \times m} \), where \( C = (C_{ij}) \in \mathbb{R}^{n \times n} \) and \( C_{ij} \in \mathbb{R}^{n_i \times n_j} \). Then the nonnegative \( m \times m \) matrix \( H \) can be expressed as

\[ H = \| M^{-1}N \| + \| M^{-1} \|_{F_1}, \quad (20) \]

where

\[ L_F := \text{diag}(l_{F_1}, \ldots, l_{F_m}). \quad (21) \]

The inequality (15) is now rewritten as the asynchronous difference inequality

\[ \forall i \in \mathbb{N}, \quad z_i(k+1) \leq \sum_{j=1}^m h_{ij} z_j(d_j^i(k)). \quad (22) \]

Now, by Assumption 4,

\[ \forall k \in \mathbb{N}, \quad \forall i, j \in \mathbb{M}, \quad z_j(d_j^i(k)) \in \{ z_j(k), z_{j,1}(k), \ldots, z_{j,d}(k) \}. \]
Thus, if we stack the variables $z_i(k)$ and their delayed versions $z_{i,p}(k)$ for $i = 1, \ldots, m$, $p = 1, \ldots, d$, in an augmented vector $z_a(k)$, where

$$z_a(k)^T := (z_1(k), z_{1,1}(k), \ldots, z_{1,d}(k), \ldots, z_m(k), \ldots, z_{m,d}(k)),$$  \hspace{1cm} (23)

then we can rewrite the $m$ scalar asynchronous difference inequalities (22) as a standard vector difference inequality in $\mathbb{R}_+^{m(d+1)}$ as follows:

$$\forall k \in \mathbb{N}, \quad z_a(k + 1) \leq H_a(k)z_a(k),$$  \hspace{1cm} (24)

where, for all $k$, $H_a(k)$ is an $m(d+1) \times m(d+1)$ time-varying nonnegative matrix whose elements are the (nonnegative) elements of $H$, ones and zeros (for details of its definition and interpretation, see Appendix A).

Let $y_a(k)$ be a solution of the associated comparison-vector difference equation, i.e.

$$\forall k \in \mathbb{N}, \quad y_a(k + 1) = H_a(k)y_a(k).$$  \hspace{1cm} (25)

Then, by the Liapunov stability theorem of Kaszkurewicz, Bhaya, and Šiljak (1990) (Appendix A below), $\rho(H) < 1$ is a sufficient condition for $y_a(k)$ to tend to zero exponentially as $k \to \infty$. Since, for all $k$, $H_a(k)$ is a nonnegative matrix, the comparison lemma of Appendix B now implies that $z_a(k) \to 0$, provided that (24) and (25) start from the same initial condition. This means that the asynchronous block iteration defined in (11) is locally convergent for all initial conditions in $D$. In other words, we have proved the following:

**Theorem 2.1.** Consider a matrix $A$ with a block splitting $M - N$ as in Equation (3), subject to (4), and a block-diagonal nonlinear operator $F$ with Lipschitz-continuous components $F_i$ (Assumptions 1 and 2). If the spectral radius of the nonnegative matrix $H$ (18) is strictly less than unity and the operator $G = M^{-1}N + M^{-1}F$ is subject to Assumption 3, then the bounded-delay (Assumption 4) asynchronous block iteration (11) to solve the fixed-point problem $Ax = F(x)$ (1) is locally convergent to the fixed point $x^* \in D$ for all initial conditions in $D$.

**Comment 4.** For a nonnegative matrix $H$, there are many conditions equivalent to $\rho(H) < 1$, some of which may be easier to check—for example, the condition that $I - H$ is a nonsingular $M$-matrix. For more details see
Lemma A2, Appendix A. In addition, an obvious sufficient condition is that \( \|H\| \) is strictly less than unity in any matrix norm: the infinity norm being one which lends itself to easy computation. Note also that, by Theorem A1, we actually prove local exponential stability of the asynchronous block iteration, which is somewhat stronger than proving local convergence or attractivity.

We now give some generalizations of Theorem 2.1, noting that the first one (Theorem 2.2) has not been studied using the contraction-mapping approach, but falls out of the Liapunov approach in a natural manner.

**Theorem 2.2.** If a time-varying splitting of the matrix \( A \) is given — i.e., in equations (4)–(5), \( M_i, N_{ij} \) are replaced by \( M_i(k), N_{ij}(k) \), the \( l_{F_i} \)'s are Lipschitz constants of the time-varying \( F_i \)'s, and we redefine the quantities in (18) and (19) as

\[
    h_{ij} := \begin{cases} 
        \sup_k \left\{ \| M_i^{-1}(k) N_{ii}(k) \| + \| M_i^{-1}(k) \| l_{F_i} \right\}, & i = j, \\
        \sup_k \left\{ \| M_i^{-1}(k) N_{ij}(k) \| \right\}, & i \neq j,
    \end{cases}
\]

\[
    H := (h_{ij})
\]

— then, under Assumptions 1 to 4, the time-varying version of Theorem 2.1 holds, i.e., \( \rho(H) < 1 \) is a sufficient condition for the convergence of the time-varying asynchronous block iteration.

**Proof.** The only change in the proof is that we now use the full power of the time-varying Liapunov stability result (Theorem A1) to prove stability of the associated comparison-vector difference equation.

**Comment 5.** In the numerical-analysis literature, the word "nonstationary" is commonly used instead of "time-varying." Thus Theorem 2.2 is a generalization of Theorem 2.1 to the nonstationary case.

We now state a generalization of Theorem 2.1 in which a different relaxation factor \( \omega_i \) may be used for each processor. This is useful in a variety of situations in which different processors have different values of optimal relaxation factors.
THEOREM 2.3. If relaxation factors $\omega_i > 0, \forall i \in m$, are introduced into the asynchronous iteration (11) as

$$x_i = \sum_{j=1}^{m} M_i^{-1}N_{ij}x_j(d_i^j(k)) + M_i^{-1}F_i(x_i(d_i^i(k))),$$  \hspace{1cm} (28)

$$x_i(k+1) = x_i(d_i^i(k)) + \omega_i[x_i - x_i(d_i^i(k))],$$  \hspace{1cm} (29)

and if $\rho(H_\omega) < 1$, where $H_\omega = (h_{ij})$, $\omega = (\omega_1, \ldots, \omega_m)$, and

$$h_{ij} := \begin{cases} |1 - \omega_i| + \omega_i \left( \|M_i^{-1}I_{F_i} + \|M_i^{-1}N_{ii}\| \right), & i = j, \\ \omega_i \|M_i^{-1}N_{ij}\|, & i \neq j, \end{cases}$$  \hspace{1cm} (30)

then the relaxed asynchronous iteration defined by (28) and (29) converges.

Proof. Virtually identical to that of Theorem 2.1.

COROLLARY 2.4. If

$$\forall i \in m, \quad \omega_i = \omega,$$  \hspace{1cm} (31)

then

$$H_\omega - \omega I + |1 - \omega| I,$$  \hspace{1cm} (32)

so that $H_1 = H$ of (19). Let $\rho(H) < 1$. Since $H \geq 0$,

$$\rho(H_\omega) = \omega \rho(H) + |1 - \omega|.$$  \hspace{1cm} (33)

Thus, under the classical condition (Chazan and Miranker, 1969)

$$0 < \omega < \frac{2}{1 + \rho(H)},$$  \hspace{1cm} (34)

we can conclude that $\rho(H_\omega) < 1$, which guarantees convergence.

COMMENT 6. Using the techniques of Theorem 2.1, it is possible to define convergent asynchronous versions of the parallel multisplitting methods introduced in White (1986) for the nonlinear algebraic equation $Ax + F(x) = b$, where $A$ is an $M$-matrix, $F$ a diagonal function with continuous
nondecreasing components, and \( b \) a vector in \( \mathbb{R}^n \). It is also possible to consider overlapping block decompositions of \( A \) (in this context see Sezer and Šiljak, 1990).

As pointed out in the introduction, various sufficient conditions for the convergence of asynchronous iterations have been derived in the literature under slightly different hypotheses on the classes of allowable asynchronisms. For instance, let us assume with Miellou (1974) and El Tarazi (1982) that the delays are functions of \( j \) (sending processor) and \( k \) (time) only: in other words, for each time \( k \), the same set of delayed variables is used by all the processors which are updating. Under this slightly more restrictive hypothesis, our result becomes equivalent to Miellou's (1974) result that, if \( G \) is a contraction with respect to a vectorial norm, then a class of asynchronous iterations to solve the fixed-point equation \( u = G(u) \) converges. Miellou also observed that a contraction with respect to a vectorial norm is always a contraction in an appropriately chosen weighted maximum norm. In fact, El Tarazi (1982) proved that if \( G \) is a weighted maximum-norm contraction, then a class of asynchronous iterations to solve \( u = G(u) \) converges. In addition to the hypothesis on the delay mentioned above, the class of iterations considered by Miellou and El Tarazi is also somewhat different from ours in that a uniform upper bound on the delays need not exist; this assumption is replaced by a "no-starvation" condition for each processor.

3. EXAMPLE: ALMOST LINEAR LOAD-FLOW EQUATION

In this section we point out that the load-flow problem of electrical power networks can be formulated as an almost linear equation of the type \( Ax = F(x) \), where \( A \) is an admittance matrix, which is generally not an \( M \)-matrix, and \( F \) is a (block) diagonal nonlinear locally Lipschitz-continuous function, which is not isotonic, so that the parallel-synchronous multisplitting methods of White (1986) are not applicable. Stott (1974) observes that practical experience shows that a class of sequential iterative methods to solve the almost linear equation of the load-flow problem has slow convergence. However, for a practical example presented below, we show that an asynchronous block-iterative method of the type considered in Section 2 above has good convergence properties and has a smaller time to convergence than its synchronous counterpart (for most initial conditions).

From Kirchhoff's laws for electrical circuits, it is easy to arrive at the so-called load-flow equation for an electrical power network. This equation has the following form:

\[
Yv = I(v) + i_c =: F(v),
\]  

\( (35) \)
where $Y$ is a complex, symmetric (not Hermitian) matrix, called the nodal admittance matrix, $v$ is a complex vector of node voltages, $i_\circ$ is a complex constant, and $I(\cdot)$ is a nonlinear complex vector-valued function (current) that is diagonal in the sense of Section 2 with Lipschitzian components. More precisely, let

$$v \in \mathbb{C}^n, \quad v = (v_1, \ldots, v_m), \quad v_i \in \mathbb{C}^{n_i}, \quad n = \sum_{i=1}^m n_i.$$ 

The nonlinearity $I(\cdot)$ is of the form

$$I : \mathbb{C}^n \to \mathbb{C}^n : (v_1, \ldots, v_m) \mapsto (I_1(v_1), \ldots, I_m(v_m)).$$

$$I_k : \mathbb{C}^{n_k} \to \mathbb{C}^{n_k} : (v_k^1, \ldots, v_k^{n_k}) \mapsto \left( \frac{S_k^1}{v_k^1}, \ldots, \frac{S_k^{n_k}}{v_k^{n_k}} \right), \quad v_k^i \in \mathbb{C} \quad \forall i,$$

where $S_k^i$ is a complex constant (conjugate of power injected at node $i$) and $v_k^i$ is the complex conjugate of $v_k^i$.

Let us assume temporarily that $n_k = 1$. Then for $I_k(\cdot)$ to satisfy a Lipschitz condition in a domain $D \subset \mathbb{C}$, we must have

$$\forall z, w \in D, \quad |I_k(z) - I_k(w)| \leq L_k |z - w| \quad \text{for some } L_k \in \mathbb{R},$$

or

$$\forall z, w \in D \quad \left| \frac{S_k^*}{z^*} - \frac{S_k^*}{w^*} \right| \leq L_k |z - w|,$$

or

$$|S_k| \cdot |w^* - z^*| \leq L_k |z - w| \cdot |z^* w^*|.$$ 

Since for all $z, w$ we have $|z - w| = |w^* - z^*|$ and $|z^* w^*| = |zw|$, this means that, for $I_k(\cdot)$ to satisfy a Lipschitz condition, we must have

$$\forall z, w \in D, \quad L_k \geq \frac{|S_k|}{|w| \cdot |z|}. \quad (36)$$
For most power systems under normal conditions, the solution of (35) that is of interest lies in the neighborhood of unity (after normalization), so that it is usual to choose $D$ to be an annulus of the type $z : a \leq |z| \leq b$ and typical values are $a = 0.5$, $b = 2$. For this choice of $D$, $|z| \cdot |w| \geq a^2$ for all $z, w \in D$, so that from (36) we can conclude that

$$L_k = \frac{1}{a^2 |S_k|}$$

is a suitable choice of Lipschitz constant for $I_k(\cdot)$ in $D$.

For $n_k > 1$, an estimate $L_k'$ of the type (36) holds for each of the components of $I_k(\cdot)$, whence we may conclude that $L_k = \max_i(L_k')$ is a Lipschitz constant for $I_k(\cdot)$, for any choice of norm on $\mathbb{C}^{n_k}$. Consequently, $\max_k(L_k) = L$ is a Lipschitz constant for $F(\cdot)$ in (35) [and for $I(\cdot)$].

Finally, another important characteristic of (35) is that the matrix $Y$ is "almost diagonally dominant" in the sense that, for the great majority of its rows, the diagonal element is equal to the negative of the sum of the off-diagonal elements.

In this section, then, for a system of the type (35), where $Y$ is a $44 \times 44$ complex matrix that represents a 45-bus, 10-machine equivalent of the southern Brazilian power system, the asynchronous Jacobi iteration of Section 2 is compared with a synchronous implementation of the same method, showing that a considerable speedup is obtainable.

The parallel computer used to implement both algorithms is a machine based on four transputers using an interconnection network which is the complete graph on four nodes, i.e., every transputer is connected to all others.

An important feature of any implementation of an iterative method on a distributed-memory machine is the partitioning of the problem in a manner suited to the interconnection network of the processors of the machine. Since, for our machine, all processors are interconnected, this is not a critical issue. However, it is convenient to order the rows and columns of the matrix $Y$ so that it assumes the following "bordered block-diagonal form" (Hatcher, Brasch, and Van Ness, 1977):

$$Y = \begin{bmatrix}
M_1 & 0 & 0 & -N_{14} \\
0 & M_2 & 0 & -N_{24} \\
0 & 0 & M_3 & -N_{34} \\
-N_{14}^T & -N_{24}^T & -N_{34}^T & M_4
\end{bmatrix}$$

(38)
since the partition is obtainable by inspection of the network and permits the
use of a star connection: processors 1, 2, and 3 handling the block rows 1, 2,
and 3, respectively [in (38)], all connected to the root (or center-of-star)
processor 4. Below we give the synchronous and asynchronous implementa-
tions of the Jacobi-type block-iterative methods based on (11) and on the
partition (38) above.

In what follows, $S_i$ denotes the (given) vector of power injected at the $i$th
bus, and $S_i^*$ denotes the vector of injected power calculated at the $(k + 1)$th
step from the formula $S_i^* = V_i(k + 1)^*I_i(k + 1)$.

**Procedure for processor $i$, $i = 1, 2, 3$.**

$k \leftarrow 0$, $stop \leftarrow false$.
read $V_i(k), V_4(k), \varepsilon_1, \varepsilon_2$
repeat {
flag$_i$ $\leftarrow 0$
$\mathbf{b}_i \leftarrow N_i V_i + I_i(V_i(k)) + C_i s_i$
solve $M_i V_i(k + 1) = \mathbf{b}_i$ for $V_i(k + 1)$
if $||V_i(k + 1) - V_i(k)|| < \varepsilon_1$ and $||S_i - S_i^*|| < \varepsilon_2$ then
flag$_i$ $\leftarrow 1$
endif
send $V_i(k + 1), \text{flag}_i$, to proc. 4
receive $V_i(k + 1)$, $\text{stop}$ from proc. 4
$k \leftarrow k + 1$
} until ($stop = true$)

**Procedure for processor 4**

$k \leftarrow 0$, $stop \leftarrow false$.
read $V_4(k), i = 1, 2, 3, 4, \varepsilon_1, \varepsilon_2$
repeat {
flag$_4$ $\leftarrow 0$
$\mathbf{b}_4 \leftarrow \sum_{j=1}^{3} N_i V_j + I_4(V_4(k)) + C_4 s_i$
solve $M_4 V_4(k + 1) = \mathbf{b}_4$ for $V_4(k + 1)$
if $||V_4(k + 1) - V_4(k)|| < \varepsilon_1$ and $||S_4 - S_4^*|| < \varepsilon_2$ then
flag$_4$ $\leftarrow 1$
endif
receive $V_i(k + 1), \text{flag}_j$ from proc. $j$, $j = 1, 2, 3$
if flag$_k = 1$, $k = 1, 2, 3, 4$, then
$\text{stop} \leftarrow true$
endif
send $V_4(k + 1), \text{stop}$ to proc. $j$, $j = 1, 2, 3$
$k \leftarrow k + 1$
} until ($stop = true$)
For the asynchronous implementation the following modifications are made:

**PROCEDURE FOR PROCESSOR** \( i, i=1,2,3 \)

\[
\text{send } V_i(k+1), \text{flag}_i \text{ to buffer of proc. 4}
\]
\[
\text{receive } V_i(k+1), \text{stop} \text{ from local buffer}
\]

**PROCEDURE FOR PROCESSOR** 4

\[
\text{receive } V_j(k+1), \text{flag}_j, j=1,2,3, \text{ from local buffers}
\]
\[
\text{if flag}_k, k=1,2,3,4, \text{ and } ||YV-I||_\infty < \varepsilon_3 \text{ then}
\]
\[
\text{stop } \leftarrow \text{true}
\]
\[
\text{endif}
\]
\[
\text{send } V_i(k+1), \text{stop} \text{ to buffer of proc. } j, j=1,2,3
\]

**REMARKS.**

1. The computations of the tolerances of computed power are only performed when the voltage tolerance condition is satisfied.

2. The calculation \( ||YV-I||_\infty < \varepsilon_3 \) in the asynchronous case is only performed after the condition \( \text{flag}_k = 1, k=1,2,3,4 \), is satisfied.

These two implementations are compared in Table 1 for two different partitions (which specify the sizes of the diagonal blocks \( M_i \)), both of which are in the BBDF form (38), and for four different initial conditions \( V(0) \), called, respectively (i) "close"; (ii) "flat start"; (iii) "quasi flat start (up)," and (iv) "quasi flat start (down)." The initial conditions are so called because, respectively, (i) it is close to the final solution (this is the usual case in practice: good starting guesses are usually known); (ii) the \( v \)-profile is flat—all voltages are chosen to have a real part of unity and an imaginary part of zero; (iii) the upper half of the vector \( V(0) \) is "flat," and the lower half the same as in the "close" vector; (iv) vice versa.

Some comments on these results are in order. First, they are representatives of a large class of similar results obtained by us for this system. Second, for a given initial condition, various runs of the asynchronous implementation produce, as is to be expected, slightly differing times. Thus the values for asynchronous time to convergence in Table 1 are average values, and the
TABLE 1

PERFORMANCE OF THE TWO IMPLEMENTATIONS

<table>
<thead>
<tr>
<th>Initial condition</th>
<th>Close</th>
<th>Flat start</th>
<th>Quasiflat start (up)</th>
<th>Quasiflat start (down)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Implementation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dimensions of diagonal blocks $M_i$, $i = 1, 2, 3, 4$: 10, 12, 8, 14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Synchronous</td>
<td>1.74</td>
<td>3.54</td>
<td>5.90</td>
<td>5.91</td>
</tr>
<tr>
<td>Asynchronous</td>
<td>0.68</td>
<td>3.55</td>
<td>2.99</td>
<td>3.52</td>
</tr>
<tr>
<td><strong>Speedup</strong></td>
<td>2.56</td>
<td>1.00</td>
<td>1.97</td>
<td>1.68</td>
</tr>
<tr>
<td>Dimensions of diagonal blocks, $M_i$, $i = 1, 2, 3, 4$: 11, 11, 12, 10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Synchronous</td>
<td>2.12</td>
<td>3.57</td>
<td>6.25</td>
<td>^a</td>
</tr>
<tr>
<td>Asynchronous</td>
<td>0.71</td>
<td>6.50</td>
<td>4.59</td>
<td>5.09</td>
</tr>
<tr>
<td><strong>Speedup</strong></td>
<td>2.99</td>
<td>0.55</td>
<td>1.36</td>
<td>—</td>
</tr>
</tbody>
</table>

*Did not converge.*

Highest speedup observed was 3.34. Other general observations based on our computational experience for this problem are as follows:

(i) For a good initial condition, the asynchronous time to convergence is usually considerably smaller than the synchronous time.

(ii) In the absence of any information, i.e., for a "flat start," the synchronous implementation converges faster.

(iii) The optimal relaxation factor $\omega$ was observed to be unity.

(iv) For certain partitions and certain initial conditions, it is possible that the asynchronous implementation converges while its synchronous counterpart does not. This is probably related to the fact that the paths to and domains of convergence of the two implementations are different: this is an area in which further research is needed.

(v) In general, the partition has an effect on the performance of both algorithms, but the effect is much more pronounced on the asynchronous algorithm.

4. CONCLUDING REMARKS

As far as the theoretical results obtained above are concerned, we can say that the hybrid approach (contraction mapping and vector difference inequality) leads to results similar to those of other approaches; it allows greater
flexibility (time and space variation) in the choice of delays which model asynchronisms, but restricts them to be uniformly bounded in time.

From a practical viewpoint, the asynchronous algorithm outperforms the synchronous algorithm for most partitions and most initial conditions, thus indicating that it has potential for large-scale problems. However, as pointed out above, an adequate choice of partition is crucial to ensure good performance of the asynchronous algorithm, and this is a difficult problem. One approach to it has been suggested by Serzer and Šiljak (1989), for example. In this context, we point out that the theoretical conditions derived above, although difficult and time-consuming to verify in practice, provide some useful guidelines for the choice of a partition. For example, a partition that has a higher degree of block dominance than another generally leads to an iterative method that converges faster. Finally, it is clear that further work is needed on practical estimates of domains and rates of convergence of the asynchronous algorithm.

APPENDIX A. STABILITY RESULT FOR ASYNCHRONOUS PROCESSES

Let a Cartesian-product decomposition of \( \mathbb{R}^n \) be given as in (2), and let us assume that a single norm \( \| \cdot \| \) is used on all the factors \( \mathbb{R}^{n_i} \), \( \forall i \in m \). Assume also that we are in the context of asynchronous computation of Section 2 above, for the following fixed-point equation:

\[
x(k + 1) = x \mathcal{H}(x(k), k)x(k), \quad x \in \mathbb{R}^n, \tag{39}
\]

\[
\forall i \in m \quad \forall k \in \mathbb{N}, \quad x_i(k + 1) = \sum_{j=1}^{m} H_{ij}(x(k), k)x_j(k), \quad x_i \in \mathbb{R}^{n_i}, \tag{40}
\]

where \( \mathcal{H} = (H_{ij}(x(k), k)) \), and

\[
\forall k \in \mathbb{N}, \quad H_{ij}(x(k), k) \in \mathbb{R}^{n_i \times n_j}. \tag{41}
\]

In addition, the uniformly-bounded-delay assumption (10) is assumed to hold, and we write the asynchronous implementation of (40) as

\[
\forall i \in m, \quad x_i(k + 1) = \sum_{j=1}^{m} H_{ij}(x_i(d_i^1(k)), \ldots, x_m(d_m^i(k)), k)x_j(d_j^i(k)) \tag{42}
\]
Under the above assumptions, Kaszkurewicz, Bhaya, and Šiljak (1990) proved Theorem A1 below, using the Liapunov function

\[ V(k) := \max_{i = m, p = d} \left\{ w_i^{-1} \| x_i(k) \|, w_i^{-1} \| x_{i,p}(k) \| \right\}, \quad (43) \]

where \( w_1, \ldots, w_m \) are positive weights, \( d := \{0, 1, \ldots, d\} \), and

\[ x_{i,p}(k) := x_i(k - p). \quad (44) \]

**Theorem A1.** If, in a closed set \( D = D_1 \times \cdots \times D_m \subset \mathbb{R}^n \) we have

\[ \mathcal{H}(D) \subset D \]

and

\[ \rho(H) < 1, \quad (45) \]

where \( H = (h_{ij}) \) and

\[ h_{ij} := \sup_{j \in m, k \in \mathbb{N}, x_j(\cdot)} \left\{ \| H_{ij}(x_1(d_1^i(k)), \ldots, x_m(d_m^i(k)), k) \| \right\}, \quad (46) \]

then the zero solution of (42) is locally exponentially stable in the domain \( D \).

In the proof of Theorem A1, it is necessary to rewrite (42) in standard state-space form, and this is done by using the delayed variables in (44) and an augmented state vector as defined in (23), Section 2. To fix ideas, we now do this for the following simple case of Equation (42):

\[ x_1(k + 1) = h_{11}(k)x_1(d_1^1(k)) + h_{12}(k)x_2(d_2^1(k)). \quad (47) \]

\[ x_2(k + 1) = h_{21}(k)x_1(d_1^2(k)) + h_{22}(k)x_2(d_2^2(k)). \quad (48) \]

Let us also assume that the uniform bound \( d \) on the delays \( d_j^i(\cdot) \) is unity, i.e.

\[ \forall k \in \mathbb{N}, \quad \forall i, j \in \{1, 2\}, \quad d_j^i(k) \in \{k, k - 1\}. \quad (49) \]
Finally we define the binary switching functions

\[ \delta_1^1(k) = \begin{cases} 0 & \text{if } d_1^1(k) = k - 1, \\ 1 & \text{if } d_1^1(k) = k, \end{cases} \]  

(50)

\[ \delta_{1,1}^1(k) = \begin{cases} 1 & \text{if } d_1^1(k) = k - 1, \\ 0 & \text{if } d_1^1(k) = k, \end{cases} \]  

(51)

and similarly for \( \delta_2^2(\cdot) \), \( \delta_{1,2}^2(\cdot) \), \( \delta_2^3(\cdot) \), \( \delta_{2,1}^3(\cdot) \), \( \delta_2^4(\cdot) \), and \( \delta_{2,2}^4(\cdot) \). Then we can write (47, 48) in terms of the augmented state vector \( x_a(k) = (x_1(k), x_{1,1}(k), x_2(k), x_{2,1}(k))^T \) as

\[ x_a(k + 1) = H_a(k)x_a(k), \]  

(52)

where

\[ H_a(k) = \begin{bmatrix} h_{11}(k)\delta_1^1(k) & h_{11}(k)\delta_{1,1}^1(k) & h_{12}(k)\delta_2^1(k) & h_{12}(k)\delta_{1,2}^1(k) \\ 1 & 0 & 0 & 0 \\ h_{21}(k)\delta_2^2(k) & h_{21}(k)\delta_{2,1}^2(k) & h_{22}(k)\delta_2^3(k) & h_{22}(k)\delta_{2,2}^3(k) \\ 0 & 0 & 1 & 0 \end{bmatrix}. \]  

(53)

**Comment.** It is clear how to generalize this simple example. Note also that, for all \( k \), the elements of \( H_a(\cdot) \) are either zeros, ones, or elements \( h_{ij}(\cdot) \) of \( H_{ij}(\cdot) \).

The following lemma is also useful, as it provides several equivalents for (45).

**Lemma A2.** For a nonnegative matrix \( H = (h_{ij}) \), the following are equivalent:

1. There exists a positive diagonal \( D \) such that \( \|D^{-1}HD\|_\infty < 1 \).
2. The spectral radius \( \rho(H) \) is strictly less than unity.
3. \( I - H \) is quasidominant.
4. \( H \in \mathcal{D} := \{H : \exists \text{ positive diagonal } P \text{ such that } H^TPH - P \text{ is negative definite}\} \).
5. \( I - H \) is a nonsingular \( M \)-matrix.

**Proof.** See Kaszkurewicz, Bhaya, and Šiljak (1990).
APPENDIX B: THE COMPARISON LEMMA

We state and prove a time-varying discrete version of the well-known Bellman lemma, following the time-invariant version of Grujić and Šiljak (1973):

**Lemma B1.** Let $z(k; k_0, z_0) \in \mathbb{R}^b \forall k$ be a solution of the vector difference inequality

$$z(k + 1) \leq H(k)z(k), \quad (54)$$

where $\forall k, H(k) \in \mathbb{R}^{b \times b}$ and the initial condition is given by

$$z_0 = z(k_0; k_0, z_0). \quad (55)$$

Let $y(k; k_0, y_0)$ be a solution of the associated comparison-vector difference equation

$$y(k + 1) = H(k)y(k) \quad (56)$$

with initial condition given by

$$y_0 = y(k_0; k_0, y_0). \quad (57)$$

If

$$z_0 = y_0 \quad (58)$$

and

$$\forall k \geq k_0, \quad H(k) = (h_{ij}(k)) \geq 0, \quad (59)$$

i.e.

$$\forall k \geq k_0, \quad h_{ij}(k) \geq 0, \quad (60)$$

then

$$\forall k \geq k_0, \quad z(k; k_0, z_0) \leq y(k; k_0, z_0). \quad (61)$$
Proof. Let us make the following induction hypothesis:

$$\forall k \in \{k_0, k_0 + 1, \ldots, m\}, \quad z(k; k_0, z_0) \leq y(k; k_0, z_0).$$  \hfill (62)

Multiplying both sides of the inequality (62) by the nonnegative matrix $H(k)$ gives, by (54) and (56),

$$z(k + 1; k_0, z_0) \leq y(k + 1; k_0, z_0).$$  \hfill (63)

Comment. The above proof hinges on the observation that if $v^T := (v_1, \ldots, v_b)$ and $w^T := (w_1, \ldots, w_b)$ are any two vectors in $\mathbb{R}^b$ (not necessarily nonnegative) that satisfy

$$\forall i \in \{1, \ldots, b\}, \quad v_i \leq w_i,$$  \hfill (64)

which we denote

$$v \leq w,$$  \hfill (65)

and if $A \geq 0$ is any nonnegative matrix, then

$$Av \leq Aw.$$  \hfill (66)

REFERENCES


*Received 2 January 1990; final manuscript accepted 9 November 1990*