Electronic Transactions on Numerical Analysis.
Volume 4, pp. 1-13, March 1996.

ISSN 1068-9613.

ETNA
Kent State University
etna@mcs.kent.edu

NON-STATIONARY PARALLEL MULTISPLITTING AOR METHODS*

ROBERT FUSTER†, VIOLETA MIGALLÓN‡, AND JOSÉ PENADÉS‡

Abstract. Non-stationary parallel multisplitting iterative methods based on the AOR method are studied for the solution of nonsingular linear systems. Convergence of the synchronous and asynchronous versions of these methods is studied for H–matrices. Furthermore, computational results about these methods on both shared and distributed memory multiprocessors are discussed. The numerical examples presented cover the non-stationary parallel multisplitting Gauss-Seidel and SOR methods applied to the solution of the linear system yielded by a finite difference discretization of the two-dimensional Laplace's equation on a rectangular domain under Dirichlet boundary conditions. These results show that non-stationary AOR-type methods (synchronous and asynchronous) are better than the corresponding standard parallel multisplitting AOR method. Moreover, asynchronous versions always behave better than the synchronous ones.

Key words. non-stationary multisplitting methods, AOR method, asynchronous algorithms, H–matrices, parallel implementation, shared memory, distributed memory.

AMS subject classification. 65F10.

1. Introduction. In this paper we present non-stationary synchronous and asynchronous algorithms based on the multisplitting accelerated overrelaxation (AOR) method for the solution of large linear systems of the form

$$Ax = b, \tag{1.1}$$

where $A \in \mathbb{R}^{n\times n}$ is a nonsingular matrix and x and b are n-vectors. The multisplitting iterative method was introduced by O'Leary and White [20] and was further studied by other authors; see, e.g., Frommer and Mayer [8], [9], Neumann and Plemmons [18], White [29], [30], [31], and Szyld and Jones [25].

A set $\{F_j, G_j, E_j\}_{j=1}^r$ is called a multisplitting of A if $A = F_j - G_j$, $j = 1, 2, \ldots, r$, F_j are nonsingular matrices and E_j, called *weighting matrices*, are nonnegative diagonal matrices such that $\displaystyle\sum_{j=1}^r E_j = I$. The corresponding multisplitting method is defined as

$$x^{(l+1)} = \sum_{j=1}^r E_j T_j x^{(l)}, \quad l = 0, 1, 2, \ldots, \tag{1.2}$$

where $x^{(0)}$ is an arbitrary initial vector, and $T_j : \mathbb{R}^n \longrightarrow \mathbb{R}^n$, $1 \le j \le r$ are r operators defined as

$$T_j x = F_j^{-1} G_j x + F_j^{-1} b, \quad j = 1, 2, \ldots, r.$$

Since the calculations represented by each operator T_j are independent of each other, method (1.2) can be executed in parallel with an appropriated choice of the matrices

* Received September 6, 1995 . Accepted for publication March 19, 1996. Communicated by A. Ruttan

† Departament de Matemática Aplicada, Universitat Politècnica de València, E-46071 València, Spain (rfuster@mat.upv.es). This research was supported by Spanish CICYT grant number TIC91-1157-C03-01.

‡ Departamento de Tecnología Informática y Computación, Universidad de Alicante, E-03071 Alicante, Spain (violeta@dtic.ua.es, jpenades@dtic.ua.es) . This research was partially supported by both Spanish CICYT grant number TIC91-1157-C03-01 and the RECITE project number 94002.

ETNA
Kent State University
etna@mcs.kent.edu

E_j and F_j. On the other hand, a component of $T_j x^{(l)}$ does not need to be computed if the corresponding diagonal entry of E_j is zero.

As can be seen, in the multisplitting method (1.2) each local approximation at the lth global iteration is updated solving the linear system $F_j x^{(l+1)} = G_j x^{(l)} + b$. However, it is possible to update the local approximation more than once, using different local iterates on the right hand side, computed earlier. In this case we have a non-stationary multisplitting method which corresponds to the following algorithm.

ALGORITHM 1. (NON-STATIONARY MULTISPLITTING).

Given the initial vector $x^{(0)}$

$$\begin{aligned}
&\text{For } l = 0, 1, \ldots, \text{ until convergence}\\
&\quad\text{In processor } j,\ j = 1 \text{ to } r\\
&\qquad y_j^{(0)} = x^{(l)}\\
&\qquad\text{For } k = 1 \text{ to } q(l,j)\\
&\qquad\qquad F_j y_j^{(k)} = G_j y_j^{(k-1)} + b \qquad (1.3)\\
&\quad x^{(l+1)} = \sum_{j=1}^{r} E_j y_j^{(q(l,j))}.
\end{aligned}$$

Note that the number of times that each processor j solves the system defined by its operator T_j is defined by $q(l,j)$ which depends on the iteration l and the processor j.

Bru, Elsner and Neumann [2] studied two non-stationary methods (synchronous and asynchronous) based on the multisplitting method. They used the term *chaotic* for their non-stationary methods; however we prefer the term non-stationary, because in the previous literature chaotic is synonymous with asynchronous (see [7]). They give sufficient conditions which assure the convergence of both methods when A is a monotone matrix, i.e., $A^{-1} \geq O$. Later, Mas, Migallón, Penadés and Szyld [14] showed the convergence of these non-stationary methods in the context of H–matrices. Further, a background on non-stationary methods can also found in [4], [5], [6], [11], [15], and [24].

In the context of relaxed parallel multisplitting algorithms, Wang [28] presented a class of algorithms called parallel multisplitting AOR methods which are a generalization of the method given in [8]. Convergence of these methods was established for H–matrices. Let $\{D - L_j, V_j, E_j\}_{j=1}^{r}$ be a multisplitting of A where $D = \operatorname{diag}(A)$, and L_j are strictly lower triangular matrices. Note that the matrices V_j are not generally upper triangular. Each single splitting induces an iterative method, described by the operator $P_j : \mathbb{R}^n \longrightarrow \mathbb{R}^n,\ j = 1, 2, \ldots, r$, defined as

$$P_j x = (D - \mu L_j)^{-1} \left\{ \left[(1-\omega) D + (\omega - \mu) L_j + \omega V_j \right] x + \omega b \right\}, \tag{1.4}$$

where $\omega, \mu \in \mathbb{R}$ and $\omega \neq 0$. Then, given an initial vector $x^{(0)}$, the parallel multisplitting AOR method [28] produces the sequence of vectors

$$x^{(l+1)} = \sum_{j=1}^{r} E_j P_j x^{(l)}, \quad l = 0, 1, 2, \ldots. \tag{1.5}$$

Clearly, if $r = 1$ and $-L_1, -V_1$ are the strictly lower and strictly upper triangular parts of A respectively, then the scheme (1.5) reduces to the well-known AOR method [12]. Also, it is easy to observe that the scheme (1.5) includes some classic parallel

ETNA
Kent State University
etna@mcs.kent.edu

multisplitting methods, such as the ones based on the Jacobi, Gauss-Seidel, JOR and SOR multisplitting methods; see, e.g., [28].

In this paper we study the convergence of the non-stationary parallel multisplitting AOR method and its extension to an asynchronous algorithm, where each processor does not need to wait for the completion of the computation of the iterates in other processors; see Section 3. First, in the next section we present some notation, definitions and results which we refer to later. The last section contains some numerical experiments which illustrate the performance of the algorithms studied. This performance is achieved both in shared memory and distributed memory multiprocessors.

2. Preliminaries. Given a vector $x \in \mathbb{R}^n$, we say that it is nonnegative (positive), denoted $x \geq 0$ ($x > 0$), if all components of x are nonnegative (positive). Similarly, if $x, y \in \mathbb{R}^n$, $x \geq y$ ($x > y$) means that $x - y \geq 0$ ($x - y > 0$). For a vector $x \in \mathbb{R}^n$, $|x|$ denotes the vector whose components are the absolute values of the corresponding components of x. These definitions carry over immediately to matrices.

Let $x > 0$, we consider the vector norm

$$\|y\|_x = \inf\{\beta > 0 \ : \ |y| \leq \beta x\}. \tag{2.1}$$

This vector norm is monotonic and for every matrix $B \in \mathbb{R}^{n\times n}$ it satisfies $\|\,|B|x\|_x = \|B\|_x$, where $\|B\|_x$ denotes the matrix norm of B induced by the vector norm defined in (2.1).

By $Z^{n\times n}$ we denote the set of all real $n \times n$ matrices which have all nonpositive off-diagonal entries. A matrix $A \in Z^{n\times n}$ is said to be an M–matrix if A can be expressed in the form

$$A = sI - B, \text{ with } B \geq O \tag{2.2}$$

and $\rho(B) \leq s$, where $\rho(B)$ denotes the spectral radius of B. In particular, a nonsingular M–matrix is those of the form (2.2) for which $\rho(B) < s$. We recall that a matrix $A \in Z^{n\times n}$ is a nonsingular M–matrix if and only if A is a monotone matrix; see, e.g., Berman and Plemmons [1] or Varga [26].

For any matrix $A = (a_{ij}) \in \mathbb{R}^{n\times n}$, we define its comparison matrix $\langle A\rangle = (\alpha_{ij})$ by $\alpha_{ii} = |a_{ii}|, \quad \alpha_{ij} = -|a_{ij}|,\ i \neq j$. Following Ostrowski [22], [23], A is said to be an H–matrix if $\langle A\rangle$ is a nonsingular M–matrix. We remark that in this notation if A is an H-matrix, A is nonsingular (see Ostrowski [22] and Neumaier [16]). Of course, nonsingular M–matrices are special cases of H–matrices. In addition, as was already noted in [23], important classes of matrices such as strictly or irreducibly diagonally dominant matrices are H–matrices. See also [27] and the references given therein for equivalent conditions of H–matrices. H–matrices arise in many applications and were studied by a number of authors in connection with iterative solutions of linear systems; see Berman and Plemmons [1], Varga [26], Young [32] or Frommer and Szyld [10] for an extensive bibliography and for an example that shows that H–matrices need not be monotone.

Lemma 2.1. *If $A = D - B$ is an H–matrix, with $D = \operatorname{diag}(A)$, then $|D|$ is nonsingular and $\rho(|D|^{-1}|B|) < 1$.*

Proof. It is essentially the proof of the first part of Theorem 3.10 of [26]. It can also be found in [8]. □

LEMMA 2.2. *Let $A, B \in \mathbb{R}^{n\times n}$.*
(a) If A is an H–matrix, then $|A^{-1}| \le \langle A\rangle^{-1}$.
(b) If $|A| \le B$ then $\rho(A) \le \rho(B)$.
Proof. Part (a) follows by Ostrowski [22]; see also, e.g., Neumaier [16], [17]. Part (b) can be found, e.g., in [21, Theorem 2.4.9] and [26, Theorem 2.8]. ☐

THEOREM 2.3. *Let $A \in Z^{n\times n}$ which has all positive diagonal entries. A is a nonsingular M–matrix if and only if $\rho(B) < 1$, where $B = D^{-1}C$, $D = \mathrm{diag}(A)$, $A = D - C$.*
Proof. The proof of this theorem can be found, e.g., in Varga [26, Theorem 3.10] and Young [32, Theorem 7.2]. ☐

3. Non-stationary methods. Let $\{D-L_j, V_j, E_j\}_{j=1}^r$ be a multisplitting of A satisfying $D = \mathrm{diag}(A)$ and L_j are strictly lower triangular matrices. Suppose that we are interested in solving the nonsingular linear system (1.1) in a multiprocessor computer using the multisplitting AOR method (1.5). In order to get a good performance of all processors and a good load balance among processors, we assume that the task assigned to each processor j can vary in each iteration l, i.e., we suppose that the processor j applies $q(l,j)$ times the operator P_j defined in (1.4) in the lth iteration. Then, given an initial vector $x^{(0)}$ we construct the sequence of vectors

$$x^{(l+1)} = \sum_{j=1}^{r} E_j P_j^{q(l,j)} x^{(l)}, \quad l = 0,1,2,\ldots. \tag{3.1}$$

The numbers $q(l,j)$ are called *non-stationary parameters*, and then we call this iterative scheme, *non-stationary multisplitting AOR method*. Scheme (3.1) corresponds to the following algorithm.

ALGORITHM 2. (NON-STATIONARY MULTISPLITTING AOR).
Given the initial vector $x^{(0)}$

For $l = 0,1,\ldots,$ until convergence
 In processor j, $j = 1$ to r
  $y_j^{(0)} = x^{(l)}$
  For $k = 1$ to $q(l,j)$

$$(D-\mu L_j)y_j^{(k)} = [(1-\omega)D + (\omega-\mu)L_j + \omega V_j]\, y_j^{(k-1)} + \omega b \tag{3.2}$$

$$x^{(l+1)} = \sum_{j=1}^{r} E_j y_j^{(q(l,j))}. \tag{3.3}$$

We remark that the calculations (3.2) correspond to the following scheme.

If $\mu \ne 0$ then

$$(D-\mu L_j)\overline{y}_j^{(k)} = [(1-\mu)D + \mu V_j]\, y_j^{(k-1)} + \mu b$$
$$y_j^{(k)} = \tfrac{\omega}{\mu}\overline{y}_j^{(k)} + \left(1-\tfrac{\omega}{\mu}\right) y_j^{(k-1)}.$$

If $\mu = 0$ then

$$y_j^{(k)} = \left[(1-\omega)I + \omega D^{-1}(L_j+V_j)\right] y_j^{(k-1)} + \omega D^{-1} b.$$

Clearly, when $q(l,j) = 1$, Algorithm 2 reduces to the multisplitting AOR method studied by Wang [28].

Algorithm 2 is synchronous in the sense that step (3.3) is performed only when all vectors $y_j^{(q(l,j))}$, $j = 1, 2, \ldots, r$, are calculated. Alternatively, each part of $x^{(l+1)}$, i.e., $E_j y_j^{(q(l,j))}$, can be updated as soon as the vector $y_j^{(q(l,j))}$ is calculated, without waiting for the other parts of $x^{(l+1)}$ to be updated. To construct an asynchronous version of Algorithm 2 we consider a different scheme where all processors are always working without waiting for information from the other processors.

Let $\{j_l\}_{l=0}^{\infty}$, $1 \le j_l \le r$, be a sequence of integers that indicates the processor which updates the approximation to the solution at the lth iteration. Let $r_l - 1$ be the number of times that processors other than the j_lth processor update the approximation of the solution during the time interval in which the j_lth processor's calculations are performed. This implies that r_l is the smallest positive integer such that $j_l = j_{l+r_l}$. As is customary in the description of asynchronous algorithms (see, e.g., [2]) we assume that the sequence of integers $\{j_l\}_{l=0}^{\infty}$, $1 \le j_l \le r$, is a *regulated sequence*. This means that there exists a positive integer m such that each of the integers $1, 2, \ldots, r$ appears at least once in every m consecutive elements of the sequence. With this notation we consider the following asynchronous scheme:

$$x^{(l+r_l)} = (I - E_{j_l}) x^{(l+r_l-1)} + E_{j_l} P_{j_l}^{q(l,j_l)} x^{(l)}, \quad l = 0, 1, 2, \ldots, \tag{3.4}$$

which corresponds to the following algorithm.

Algorithm 3. (Asynchronous Non-stationary Multisplitting AOR).
Given the initial vector $x^{(0)}$

In processor j_l, $l = 0, 1, \ldots$, until convergence
$y_{j_l}^{(0)} = x^{(l)}$
For $k = 1$ to $q(l, j_l)$
$$(D - \mu L_{j_l}) y_{j_l}^{(k)} = \left[(1-\omega)D + (\omega - \mu) L_{j_l} + \omega V_{j_l}\right] y_{j_l}^{(k-1)} + \omega b$$
$$x^{(l+r_l)} = (I - E_{j_l}) x^{(l+r_l-1)} + E_{j_l} y_{j_l}^{(q(l,j_l))}.$$

Note that when the weighting matrices form a partition of the identity, i.e., when the following relations hold

$$E_i E_j = \begin{cases} E_i, & \text{if } i = j, \\ O, & \text{if } i \ne j, \end{cases} \tag{3.5}$$

formulation (3.4) can be rewritten as

$$x^{(l+r_l)} = (I - E_{j_l}) x^{(l+r_l-1)} + E_{j_l} \sum_{j=1}^{r} E_j P_j^{q(l,j)} x^{(l)}. \tag{3.6}$$

However, when relations (3.5) do not hold, schemes (3.4) and (3.6) represent different asynchronous algorithms. In formulation (3.4) all processors are always working without synchronization with other processors, while iteration (3.6) implies a certain degree of synchronization between groups of processors.

4. Convergence.

In this section we show the convergence of the synchronous scheme (3.1) (or Algorithm 2), and asynchronous schemes (3.4) (or Algorithm 3) and (3.6).

The iteration scheme (3.1) can be written as

$$x^{(l+1)} = H^{(l)} x^{(l)} + c^{(l)}, \quad l = 0, 1, 2, \ldots, \tag{4.1}$$

where $H^{(l)}$ are the iteration matrices

$$H^{(l)} = \sum_{j=1}^{r} E_j J_j^{q(l,j)}, \quad l = 0, 1, 2, \ldots, \tag{4.2}$$

with

$$J_j = (D - \mu L_j)^{-1} \left[(1-\omega) D + (\omega - \mu) L_j + \omega V_j \right], \quad j = 1, 2, \ldots, r,$$

and

$$c^{(l)} = \sum_{j=1}^{r} E_j \left(\sum_{i=0}^{q(l,j)-1} J_j^i \right) \omega (D - \mu L_j)^{-1} b, \quad l = 0, 1, 2, \ldots.$$

Let ξ be the exact solution of (1.1) and let $\epsilon^{(l+1)} = x^{(l+1)} - \xi$ be the error at $l+1$ iteration. It is easy to prove that ξ is a fixed point of (4.1). Thus,

$$\epsilon^{(l+1)} = H^{(l)} \epsilon^{(l)} = \cdots = H^{(l)} H^{(l-1)} \cdots H^{(0)} \epsilon^{(0)}, \quad l = 0, 1, 2, \ldots.$$

So, the sequence of error vectors $\{\epsilon^{(l)}\}_{l=0}^{\infty}$ generated by iteration (3.1) converges to the vector 0, if and only if $\lim_{l \to \infty} H^{(l)} H^{(l-1)} \cdots H^{(0)} = O$. Hence, by Lemma 3 of [3], it suffices to prove that $\|H^{(l)}\| \leq \alpha$, $l = 0, 1, 2, \ldots$, for some $\alpha < 1$ and some matrix norm $\|\cdot\|$.

The following theorem shows the convergence of scheme (3.1), or Algorithm 2, when A is an H–matrix and $0 \leq \mu \leq \omega < \frac{2}{1+\rho(|D|^{-1}|B|)}$, with $\omega \neq 0$, where $D = \operatorname{diag}(A)$ and $A = D - B$. Clearly from Lemma 2.1, $|D|$ is a nonsingular matrix and $\rho(|D|^{-1}|B|) < 1$.

THEOREM 4.1. *Let $A = D - L_j - V_j = D - B$, $j = 1, 2, \ldots, r$ be an H–matrix, where $D = \operatorname{diag}(A)$ and L_j are strictly lower triangular matrices. Assume that $|B| = |L_j| + |V_j|$, $j = 1, 2, \ldots, r$. If $0 \leq \mu \leq \omega < \frac{2}{1+\rho}$, with $\omega \neq 0$, where $\rho = \rho(|D|^{-1}|B|)$, and $q(l,j) \geq 1$, $l = 0, 1, 2, \ldots$, $j = 1, 2, \ldots, r$. Then the iteration (3.1) converges to the solution of the linear system (1.1) for all initial vector $x^{(0)} \in \mathbb{R}^n$.*

Proof. By Theorem 2.3 and part (b) of Lemma 2.2, if $0 \leq \mu < \frac{2}{1+\rho}$, the matrices $\langle D - \mu L_j \rangle$ are H–matrices. Then, using part (a) of Lemma 2.2, some manipulations in (4.2) yield

$$|H^{(l)}| \leq \sum_{j=1}^{r} E_j \left[M_j(\mu)^{-1} \left| (1-\omega) D + (\omega - \mu) L_j + \omega V_j \right| \right]^{q(l,j)}, \tag{4.3}$$

with

$$M_j(\mu) = |D| - \mu |L_j| = \langle D - \mu L_j \rangle, \quad j = 1, 2, \ldots, r. \tag{4.4}$$

Consider two cases.

(i) $0 \leq \mu \leq \omega \leq 1$, with $\omega \neq 0$. Denoting for $j = 1, 2, \ldots, r$

$$N_j^1(\mu, \omega) = (1-\omega)|D| + (\omega - \mu)|L_j| + \omega |V_j| \geq O,$$

we obtain

$$(4.5)\qquad |H^{(l)}| \le \sum_{j=1}^{r} E_j \left(M_j(\mu)^{-1}N_j^1(\mu,\omega)\right)^{q(l,j)}, \quad l = 0,1,2,\ldots.$$

On the other hand, it is easy to obtain

$$(4.6)\qquad M_j(\mu) - N_j^1(\mu,\omega) = \omega\langle A\rangle, \quad j = 1,2,\ldots,r.$$

Consider any fixed positive vector e (e.g., with all components equal to 1), and $x = \omega^{-1}\langle A\rangle^{-1}e$. Since $\omega^{-1}\langle A\rangle^{-1} \ge O$ and no row of $\omega^{-1}\langle A\rangle^{-1}$ can have all null entries, we get $x > 0$. By the same arguments $M_j(\mu)^{-1}e > 0,\ j = 1,2,\ldots,r$. We have from (4.6) that

$$x - M_j(\mu)^{-1}N_j^1(\mu,\omega)x = M_j(\mu)^{-1}\omega\langle A\rangle x = M_j(\mu)^{-1}e > 0.$$

Then $M_j(\mu)^{-1}N_j^1(\mu,\omega)x < x$, and thus there exists a real constant $0 \le \alpha_1 < 1$, such that for all $j = 1,2,\ldots,r$, it satisfies

$$(4.7)\qquad M_j(\mu)^{-1}N_j^1(\mu,\omega) \le \alpha_1 x < x.$$

From (4.5) and (4.7) it follows

$$|H^{(l)}|x \le \sum_{j=1}^{r} E_j \left(M_j(\mu)^{-1}N_j^1(\mu,\omega)\right)^{q(l,j)} x \le \alpha_1 x, \quad l = 0,1,2,\ldots,$$

with $0 \le \alpha_1 < 1$ and $x > 0$. Then, using the matrix norm induced by the vector norm (2.1), one obtains $\|H^{(l)}\|_x \le \alpha_1 < 1,\ l = 0,1,2\ldots.$
(ii) $0 \le \mu \le \omega,\quad 1 < \omega < \frac{2}{1+\rho}$. Letting

$$N_j^2(\mu,\omega) = (\omega-1)|D| + (\omega-\mu)|L_j| + \omega|V_j| \ge O,$$

by the definition of $M_j(\mu)$ in (4.4) we have from (4.3) that

$$|H^{(l)}| \le \sum_{j=1}^{r} E_j \left(M_j(\mu)^{-1}N_j^2(\mu,\omega)\right)^{q(l,j)}, \quad l = 0,1,2,\ldots.$$

On the other hand, we have

$$M_j(\mu) - N_j^2(\mu,\omega) = (2-\omega)|D| - \omega(|L_j| + |V_j|) = (2-\omega)|D| - \omega|B|, \quad j = 1,2,\ldots,r.$$

Obviously, the matrix $(2-\omega)|D| - \omega|B| \in Z^{n\times n}$ with all positive diagonal entries. Furthermore, as $1 < \omega < \frac{2}{1+\rho}$, $\rho((2-\omega)^{-1}|D|^{-1}\omega|B|) < 1$, and then from Theorem 2.3 this implies that $(2-\omega)|D| - \omega|B|$ is a nonsingular M–matrix.
Consider the vector $y = \left((2-\omega)|D| - \omega|B|\right)^{-1} e$, with $e > 0$ (e.g., $e = (1,1,\ldots,1)^t$ as in the previous case). Reasoning as in case (i), there exists $0 \le \alpha_2 < 1$ such that $M_j(\mu)^{-1}N_j^2(\mu,\omega)y \le \alpha_2 y < y$. Then, $\|H^{(l)}\|_y \le \alpha_2 < 1,\quad l = 0,1,2,\ldots.$ □

Now we study the convergence of asynchronous scheme (3.6) under similar hypotheses as those for Theorem 4.1.

THEOREM 4.2. *Let $A = D - L_j - V_j = D - B,\ j = 1,2,\ldots,r,$ be an $H-$matrix, where $D = \mathrm{diag}(A)$ and L_j are strictly lower triangular matrices. Assume*

ETNA
Kent State University
etna@mcs.kent.edu

that $|B| = |L_j| + |V_j|,\ j = 1, 2, \ldots, r$. *Given a regulated sequence of integers* $\{j_l\}_{l=0}^{\infty}$, *if* $0 \le \mu \le \omega < \frac{2}{1+\rho}$, *with* $\omega \ne 0$, *where* $\rho = \rho(|D|^{-1}|B|)$, *and* $q(l, j) \ge 1,\ l = 0, 1, 2, \ldots,\ j = 1, 2, \ldots, r$, *the iteration (3.6) converges to the solution of the linear system (1.1) for all initial vector* $x^{(0)} \in \mathbb{R}^n$.

Proof. In Theorem 4.1 we have shown that for $0 \le \mu \le \omega < \frac{2}{1+\rho}$, with $\omega \ne 0$, there exists a real constant $0 \le \alpha < 1$ and a positive vector z such that $|H^{(l)}|z \le \alpha z$. Thus, the proof follows in the same way as the proof of [14, Theorem 3.2] which in turn is based on [2, Theorem 2.2]. □

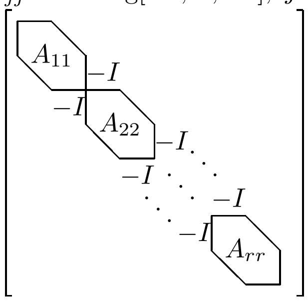
Remarks:

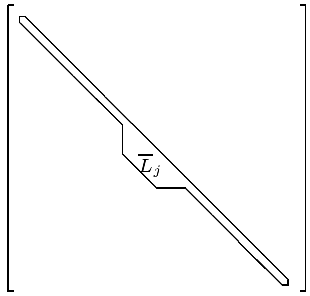
1. The convergence proof of iteration (3.4) or equivalently Algorithm 3 is similar to the one used in Theorem 4.2 using that $|J_j|z \le \alpha z$ for some positive vector z and a real constant $0 \le \alpha < 1,\ j = 1, 2, \ldots, r$.

2. When $\omega = \mu$ and $q(l, j) = 1$, iteration (3.1) coincides with [8, Method (4)]. Thus, Theorems 4.1 and 4.2 also show the convergence of the non-stationary synchronous and asynchronous versions of [8, Method (4)], under similar hypotheses as in [8, Theorem 4.2]

5. Numerical Experiments. In this section we implemented non-stationary synchronous and asynchronous Algorithms 2 and 3 respectively, described in the previous sections. We considered the problem of Laplace's equation, $\nabla^2 u = u_{ss} + u_{tt} = 0$, satisfying Dirichlet boundary conditions on the rectangle $\Omega = [0, a] \times [0, b]$, $u(s, 0) = u(0, t) = u(s, b) = 0,\ u(a, t) = 100,\ 0 \le s \le a,\ 0 \le t \le b$. We discretized, using five point finite differences, the domain Ω with $J \times K$ points equally spaced by h. This discretization yields a linear system $Ax = b$, where A is block tridiagonal, $A = \text{tridiag}[-I, C, -I]$, where I and C are $K \times K$ matrices, I is the identity, and $C = \text{tridiag}[-1, 4, -1]$. Note that A has $J \times J$ blocks of size $K \times K$. Clearly, A is a nonsingular M–matrix, and is therefore also an H–matrix.

In order to define the splittings $A = D - L_j - V_j,\ j = 1, 2, \ldots, r$, with $D = \text{diag}(A)$, we denote $A_{jj} = \text{tridiag}[-I, C, -I],\ j = 1, 2, \ldots, r$. Then we write A and $D - L_j$ as

$A =$ 

, $D - L_j =$ 

,

where each matrix $\overline{L}_j,\ j = 1, 2, \ldots, r$, is the lower triangular part of A_{jj}. We point out that matrices $V_j,\ j = 1, 2, \ldots, r$, are not upper triangular matrices. The matrices $E_j,\ j = 1, 2, \ldots, r$, are partitioned according to the structure of A with the jth block equal to the identity and the remainder blocks are zero. It is easy to prove that $|L_j| + |V_j| = |D - A|$, hence Theorems 4.1 and 4.2 assure the convergence for Algorithms 2 and 3, respectively.

We are especially interested in analyzing the behavior of the non-stationary multisplitting iterative AOR-type methods as compared to the stationary one. On the other hand, as Niethammer [19] noted, the question of determining the optimal relaxation parameters is of more theoretical than of practical interest. Particularly, in many applications, such as the model problem of the Laplace's equation on a rectangle, only a very small increase of the rate of convergence can be expected when one uses the best AOR method instead of the best SOR method; see also Hadjidimos [13]. We ran several examples of the non-stationary multisplitting AOR methods and similar conclusions were obtained. Therefore, only numerical experiments based on the Gauss-Seidel and SOR multisplitting methods are reported here.

Experiments were performed, with matrices of different orders, on a shared memory multiprocessor Alliant FX/80 with 8 processors, and on a distributed memory multiprocessor IBM SP2 using 8 processors. Thus, in order to fully utilize the 8 processors, we let $r = 8$. The stopping criterion used was

$$\sum_{i=1}^{n} |x_i^{(l)} - x_i^{(l-1)}| < 5 \cdot 10^{-10}$$

and the initial vector was $x^{(0)} = (1, 1, \ldots, 1)^t$.

The conclusions are similar on both multiprocessors. So, Table 5.1 shows the behavior of some synchronous non-stationary models on these multiprocessors, when $\omega = \mu = 1$ and the matrix A is of size 50000. Matrix A has $J = 500$ diagonal blocks C of size $K = 100$. The size of A_{jj} is 5000 for $j = 1, 2, \ldots, 6$ and 10000 for the rest. The values of a, b of the domain Ω such that the discretization yields this coefficient matrix are shown in Figure 5.1.a. We use the notation $6^6 3^2$ to represent the non-stationary parameters $q(l, j) = 6$ for $j = 1, 2, \ldots, 6$ and $q(l, j) = 3$ for $j = 7, 8$, for all $l = 0, 1, 2, \ldots$, i.e., to represent that the first six processors update their part of the vector six times and each of the last two processors updates its part of the vector three times. Similar notation is used for other values of the non-stationary parameters. In this paper all times are reported in seconds. The speed-up is calculated as $\frac{\text{CPU time of sequential algorithm}}{\text{Time of parallel algorithm}}$. We note that in shared memory we use the CPU time for the parallel algorithm, while in distributed memory we use the REAL time. This is due to the fact that there is communication among processors in the latter case. Processors achieve about 55 % – 89 % of efficiency for each case.

It can also be observed that the number of global iterations of the non-stationary algorithms is less when the parameters $q(l, j)$ are increased. Furthermore, if the decrease in the number of global iterations balances the realization of more inner updates then less execution time is observed. On the other hand, when $q(l, j) = 1$, $\omega = \mu = 1$, Algorithm 2 reduces to the well-known parallel multisplitting Gauss-Seidel method. Then it is interesting to compare the sequential results of Table 5.1 with the classic Gauss-Seidel algorithm whose results are in Table 5.2. It is observed that except for some values of $q(l, j)$ the sequential non-stationary multisplitting method is better than the Gauss-Seidel method. Obviously, this situation is independent of the computer used.

Now, we report results of non-stationary methods with $\omega = \mu \neq 1$ (non-stationary parallel multisplitting SOR method), for two matrices of size 4096 and 5632. The matrix of size 4096 has 64 blocks C of size 64, the size of A_{jj} is 1024 for $j = 1, 2$, 512 for $j = 3, 4$ and 256 for the rest. The matrix of size 5632 has 11 blocks C of size 512, the matrices A_{jj} are of size 1024 for $j = 1, 2$ and 512 for the rest. Note that the bandwidth of the matrix of size 4096 is relatively small compared to its size. This situation does not hold for the matrix of size 5632. Figures 5.1.b, 5.1.c show the domains Ω whose discretization yields this matrices of size 4096 and 5632 respectively.

We considered different synchronous non-stationary iterations depending on the parameters $q(l, j)$, and for each method we recorded the CPU time in seconds on the Alliant FX/80 as a function of different parameters (ω, μ), $\omega = \mu$. Figures 5.2 and 5.3

ETNA
Kent State University
etna@mcs.kent.edu

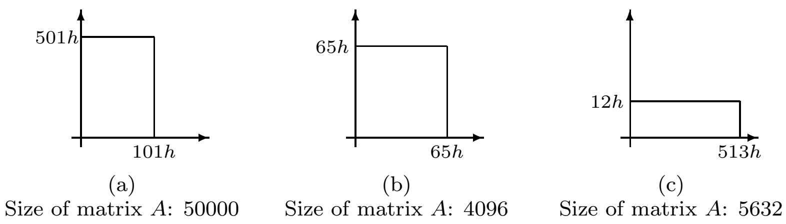


FIG. 5.1. *Domains $\Omega = [0, a] \times [0, b]$ yielding the linear systems of the numerical experiments, where h is the width of the mesh.*

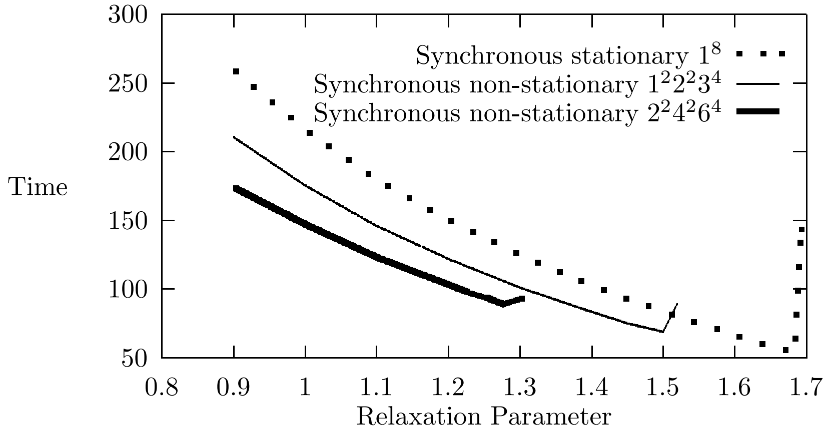


FIG. 5.2. *Synchronous algorithms, size of A:* 4096.

display the results for the matrices described above. Differences depending upon the bandwidth of A can be observed. In Figure 5.3 we see that there are cases where the larger the non-stationary parameters the longer the execution time. This is due to the fact that the larger the parameters the larger the subsystem that each processor solves in each local iteration. However, a non-stationary synchronous model better than the other ones is always obtained. Figure 5.2 shows that a relative small bandwidth assures better performance when we increase the non-stationary parameters. Results were similar for all tested matrices and both multiprocessors.

On the other hand, in those synchronous models we can obtain a load balance among processors by a good choice of the non-stationary parameters. But this is not easy to achieve. For that purpose we can use asynchronous algorithms, where the work among processor is balanced dynamically. The experiments performed showed that the asynchronous non-stationary algorithm behaves better than the synchronous non-stationary one. Also, we observed that an asynchronous algorithm setting $q(l, j) = 1$ accelerates any synchronous non-stationary algorithm. Furthermore, similar conclusions about the bandwidth of A are obtained.

In Figure 5.4, the CPU times on the Alliant FX/80 multiprocessor, of some asynchronous models are presented for the matrix of size 4096 described above. We note that when the bandwidth of A is small, we can choose non-stationary parameters greater than one, which have a better performance. If the bandwidth is not small, generally $q(l, j) = 1$ gives the best asynchronous scheme.

Finally, we note that, the choice of optimal sequences $q(l, j)$ (or simply good

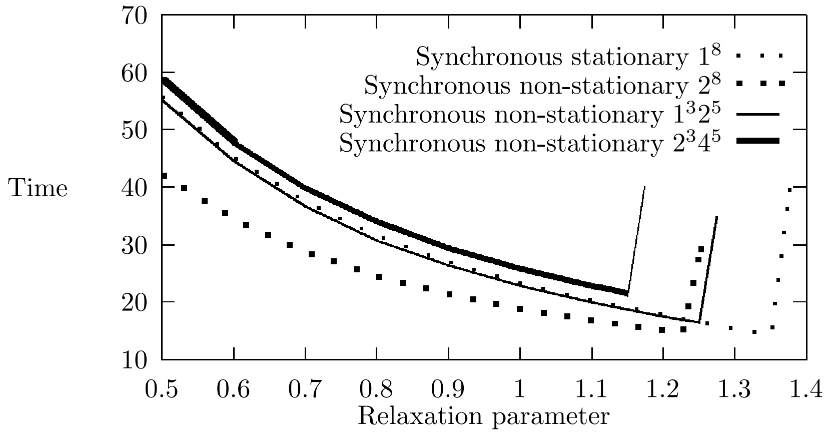


FIG. 5.3. *Synchronous algorithms, size of A:* 5632.

ones) is problem dependent, and not very well understood. In our experience, it often suffices to choose non-stationary parameters a little greater than one to obtain good overall convergence results. We also note that non-stationary algorithms allow the use of different local criteria to counterweight dynamically the work in each processor, producing a good load balance and faster convergence.

Acknowledgements. The authors would like to thank the referee, whose questions and recommendations led to improvements in the paper, and to Daniel Szyld for reading and commenting on this paper.

REFERENCES

[1] Abraham Berman and Robert J. Plemmons, *Nonnegative Matrices in the Mathematical Sciences*, Third ed., Academic Press, New York, 1979.
[2] Rafael Bru, Ludwing Elsner, and Michael Neumann, *Models of parallel chaotic iteration methods*, Linear Algebra Appl., 103 (1998), pp. 175-192.
[3] Rafael Bru and Robert Fuster, *Parallel chaotic extrapolated Jacobi method*, Appl. Math. Lett., 3 (1990), pp. 65-69.
[4] Rafael Bru, Violeta Migallón, and José Penadés, *Chaotic inner–outer iterative schemes*, In Proceedings of the Fifth SIAM Conference on Applied Linear Algebra, J.G. Lewis, ed., SIAM Press, Philadelphia, 1994, pp. 434-438.
[5] ———, *Chaotic methods for the parallel solution of linear systems*, Computing Systems in Engineering, 6(4,5) (1995), pp. 385-390.
[6] Rafael Bru, Violeta Migallón, José Penadés, and Daniel B. Szyld, *Parallel, synchronous and asynchronous two–stage multisplitting methods*, Electron. Trans. Numer. Anal., 3 (1995), pp. 24-38.
[7] D. Chazan and W. Miranker, *Chaotic relaxation*, Linear Algebra Appl., 2 (1969), pp. 199-222.
[8] Andreas Frommer and Günter Mayer, *Convergence of relaxed parallel multisplitting methods,* Linear Algebra Appl., 119 (1989), pp. 141-152.
[9] Andreas Frommer and Günter Mayer, *Parallel interval multisplittings*, Numer. Math., 56 (1989), pp. 255-267.
[10] Andreas Frommer and Daniel B. Szyld, *H-splittings and two-stage iterative methods.* Numer. Math., 63 (1992), pp. 345-356.
[11] Robert Fuster, Violeta Migallón, and José Penadés, *Parallel chaotic extrapolated Jacobi–like methods*, Linear Algebra Appl., to appear.

[12] Apostolos Hadjidimos, *Accelerated overrelaxation method*, Math. Comp., 32 (1978), pp. 149-157.
[13] Apostolos Hadjidimos, *A survey of the iterative methods for the solution of linear systems by extrapolation*, J. Comput. Appl. Math., pp. 20 (1987), pp. 37-51.
[14] José Mas, Violeta Migallón, José Penadés, and Daniel B. Szyld, Non-stationary parallel relaxed multisplitting methods, Linear Algebra Appl., to appear, 237/238 (April 1996).
[15] Violeta Migallón. *Modelos Iterativos Caóticos Síncronos y Asíncronos para la Resolución de Sistemas Lineales*. PhD thesis, Departamento de Tecnología Informática y Computación, Universidad de Alicante, November 1993. In Spanish.
[16] Arnold Neumaier, *The extremal case of some matrix inequalities*, Arch. Math., 43 (1984), pp. 137-141.
[17] ———, *New techniques for the analysis of linear interval equations*, Linear Algebra Appl., 58 (1984), pp. 273-325.
[18] Michael Neumann and Robert J. Plemmons, *Convergence of parallel multisplitting iterative methods for M–matrices*, Linear Algebra Appl., 88-89 (1987), pp. 559-573.
[19] Wilhelm Niethammer, *On different splittings and the associated iteration methods*, SIAM J. Numer. Anal., 16 (1979), pp. 186-200.
[20] Dianne P. O'Leary and Robert E. White, *Multi-splittings of matrices and parallel solution of linear systems*, SIAM J. Algebraic Discrete Methods, 6 (1985), pp. 630-640.
[21] James M. Ortega and Werner C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Prentice-Hall International, Inc, New Jersey, 1989.
[22] Alexander M. Ostrowski, *Über die determinanten mit überwiegender hauptdiagonale*, Coment. Math. Helv., 10 (1937), pp. 69-96.
[23] ———, *Determinanten mit überwiegender hauptdiagonale und die absolute konvergenz von linearen iterations prozessen*, Coment. Math. Helv., 30 (1956), pp. 175-210.
[24] José Penadés, *Métodos Iterativos Paralelos para la Resolución de Sistemas Lineales basados en Multiparticiones*. PhD thesis, Departamento de Tecnología Informática y Computación, Universidad de Alicante, December 1993. In Spanish.
[25] Daniel B. Szyld and Mark T. Jones, *Two–stage and multisplitting methods for the parallel solution of linear systems*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 671–679.
[26] Richard S. Varga *Matrix Iterative Analysis*. Prentice Hall, 1962.
[27] ———, On recurring theorems on diagonal dominance, Linear Algebra Appl., pp. 13 (1976), pp. 1-9.
[28] Deren R. Wang *On the convergence of the parallel multisplitting AOR algorithm*, Linear Algebra Appl., 154-156 (1991), pp. 473–486.
[29] Robert E. White. Multisplittings and parallel iterative methods, Comput. Methods Appl. Mech. and Engrg, 64 (1987), pp. 567-577.
[30] ———, Multisplitting with different weighting schemes, SIAM J. Matrix Anal. Appl., 10 (1989), pp 481–493.
[31] ———, Multisplitting of a symmetric positive definite matrix. SIAM J. Matrix Anal. Appl., 11 (1990), pp. 69–82.
[32] David M. Young *Iterative Solution of Large Linear Systems*. Academic Press, New York, 1972.

ETNA
Kent State University
etna@mcs.kent.edu

		Alliant FX/80			IBM SP2		
$q(l,j)$	Iter.	Seq. time	Par. time	Speed-up	Seq. time	Par. time	Speed-up
1^8	51656	67717.8	13229.3	5.1	2605.5	591.3	4.4
2^8	26607	40853.1	8091.4	5.0	1499.8	319.3	4.7
4^8	13706	27489.9	5530.9	5.0	917.7	209.4	4.4
10^8	5710	20473.2	4124.8	5.0	609.5	107.8	5.7
30^8	1998	21395.5	4306.2	5.0	673.7	127.6	5.3
$6^6 3^2$	15634	35019.2	5852.1	6.0	1462.7	215.1	6.8
$17^1 15^5 8^1 9^1$	5936	26556.6	4205.3	6.3	770.4	108.5	7.1
$40^1 30^5 18^1 20^1$	2757	26811.3	4470.8	6.0	847.7	123.4	6.9

Table 5.1
Synchronous non-stationary models on multiprocessors Alliant FX/80 and IBM SP2. Size of matrix A: 50000.

	Alliant FX/80	IBM SP2
Iter.	CPU time	CPU time
51240	27057.7	1066.2

Table 5.2
Sequential Gauss-Seidel algorithm on multiprocessors Alliant FX/80 and IBM SP2. Size of matrix A: 50000.

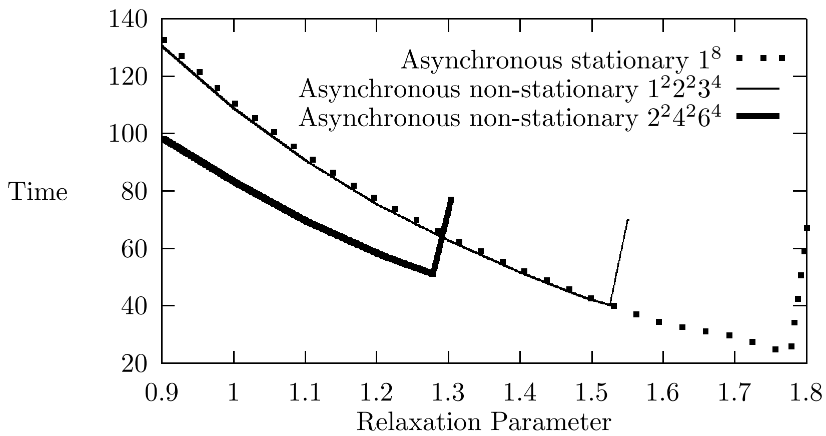


Fig. 5.4. *Asynchronous algorithms, size of A:* 4096.