DIFFERENT MODELS OF PARALLEL ASYNCHRONOUS ITERATIONS WITH OVERLAPPING BLOCKS

Daniel B. Szyld

Report 97-81
August 1997

Different Models of Parallel Asynchronous Iterations with Overlapping Blocks

Daniel B. Szyld*

Dedicated to the memory of Paulo Jorge Paes Leme

Abstract

Different computational and mathematical models used in the programming and in the analysis of convergence of asynchronous iterations for parallel solution of linear systems of algebraic equations are studied. The differences between the models are highlighted. Special consideration is given to models that allow for overlapping blocks, i.e., for the same variable being updated by more than one processor.

Keywords. Linear systems, H-matrices, chaotic relaxation, iterative methods, parallel algorithms, asynchronous algorithms.

AMS Subject Classification: 65F10. **CR:** G 1.3

1 Introduction

In this paper we track the development of different models of asynchronous iterations. We trace the history of these parallel methods, where each processor executes its own instructions, reading data produced by other processors, but not waiting for new data if the latter is not yet available. There is no synchronization barrier to overcome, and thus all processors compute their tasks with no interruption. These methods usually have a slower asymptotic convergence rate than methods which wait for newer information, but because they do not wait, in many instances they produce an approximation to the solution of the problem in much less computational time.

*Department of Mathematics, Temple University, Philadelphia, Pennsylvania 19122-6094, USA (szyld@math.temple.edu). This work was supported by the National Science Foundation grant DMS-9625865.

(References are not mentioned in this introduction, but in each section when the discussion brings attention to them.)

Asynchronous methods have been proposed and applied to different types of linear and nonlinear algebraic equations, optimization problems, and differential equations. To keep the discussion focused, we restrict our study mostly to linear equations. Furthermore, we concentrate on block methods with overlaps, i.e., when some variables are approximated by more than one processor. This is the computational model which is most beneficial.

We make the distinction between computational models (represented in this paper by a pseudocode) and mathematical models, which give an expression of the iterate as a function of the previous ones. Mathematical models are used to describe the computational ones, and can therefore be used to study their convergence properties. We describe different mathematical models in a similar manner, exposing how they relate to each other. We show how the different mathematical models have attempted to describe the computational model, but not always succeeded in capturing some elements. For example, not all mathematical models had vectors keeping the different values that the overlapping variables are taking in the different processors.

2 First computational model

We begin by considering a *computational model* for the parallel asynchronous solution (by blocks) of a large and sparse linear system of algebraic equations of the form

$$\mathbf{A}\mathbf{v} = \mathbf{b}, \tag{1}$$

where $\mathbf{A} = (a_{ij}) \in \mathbb{R}^{n\times n}$ is a nonsingular matrix, and $\mathbf{b} \in \mathbb{R}^n$ is given. To that end, let $\mathbb{I}_n = \{1, 2, \ldots, n\}$ and L sets $G_\ell \subset \mathbb{I}_n$, $\ell = 1, \ldots, L$, so that $\cup_{\ell=1}^{L} G_\ell = \mathbb{I}_n$, be the sets defining the blocks. We call the set $G = \{G_1, G_2, \ldots, G_L\}$ a *covering* of $\mathbb{I}_n$. The variables with indices in $G_i \cap G_j$ $(i \neq j)$ are said to be the variables with overlap. If $G_i \cap G_j = \emptyset$ for all $i \neq j$, i, $j = 1, \ldots, L$, the set G is a *partition* of $\mathbb{I}_n$. Let n_ℓ be the cardinality of G_ℓ, $\ell = 1, \ldots, L$. The set G determines a covering (or partition) of any vector $\mathbf{x} \in \mathbb{R}^n$, so that if $G_\ell = \{\alpha_1, \alpha_2, \ldots, \alpha_{n_\ell}\}$, we have the subvectors $\mathbf{x}_\ell = (x_{\alpha_1}, x_{\alpha_2}, \ldots, x_{\alpha_{n_\ell}})^T \in \mathbb{R}^{n_\ell}$, $\ell = 1, \ldots, L$. Similarly, the submatrices (or blocks) of $\mathbf{A}$ determined by G are $\mathbf{A}_{\ell k} = (a_{\alpha_i \gamma_j}) \in \mathbb{R}^{n_\ell \times n_k}$, where $\alpha_i \in G_\ell$, $i = 1, \ldots, n_\ell$ and $\gamma_j \in G_k$, $j = 1, \ldots, n_k$.

In practice, the set G could be determined by the geometry of the underlying problem, e.g., when domain decomposition methods are used and $\mathbf{A}$ corresponds to a discretization of a differential equation [59] (each subdomain containing the variables in $\mathbf{x}_\ell$, say, and the variables with overlap correspond to the overlap between the subdomains), or it may originate from a reordering algorithm applied to the sparse matrix $\mathbf{A}$, as done, e.g., in [7], [50], [58]. Depending on the geometry, or on the ordering algorithm used, the cardinality of sets G_ℓ, n_ℓ, may vary over a wide range, in other words, some of the square matrices $\mathbf{A}_{\ell\ell}$ (which are assumed to be nonsingular throughout the paper) may be small while others may be large. This occurs, e.g., when the solution of a differential equation needs to be resolved in some subdomains with more detail than in others. In the algorithms discussed in this paper, the $n_\ell \times n_\ell$ linear systems are solved by different processors, and thus, if these systems have a similar sparsity pattern, the corresponding computational effort also varies widely. This is usually referred to as load imbalance. Asynchronous methods like the ones studied in this paper overcome the problem of load imbalance by continuing the computations with the available data; see, e.g., the numerical experiments in [34].

It is useful sometimes to assume that the variables (and equations) are renumbered so that the indices within each set G_ℓ are numbered consecutively, and that if $\alpha \in G_\ell$, $\gamma \in G_k$ with $\ell < k$, then $\alpha \leq \gamma$. This is not always possible in the case of overlap, though. We will nevertheless adopt this notation, and note that all the material covered in this paper can be shown without this additional assumption. We observe that only in the case that G is a partition of $\mathbb{I}_n$, we can write $\mathbf{x}^T = (\mathbf{x}_1^T, \mathbf{x}_2^T, \ldots, \mathbf{x}_L^T)$ and

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \cdots & \mathbf{A}_{1L} \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \cdots & \mathbf{A}_{2L} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{L1} & \mathbf{A}_{L2} & \cdots & \mathbf{A}_{LL} \end{bmatrix}. \tag{2}$$

If the set G is not a partition, but a covering, then, there are overlapping blocks, and there are some variables belonging to more than one subvector $\mathbf{x}_\ell$, and the representation (2) is no longer appropriate. We emphasize that in many iterative methods, the overlap plays an important role in qualitatively improving the convergence of the methods; see, e.g., [1], [15], [22], [25], [32], [33], [38], [40] [41], [43], [59], [61], [66]. This is why we concentrate on the overlapping cases in this paper.

We are ready to describe the first asynchronous computational model, i.e., how the computer is actually programmed to execute its instructions. In the rest of the paper, we refer to this model simply as *asynchronous iterations.*

Given an initial approximation $\mathbf{x}$ to the solution of (1), each processor of a parallel computer executes the following procedure, independently of each other.

```
1. determine $\ell$.
2. read $\mathbf{x}_k$, $k \neq \ell$.
3. solve (or approximate) $\mathbf{A}_{\ell\ell}\mathbf{x}_\ell = \mathbf{b}_\ell - \sum_{k\neq\ell}\mathbf{A}_{\ell k}\mathbf{x}_k$.
4. write $\mathbf{x}_\ell$.
```

Steps 1 to 4 are repeated until some termination or stopping criterion is met.

We do not discuss the stopping criterion in this paper, and instead refer the reader to [27] and the references given therein. We say that this is an asynchronous iteration since there is no synchronization between the execution of the instructions of the processor with those of any other processor, i.e., there is no wait for data during execution. We emphasize that in this model, the processor reads all subvectors $\mathbf{x}_k$, $k \neq \ell$ in step 2 which are available *before* the beginning the execution of step 3. Observe that when there is overlap, if the index α belongs to more than one set G_ℓ, say, then we may have two or more *different* values for x_α at any given time.

This asynchronous iteration is a generalization of the block Jacobi method. For a description of the block Jacobi method and its convergence analysis, see any of the classical texts, e.g., [9], [63], [67].

We have presented this computational model in a very general way, so as to include programming different types of architectures. For example, consider a distributed computer with L processors, in which each processor has local memory but there is no global memory. Then each processor has a value of ℓ assigned to it from the beginning of all computations, and step 1 is trivial. The `write` instruction in step 4 implies that the values of the subvector $\mathbf{x}_\ell$ are broadcasted, or sent to all the other processors. Thus, the `read` instruction in step 2 is a local read, i.e., it reads the values received previously, until the moment before step 3 begins. Any values of subvectors $\mathbf{x}_k$ received while step 3 is executing are not used until the next iteration (the situation in which these values are used during the current iteration is considered in the computational model described in Section 5). On the other hand, if there are fewer than L processors, the processes 1 to 4 combined can be considered as a token in a queue, and the value of ℓ is part of that token; see, e.g., [16], [17]. The free processor grabs a token from the queue, executes it, and puts it back at the end of the queue.

In this general setting the computational model presented here represents the way asynchronous iterations are programmed in most cases reported in the literature; see, e.g., [11]. In this paper we concentrate on the solution of the linear system (1), but most of our analysis applies also to the solution of nonlinear systems using asynchronous fixed point iterations. In other words, given a nonlinear function $\mathbf{F} : \mathbb{R}^n \rightarrow \mathbb{R}^n$, the solution of $\mathbf{F}(\mathbf{x}) = \mathbf{x}$ is obtained by a similar computational model, where at step 3, the processor solves for $\mathbf{x}_\ell$ (or its approximation) in the equation $\mathbf{F}_\ell(\mathbf{x}_1, \ldots, \mathbf{x}_\ell, \ldots, \mathbf{x}_L) = \mathbf{x}_\ell$. This model then relates to the work by many authors; see, e.g., [6], [10], [14], [26], [37], [42], [45], [47], [48], [56], [62], and the references given therein. See also further extensions in Section 6.

3 First Mathematical Models

The first *mathematical model* analyzing the convergence of the asynchronous iteration dates from 1959. At that time there were no parallel machines, and Schechter [57] studied his method as a block counterpart to the (point) *free steering* iterations due to Ostrowski [51]. Schechter explicitly considered overlapping blocks. Ostrowski [52] later studied this block version (without overlap) as well. Even though the computational model they had in mind was strictly sequential, having the (single-processor) computer execute the asynchronous iteration steps 1 to 4 repeatedly in a sequential manner, their convergence analyses equally apply to the parallel setting. We emphasize the distinction between the mathematical models, which attempt to represent mathematically the asynchronous iterations, i.e., the computational model, and the latter, which represent the way the computations are carried out.

In order to describe the mathematical models, let i be the iteration number associated with the time at which the asynchronous iteration procedure is executed. To be precise, we associate to the tag i, the time at the beginning of the execution of step 2. The subvectors of $\mathbf{x}$ are tagged with this iteration number when they are computed. Thus, at the end of steps 3 and 4, $\mathbf{x}_\ell^i$ has been computed and stored. Define $N_i \subset \mathbb{I}_L = \{1, \ldots, L\}$ as $\ell \in N_i$ if $\mathbf{x}_\ell$ is computed at the time associated with the tag i. Let $r(k, i)$ be the tag of the subvector $\mathbf{x}_k$ available at the beginning of the computation of $\mathbf{x}_\ell^i$ (if $\ell \in N_i$), i.e., at step 2, $\mathbf{x}_k^{r(k,i)}$, which was (previously) computed at the time associated with the tag $r(k, i)$, is read. With this notation $\mathbf{x}_\ell^{r(\ell,i)}$ is the subvector $\mathbf{x}_\ell$ available at the beginning of the computation of $\mathbf{x}_\ell^i$ (if $\ell \in N_i$). This vector is not used in the models described in this section, but it is used in a model described in Section 4. Sometimes the difference $d(k, i) = i - r(k, i)$ is called a *delay*. If all delays are zero, i.e., if there are no delays, and if there is no overlap, the asynchronous iteration behaves like a standard (synchronous) iterative method, such as, e.g., block Jacobi, etc. With this notation, the mathematical model in [57] and [52] is as follows.

Let an initial approximation to the solution of (1), $\mathbf{x}^0$ be given, then, for $i = 1, \ldots,$

$$\mathbf{x}_\ell^i = \begin{cases} \mathbf{x}_\ell^{i-1} & \text{if } \ell \notin N_i \\ \text{the solution of } \mathbf{A}_{\ell\ell}\mathbf{x}_\ell^i = \mathbf{b}_\ell - \sum\limits_{k \neq \ell} \mathbf{A}_{\ell k}\mathbf{x}_k^{r(k,i)} & \text{if } \ell \in N_i. \end{cases} \tag{3}$$

The basic assumptions used in the mathematical models in [52], [57] are the following.

(i) $r(k,i) \le i$ for all $k \in \mathbb{I}_L$, $i = 1, \ldots.$
(ii) The set $\{i \mid \ell \in N_i\}$ is unbounded for all $\ell \in \mathbb{I}_L$.
(iii) $\lim\limits_{i\to\infty} r(k,i) = \infty$ for all $k \in \mathbb{I}_L$.

We should mention that assumption (iii), which states that, as the iteration proceeds, iterates which lag too far behind are no longer used, is only stated implicitly in [57], while assumption (i), which states that only previously computed iterates can be used, is also only implicit in [52], [57]. Condition (ii) states that no subvector ceases to be renewed as the iterations proceed.

We say that (3), together with (i)–(iii), form the *basic mathematical model of asynchronous iterations*. Before we state the convergence theorems from [52], [57] associated with this mathematical model, we introduce some notation and definitions, following, e.g., [9], [63]. Given a vector $\mathbf{x} \in \mathbb{R}^n$, we say that it is nonnegative (positive), denoted $\mathbf{x} \ge 0$ ($\mathbf{x} > 0$), if all components of $\mathbf{x}$ are nonnegative (positive). Similarly, if $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, $\mathbf{x} \ge \mathbf{y}$ means that $\mathbf{x} - \mathbf{y} \ge 0$. These definitions carry over immediately to matrices. For any matrix $\mathbf{A} = (a_{ij}) \in \mathbb{R}^{n\times n}$, we define its comparison matrix $\langle \mathbf{A} \rangle = (\alpha_{ij})$ by $\alpha_{ii} = |a_{ii}|$, $\alpha_{ij} = -|a_{ij}|$, $i \neq j$. A nonsingular matrix $\mathbf{A}$ is called monotone if $\mathbf{A}^{-1} \ge O$, it is called an M-matrix if it has non–positive off–diagonal entries and it is monotone, and it is called an H-matrix, if $\langle \mathbf{A} \rangle$ is an M-matrix. H-matrices are generalized diagonally dominant matrices; see, e.g., [64]. The representation $\mathbf{A} = \mathbf{M} - \mathbf{N}$ is called a splitting if $\mathbf{M}$ is nonsingular, it is called and M-splitting if $\mathbf{M}$ is an M-matrix and $\mathbf{N} \ge O$, a regular splitting if $\mathbf{M}^{-1} \ge O$ and $\mathbf{N} \ge O$, and a weak regular splitting if $\mathbf{M}^{-1} \ge O$ and $\mathbf{M}^{-1}\mathbf{N} \ge O$.

Given a set G, and appropriate norms in $\mathbb{R}^{n_\ell}$, for $\mathbf{A} \in \mathbb{R}^{n\times n}$ we define its type-I comparison matrix $\langle\langle \mathbf{A} \rangle\rangle = (\bar{\alpha}_{\ell k}) \in \mathbb{R}^{L\times L}$ as $\bar{\alpha}_{\ell\ell} = \|\mathbf{A}_{\ell\ell}^{-1}\|^{-1}$, $\bar{\alpha}_{\ell k} = -\|\mathbf{A}_{\ell k}\|$, $\ell \neq k$, $\ell, k \in \mathbb{I}_L$. This name is used in [3]; see also [29], [53], [55].

Theorem 3.1 [57] *Let G be a covering (or a partition) of $\mathbb{I}_n$. Let $\mathbf{A}$ and each of the matrices $\mathbf{A}_{\ell\ell}$ be monotone, $\ell \in \mathbb{I}_L$. Let $(-\mathbf{A}_{\ell k}) \ge O$, $\ell \neq k$, $\ell, k \in \mathbb{I}_L$. Let conditions* (i)–(iii) *hold. Then, the asynchronous method* (3) *converges to the solution of* (1) *for any initial vector* $\mathbf{x}^0$.

We point out again that some components of the vector $\mathbf{x}$, say x_j, have different values when the index j belongs to more than one set G_ℓ. It follows from Theorem 3.1 that *each* copy of x_j converges to the *same* value, namely v_j, where $\mathbf{v}$ is the solution of (1). This situation is similar to the convergence of the Schwarz alternating procedure with overlapping subdomains; see, e.g., [59].

Theorem 3.2 [52] *Let G be a partition of $\mathbb{I}_n$. Let $\mathbf{A}$ be such that $\langle\langle\mathbf{A}\rangle\rangle$ is an M-matrix. Let conditions* (i)–(iii) *hold. Then, the asynchronous method* (3) *converges to the solution of* (1) *for any initial vector* $\mathbf{x}^0$.

We comment briefly on the proof of Theorem 3.1 in [57]. Given a fixed covering G, the norm of a vector $\mathbf{x} \in \mathbb{R}^n$ used for the convergence analysis is

$$||\mathbf{x}|| = \sum_{\ell=1}^{L} ||\mathbf{x}_\ell||_\ell, \tag{4}$$

where $||\cdot||_\ell$ is any norm in $\mathbb{R}^{n_\ell}$ (recall that there is overlap between these subvectors). For the proof, the matrices $\mathbf{A}_{\ell\ell}^{-1}$ are embedded into $n \times n$ nonnegative matrices $\mathbf{K}_\ell = (\kappa_{ij})$, so that $\kappa_{ij} = 0$ if either i or $j \notin G_\ell$, and the nonzero part has the corresponding elements of $\mathbf{A}_{\ell\ell}^{-1}$. Then, nonnegative matrices of the form $\mathbf{T}_\ell = \mathbf{I} - \mathbf{K}_\ell\mathbf{A}$ are shown to map the residual at one step to the one at the next step. We also note that the corresponding theorem in [57] is more general, including in its convergence theory under- and overrelaxation, and regular splittings other than block Jacobi.

The first mathematical model of asynchronous iterations with parallel machines in mind is due to Chazan and Miranker [24]. Their model corresponds to point methods, i.e., $L = n$, and $G_\ell = \{\ell\}$, $\ell \in \mathbb{I}_n$. They mention the block method as a simple extension, but their analysis is restricted to the nonoverlapping case. Their mathematical model is similar to (3) (in the point case), with the following assumptions.

(i′) $0 \leq d(k, i)$ for all $k \in \mathbb{I}_L$, $i = 1, \ldots$.
(ii′) $\ell \in N_i$ for some i infinitely often for all $\ell \in \mathbb{I}_L$.
(iii′) There exists a fixed integer d such that $d(k, i) < d$ for all $k \in \mathbb{I}_L$, $i = 1, \ldots$.

It is easy to see that conditions (i)–(ii) and (i′)–(ii′) are equivalent. If the sequence $r(k, i)$ satisfies condition (ii′), or equivalently (ii), some authors call this sequence *admissible*; see, e.g., [18], [20]. In these papers, a sequence satisfying (iii′) is called *regulated.* Condition (iii) is more general than (iii′) since no uniform bound d is required, as pointed out in, e.g., [6], [11], [21]; see also [60] for an analysis of a condition other than (iii′). Most convergence results in this paper correspond to the more general assumption (iii). We should mention though that there are models where the additional assumption on uniformity is required; see, e.g., [11, Ch. 7].

4 The Multisplitting Models

O'Leary and White [49] introduced the multisplitting method for the parallel solution of a system of the form (1), in which each processor computes a different approximation to the solution, and then a weighted average of these approximations is taken as the iterate. Only when these weights are nonzero the corresponding components need to be computed. This method has been extensively studied and extended to nonlinear problems, differential equations, and other directions; see, e.g., [18], [23], [40], [39], [54], [65]. For the multisplitting method, let $\mathbf{A} = \mathbf{M}_\ell - \mathbf{N}_\ell$, $\ell \in \mathbb{I}_L$, be splittings, and let $\mathbf{E}_\ell \in \mathbb{R}^{n\times n}$ be nonnegative diagonal *weighting* matrices such that $\sum_{i=1}^{L} \mathbf{E}_\ell = \mathbf{I}$. If the weighting matrices $\mathbf{E}_\ell$ consist of all zeros and ones, then, we say that there is no overlap. There is overlap if some entries are such that $0 < (\mathbf{E}_\ell)_{jj} < 1$ for some ℓ, j.

Multisplittings have proved to be a good mathematical tool to analyze different types of block methods. We describe here mathematical models of asynchronous iterations based on this idea. To that end, consider weighting matrices $\mathbf{E}_\ell$ with nonzeros only in diagonal entries with indices in G_ℓ, and splittings $\mathbf{A} = \mathbf{M}_\ell - \mathbf{N}_\ell$ so that $(\mathbf{M}_\ell)_{\ell_i \ell_j} = a_{\ell_i \ell_j}$, $i, j = 1, \ldots, n_\ell$, i.e., so that the $\mathbf{M}_\ell$ has $\mathbf{A}_{\ell\ell}$ as the ℓth diagonal block. Usually, $\mathbf{M}_\ell$ is taken as block diagonal. The diagonal blocks other than the ℓth are not relevant in these mathematical models since they do not influence the outcome of the computations being modeled, and they can be taken to be, e.g., the identity or the diagonal blocks of $\mathbf{A}$. With the splittings thus constructed, $\mathbf{E}_\ell \mathbf{M}_\ell^{-1}\mathbf{y} = \mathbf{A}_{\ell\ell}^{-1}\mathbf{y}_\ell$ for any vector $\mathbf{y} \in \mathbb{R}^n$. We note that $\mathbf{E}_\ell \mathbf{M}_\ell^{-1}$ is therefore equal to the matrix $\mathbf{K}_\ell$ used by Schechter in his proof of Theorem 3.1, and described in Section 3. With this notation, the hypothesis in Theorem 3.1 is that $\mathbf{A} = \mathbf{M}_\ell - \mathbf{N}_\ell$ are regular splittings, $\ell \in \mathbb{I}_L$. Denote by $\tilde{\mathbf{x}}_\ell$ the vector computed by the ℓth splitting, $\ell \in \mathbb{I}_L$. This occurs, e.g., in the computation by the ℓth processor. Observe that $\tilde{\mathbf{x}}_\ell \in \mathbb{R}^n$ while $\mathbf{x}_\ell \in \mathbb{R}^{n_\ell}$ and in these models we consider $\mathbf{x}_\ell$ as a subvector of $\tilde{\mathbf{x}}_\ell$.

The first mathematical model of asynchronous iterations using multisplittings was presented by Bru, Elsner and Neumann [18], using assumptions (i′)–(iii′). Its extension to the block case can be found, e.g., in [19], [20], [44]. The following description of this mathematical model is not exactly the one used in these references, but it can be shown to be equivalent in the non-overlap case.

Given an initial vector $\mathbf{x}^0$, set $\tilde{\mathbf{x}}_\ell^0 = \mathbf{x}^0$, for $\ell \in \mathbb{I}_L$, then, for $i = 1, \ldots,$

$$\tilde{\mathbf{x}}_\ell^i = \begin{cases} \tilde{\mathbf{x}}_\ell^{i-1} & \text{if } \ell \notin N_i \\ (\mathbf{I} - \mathbf{E}_\ell)\tilde{\mathbf{x}}^{r(\ell,i)} + \mathbf{E}_\ell \mathbf{M}_\ell^{-1}\left[\mathbf{N}_\ell\left(\sum_{k=1}^{L} \mathbf{E}_k \tilde{\mathbf{x}}_k^{r(k,i)}\right) + \mathbf{b}\right] & \text{if } \ell \in N_i. \end{cases} \tag{5}$$

As pointed out recently in [3], the model (5) (and its original formulation in [18]) represents the asynchronous iterations described in Section 2 only in the case of non-overlap, i.e., when G is a partition and $(\mathbf{E}_\ell)_{jj} = 1$, if $j \in G_\ell$, and zero otherwise. In the case of overlap, this model would represent another computational algorithm, where a convex combination of the old iterate and the new one is computed; see (5). To our knowledge this overlap case has not been implemented in practice; e.g., the experiments in [19], [20], [44], correspond to partitions of $\mathbb{I}_n$.

The second mathematical model presented in this section is the linear counterpart to that in [3], and generalizes the models in [21], [35]. It is another attempt to represent the asynchronous iterations in the case of overlap.

Given an initial vector $\mathbf{x}^0$, set $\tilde{\mathbf{x}}_\ell^0 = \mathbf{x}^0$, for $\ell \in \mathbb{I}_L$, then, for $i = 1, \ldots,$

$$\tilde{\mathbf{x}}_\ell^i = \begin{cases} \tilde{\mathbf{x}}_\ell^{i-1} & \text{if } \ell \notin N_i \\ \mathbf{M}_\ell^{-1}\left[\mathbf{N}_\ell\left(\sum_{k=1}^{L} \mathbf{E}_k \tilde{\mathbf{x}}_k^{r(k,i)}\right) + \mathbf{b}\right] & \text{if } \ell \in N_i, \end{cases} \tag{6}$$

and at the end of the process, the approximation to the solution of (1) is $\mathbf{x}^i = \sum_{\ell=1}^{L} \mathbf{E}_\ell \tilde{\mathbf{x}}_\ell^i$.

Note that in contrast to model (3), the possible different values in the overlap are weighted at each step and combined. We mention here that a similar construct is present in the models studied in [11, sec. 7.7], [13], where convex combinations of the iterates are taken.

The last model in this section is the asynchronous weighted additive Schwarz method (AWAS) introduced recently in [34]. It is a mathematical model which keeps L separate vectors, one for each set G_ℓ (we abuse slightly the notation and call these vectors $\tilde{\mathbf{x}}_\ell \in \mathbb{R}^n$), and thus allows for different values of the same component in the overlap to be represented, just as in the discussion after Theorem 3.1. This is accomplished by having L separate sets of weighting matrices. In fact, one can have separate sets of matrices in each iteration i to reflect possible different choices of the overlapped variables, depending, e.g., on which one has been computed most recently. The weighting matrices are then $\mathbf{E}_{k\ell}^i$, such that $\sum_{k=1}^{L} \mathbf{E}_{k\ell}^i = \mathbf{I}$, for $\ell \in \mathbb{I}_L$,

$i = 1, \ldots$; see also [4], where similar weighting matrices are used. The AWAS model is as follows.

Given (possibly different) initial approximations $\tilde{\mathbf{x}}_\ell^0$, $\ell \in \mathbb{I}_L$, for $i = 1, \ldots$,

$$\tilde{\mathbf{x}}_\ell^i = \begin{cases} \tilde{\mathbf{x}}_\ell^{i-1} & \text{if } \ell \notin N_i \\ \sum\limits_{k=1}^{L} \mathbf{E}_{k\ell}^i \mathbf{M}_k^{-1} \left(\mathbf{N}_k \tilde{\mathbf{x}}_k^{\tilde{r}(k,i)} + \mathbf{b} \right) & \text{if } \ell \in N_i. \end{cases} \tag{7}$$

We emphasize that this is a mathematical model, and that the computations in (7) are not all performed in the same processor. In fact, each computation

$$\mathbf{M}_k^{-1} \left(\mathbf{N}_k \tilde{\mathbf{x}}_k^{\tilde{r}(k,i)} + \mathbf{b} \right) \tag{8}$$

takes place in a different processor, $k \in \mathbb{I}_L$. Here the tag $\tilde{r}(k,i)$ corresponds to the time where the computation (8) is about to begin. These tags are thus different than those in the other mathematical models, though they satisfy conditions (i)–(iii) as well. In practical implementations, it is not necessary to keep the L vectors of length n in memory, since only their subvectors $\mathbf{x}_\ell$ of length n_ℓ have useful (non-redundant) information. Thus $\sum\limits_{\ell=1}^{L} n_\ell$, and not nL memory locations suffice to store the iteration vectors.

Both models (6) and (7) were shown to be convergent under conditions (i)–(iii) and additional hypotheses (e.g. on the splittings) when $\mathbf{A}$ is either a monotone matrix or an H-matrix. For brevity, we only reproduce the result from [34], as Theorem 4.1 below, where the notation $|\mathbf{A}|$ stands for a matrix whose entries are $|a_{ij}|$. Extensive experiments reported in [34] reveal that, in the case of overlap, computations with (7), i.e., the weighted additive Schwarz version – which truly represents the asynchronous iterations, keeping separate values for the variables in the overlap – outperforms the computations with (6), i.e., the multisplitting version.

Theorem 4.1 *Let the diagonal nonnegative weighting matrices* $\mathbf{E}_{\ell k}^i$ *be such that* $\sum\limits_{k=1}^{L} \mathbf{E}_{k\ell}^i = \mathbf{I}$*, for* $\ell \in \mathbb{I}_L$*,* $i = 1, \ldots$*. Let conditions* (i)–(iii) *hold. Then, for each* $\ell \in \mathbb{I}_L$*, each sequence* $\{\tilde{\mathbf{x}}_\ell^i\}$ *computed by the asynchronous method* (7) *converges to the solution of* (1) *for any set of initial vectors* $\tilde{\mathbf{x}}_\ell^0$*,* $\ell \in \mathbb{I}_L$*, in the following two cases.*
(a) $\mathbf{A}^{-1} \geq O$ *and each splitting* $\mathbf{A} = \mathbf{M}_\ell - \mathbf{N}_\ell$ *is weak regular,* $\ell \in \mathbb{I}_L$*.*
(b) $\mathbf{A}$ *is an* H*-matrix and* $\langle \mathbf{A} \rangle \leq \langle \mathbf{M}_\ell \rangle - |\mathbf{N}_\ell|$*,* $\ell \in \mathbb{I}_L$*.*

The proof of Theorem 4.1, as well as those in [3], [35] are based on extensions of a convergence result of El Tarazi [28], like the following.

Theorem 4.2 [35] *Let G be a covering (or a partition) of $\mathbb{I}_n$. Let $S(i)$ be a sequence of operators on $\mathbb{R}^n$ having a common fixed point $\mathbf{x}_*$. Let $\|\cdot\|_\ell$ be a norm on $\mathbb{R}^{n_\ell}$, $\ell \in \mathbb{I}_L$. Let $\mathbf{a} \in \mathbb{R}^L$, $\mathbf{a} > 0$ and denote $\|\cdot\|_{\mathbf{a}}$ the weighted max-norm on $\mathbb{R}^n$ given by*

$$\|\mathbf{x}\|_{\mathbf{a}} = \max_{\ell=1,\cdots,L}\{\frac{1}{a_\ell}\|\mathbf{x}_\ell\|_\ell\}. \tag{9}$$

For all $i = 1, 2, \ldots$, assume that there exists a constant $\theta \in [0, 1)$ such that

$$\|S(i)\mathbf{x} - \mathbf{x}_*\|_{\mathbf{a}} \le \theta\|\mathbf{x} - \mathbf{x}_*\|_{\mathbf{a}} \quad \textit{for all } \mathbf{x} \in \mathbb{R}^n.$$

Assume further that conditions (i)–(iii) *hold. Then the asynchronous iteration*

$$\mathbf{x}_\ell^i = \begin{cases} x_\ell^{i-1} & \text{if } \ell \notin N_i \\ S(i)_\ell\left(\left((\mathbf{x}_1^{r(1,i)})^T, \ldots, (\mathbf{x}_L^{r(L,i)})^T\right)^T\right) & \text{if } \ell \in N_i, \end{cases}$$

$i = 1, 2, \ldots$, converges to $\mathbf{x}_*$ for any initial guess $\mathbf{x}_0$.

We point out that, as a consequence of using Theorem 4.2, the results in the last few years, both for linear and nonlinear systems, all use the weighted max norm (9), as compared to the norm (4), say.

5 Totally Asynchronous Iterations

We present a computational model in which the (components of the) subvectors (or vectors) are read only when needed. This implies that these might be newer versions than those at the beginning of the iteration as in the asynchronous iterations of Section 2.

Given an initial approximation $\mathbf{x}$ to the solution of (1), each processor of a parallel computer executes the following procedure, independently of each other.

```
1. determine $\ell$, and set $\mathbf{y} \leftarrow \mathbf{b}_\ell$.
2. for each $k \in \mathbb{I}_L$, $k \neq \ell$, read $\mathbf{x}_k$, and compute $\mathbf{y} \leftarrow \mathbf{y} - \mathbf{A}_{\ell k}\mathbf{x}_k$.
3. solve (or approximate) $\mathbf{A}_{\ell\ell}\mathbf{x}_\ell = \mathbf{y}$.
4. write $\mathbf{x}_\ell$.
```

Steps 1 to 4 are repeated until some termination or stopping criterion is met.

We emphasize the distinction between the step 2 here and the computational model in Section 2. New information from other processors keeps arriving while the computation of the right-hand side progresses, and some components of the subvector $\mathbf{x}_k$ might be different when used than at the beginning of the computation of the step. In other words, included in this computational model are models in which part of the data is read during the computational steps. For example, the model might include reads of some components of subvectors, which become available during the calculations, i.e., partial updatings. Following [35], we call this model *totally asynchronous iterations.*

It follows that this totally asynchronous iterations model reduces to the asynchronous iterations model of Section 2, e.g., when no new information arrives to processor ℓ, say, between the beginning of step 1 and step 3. This model can be expected to converge faster, since newer information is incorporated as it becomes available. However, there are cases, e.g., in the two-stage methods mentioned in Section 6, where as the newer information arrives, more computation is required. In these cases, the possible advantage of this formulation depends, e.g., on the proportion of this new computation vis a vis the overall computation, and thus on the specific problem being solved; see the discussion in [35, Section 4].

Mathematical models representing totally asynchronous iterations can be found in [21], [35], and the recent paper by Miellou, El Baz and Spiteri [46]. In this last reference, it is shown, under the hypothesis that the splittings are M-splittings, that if the initial guess is a supersolution, i.e., if $\mathbf{x}^0 \geq \mathbf{v}$, then the sequence of approximations produced by the algorithm is a monotonically decreasing sequence converging to the solution of (1). It is also the only reference we know of in which computational results with totally asynchronous iterations are reported. The proofs in the other references apply to any initial vector, and to more general splittings, and use extensions of a very general theorem due to Frommer [30], where the spaces do not need to be normed, but just to be topological product spaces with some compatibility between the topology and the partial order.

6 Final Comments

The computational and mathematical models collected in this paper apply to more general cases. Two-stage methods are those in which the linear systems in step 3 of either the asynchronous or the totally asynchronous iteration is approximated by several iterations of an inner iterative method, e.g., associated with splittings (in $\mathbb{R}^{n_\ell}$) $\mathbf{A}_{\ell\ell} = \mathbf{P}_\ell - \mathbf{Q}_\ell$, or in the case of multisplitting models, with inner splittings (in $\mathbb{R}^n$) $\mathbf{M}_{\ell\ell} = \mathbf{F}_\ell - \mathbf{G}_\ell$; see, e.g., [21], [34], [35], [44]. We note that a more general totally asynchronous iteration model applies to the two-stage methods, since new data can be read in each of the inner iterations if available; see [21], [35]. Another extension of the models discussed in this paper relates to different relaxation alternatives, including multiparameter versions; see, e.g., [2], [5], [20], [21], [36], [44]. Many of the convergence theorems mentioned in this paper apply to the two-stage methods and to the multiparameter versions, by using the induced splitting of the corresponding iteration matrix; see, [8], [34]. Finally, we mention two surveys by Bertsekas and Tsitsiklis [12] and Frommer [31], where several aspects of asynchronous iterations are discussed.

Acknowledgements. We thank Michele Benzi, Andreas Frommer, Violeta Migallón, and José Penadés for carefully reading an earlier draft. Their comments have helped improve our presentation.

References

[1] Götz Alefeld, Ingrid Lenhardt, and Günter Mayer. On multisplitting methods for band matrices. *Numerische Mathematik*, 75:267–292, 1997.

[2] Zhong-Zhi Bai. Variants of the synchronous and asynchronous matrix multisplitting unsymmetric AOR methods. *Journal of Fudan University (Natural Science)*, 34:139–148, 1995.

[3] Zhong-Zhi Bai, Violeta Migallón, José Penadés, and Daniel B. Szyld. Block and asynchronous two-stage methods for mildly nonlinear systems. Research Report 97–51, Department of Mathematics, Temple University, Philadelphia, Pa., May 1997.

[4] Zhong-Zhi Bai, Jia-Chang Sun, and De-Ren Wang. A unified famework for the construction of various matrix multisplitting iterative methods for large sparse systems of linear equations. *Computers & Mathematics with Applications*, 32:51–76, 1996.

[5] Zhong-Zhi Bai, De-Ren Wang, and D.J. Evans. Models of asynchronous parallel multisplitting relaxed iterations. *Parallel Computing*, 21:565–582, 1995.

[6] Gérard M. Baudet. Asynchronous iterative methods for multiprocessors. *Journal of the Association for Computing Machinery*, 25:226–244, 1978.

[7] Michele Benzi, Hwajeong Choi, and Daniel B. Szyld. Threshold ordering for preconditioning nonsymmetric problems. In G. Golub et al., editors, *Proceedings of the Workshop on Scientific Computing 97 – Hong Kong.* Lecture Notes in Computer Science, pages 159–165. Springer, 1997. To appear.

[8] Michele Benzi and Daniel B. Szyld. Existence and uniqueness of splittings for stationary iterative methods with applications to alternating methods. *Numerische Mathematik*, 76:309–321, 1997.

[9] Abraham Berman and Robert J. Plemmons. *Nonnegative Matrices in the Mathematical Sciences.* Academic Press, New York, third edition, 1979. Reprinted by SIAM, Philadelphia, 1994.

[10] Dimitri P. Bertsekas. Distributed asynchronous computation of fixed points. *Mathematical Programming*, 27:107–120, 1983.

[11] Dimitri P. Bertsekas and John N. Tsitsiklis. *Parallel and Distributed Computation.* Prentice Hall, Englewood Cliffs, New Jersey, 1989.

[12] Dimitri P. Bertsekas and John N. Tsitsiklis. Some aspects of parallel and distributed iterative algorithms–a survey. *Automatica*, 27:3–21, 1991.

[13] Dimitri P. Bertsekas, John N. Tsitsiklis, and Michael Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE Transactions on Automatic Control*, AC-31:803–812, 1986.

[14] Amit Bhaya, Eugenius Kaszkurewicz, and Francisco Mota. Asynchronous block-iterative methods for almost linear equations. *Linear Algebra and its Applications*, 154–156:487–508, 1991.

[15] Petter E. Bjørstad and Olof B. Widlund. To overlap or not to overlap: A note on a domain decomposition method for elliptic problems. *SIAM Journal on Scientific and Statistical Computing*, 10:1053 – 1061, 1989.

[16] Kostantinos Blathras. *A Systematic Dataflow Reduction (Relaxation) Approach for Asynchronous Parallel Algorithms.* PhD thesis, Department of Computer and Information Science, Temple University, August 1996.

[17] Kostas Blathras, Daniel B. Szyld, and Yuan Shi. Parallel processing of linear systems using asynchronous methods, April 1997. Preprint, Temple University, Philadelphia, Pa.

[18] Rafael Bru, Ludwig Elsner, and Michael Neumann. Models of parallel chaotic iteration methods. *Linear Algebra and its Applications*, 103:175–192, 1988.

[19] Rafael Bru, Violeta Migallón, and José Penadés. Chaotic inner–outer iterative schemes. In J.G. Lewis, editor, *Proceedings of the Fifth SIAM Conference on Applied Linear Algebra*, pages 434–438. SIAM, Philadelphia, Pa., 1994.

[20] Rafael Bru, Violeta Migallón, and José Penadés. Chaotic methods for the parallel solution of linear systems. *Computing Systems in Engineering*, 6:385–390, 1995.

[21] Rafael Bru, Violeta Migallón, José Penadés, and Daniel B. Szyld. Parallel, synchronous and asynchronous two-stage multisplitting methods. *Electronic Transactions on Numerical Analysis*, 3:24–38, 1995.

[22] Xiao-Chuan Cai. An optimal two-level overlapping domain decomposition method for elliptic problems in two and three dimensions. *SIAM Journal on Scientific Computing*, 14:239–247, 1993.

[23] M. Jesús Castel, Violeta Migallón, and José Penadés. Convergence of non-stationary multisplitting methods for Hermitian positive definite matrices. *Mathematics of Computations*, 1997. To appear.

[24] D. Chazan and W. Miranker. Chaotic relaxation. *Linear Algebra and its Applications*, 2:199–222, 1969.

[25] Maksymilian Dryja and Olof B. Widlund. Domain decomposition algorithms with small overlap. *SIAM Journal on Scientific Computing*, 15:604–620, 1994.

[26] Didier El Baz. M-functions and parallel asynchronous algorithms. *SIAM Journal on Numerical Analysis*, 27:136–140, 1990.

[27] Didier El Baz. A method of terminating asynchronous iterative algorithms on message passing systems. *Parallel Algorihtms and Applications*, 9:153–158, 1996.

[28] Mouhamed Nabih El Tarazi. Some convergence results for asynchronous algorithms. *Numerische Mathematik*, 39:325–340, 1982.

[29] David G. Feingold and Richard S. Varga. Block diagonally dominant matrices and generalizations of the Gerschgorin circle theorem. *Pacific Journal of Mathematics*, 12:1241–1250, 1962.

[30] Andreas Frommer. On asynchronous iterations in partially ordered spaces. *Numerical Functional Analysis and Optimization*, 12:315–325, 1991.

[31] Andreas Frommer. Parallele asynchrone Iterationen. In J. Herzberger, editor, *Wiisenschaftliches Rechnen*, chapter 4, pages 186 – 231, Berlin, 1995. Akademie verlag.

[32] Andreas Frommer and Günter Mayer. On the theory and practice of multisplitting methods in parallel computation. *Computing*, 49:63–74, 1992.

[33] Andreas Frommer and Bert Pohl. A comparison result for multisplittings and waveform relaxation methods. *Numerical Linear Algebra with Applications*, 2:335–346, 1995.

[34] Andreas Frommer, Hartmut Schwandt, and Daniel B. Szyld. Asynchronous weighted additive Schwarz methods. *Electronic Transactions on Numerical Analysis*, 5:48–61, 1997.

[35] Andreas Frommer and Daniel B. Szyld. Asynchronous two-stage iterative methods. *Numerische Mathematik*, 69:141–153, 1994.

[36] Robert Fuster, Violeta Migallón, and José Penadés. Parallel chaotic extrapolated Jacobi–like methods. *Linear Algebra and its Applications*, 247:237–250, 1996.

[37] Luc Giraud and Pierre Spiteri. Résolution parallèle de problèmes aux limites non lineaires. *Mathematical Modelling and Numerical Analysis*, 25:579–606, 1991.

[38] William D. Gropp and Barry F. Smith. Experiences with domain decomposition in three dimensions: Overlapping Schwarz methods. In Yuri A. Kuznetsov, Jacques Périaux, Alfio Quarteroni, and Olof B. Widlund, editors, *Domain Decomposition Methods in Science and Engineering: The Sixth International Conference on Domain Decomposition*, volume 157, Providence, R.I., 1994. AMS. Held in Como, Italy, June 15–19,1992.

[39] Chiou-Ming Huang and Dianne P. O'Leary. A Krylov multisplitting algorithm for solving linear systems of equations. *Linear Algebra and its Applications*, 194:9–29, 1993.

[40] Rolf Jeltsch and Bert Pohl. Waveform relaxation with overlapping splittings. *SIAM Journal on Scientific Computing*, 16:40–49, 1995.

[41] Mark T. Jones and Daniel B. Szyld. Two-stage multisplitting methods with overlapping blocks. *Numerical Linear Algebra with Applications*, 3:113–124, 1996.

[42] E. Kaszkurewicz, A. Bhaya, and D.D. Šiljak. On the convergence of parallel asynchronous block-iterative computations. *Linear Algebra and its Applications*, 131:139–160, 1990.

[43] Yuri A. Kuznetsov. Overlapping domain decomposition methods for FE-problems with elliptic singular perturbed operators. In Roland Glowinski, Yuri A. Kuznetsov, Gérard A. Meurant, Jacques Périaux, and Olof Widlund, editors, *Fourth International Symposium on Domain Decomposition Methods for Partial Differential Equations*, Philadelphia, Pa., 1991. SIAM.

[44] José Mas, Violeta Migallón, José Penadés, and Daniel B. Szyld. Nonstationary parallel relaxed multisplitting methods. *Linear Algebra and its Applications*, 241–243:733–747, 1996.

[45] Jean Claude Miellou. Algorithmes de relaxation à retards. *RAIRO*, 9:55–82, 1975.

[46] Jean Claude Miellou, Didier El Baz, and Pierre Spiteri. A new class of asynchronous iterative algorithms with order intervals. *Mathematics of Computation.* To appear.

[47] Jean Claude Miellou and Pierre Spiteri. Un critère de convergence pour des méthodes générales de point fixe. *Mathematical Modelling and Numerical Analysis*, 19:645–669, 1985.

[48] Debasis Mitra. Asynchronous relaxations for the numerical solution of differential equations by parallel processors. *SIAM Journal on Scientific and Statistical Computing*, 8:s43–s58, 1987.

[49] Dianne P. O'Leary and Robert E. White. Multi-splittings of matrices and parallel solution of linear systems. *SIAM Journal on Algebraic and Discrete Methods*, 6:630–640, 1985.

[50] James O'Neil and Daniel B. Szyld. A block ordering method for sparse matrices. *SIAM Journal on Scientific and Statistical Computing*, 11:811–823, 1990.

[51] Alexander M. Ostrowski. Determinanten mit überwiegender Hauptdiagonale und die absolute Konvergenz von linearen Iterationsprozessen. *Comentarii Mathematici Helvetici*, 30:175–210, 1955.

[52] Alexander M. Ostrowski. Iterative solution of linear systems of functional equations. *Journal of Mathematical Analysis and Applications*, 2:351–369, 1961.

[53] Ben Polman. Incomplete blockwise factorization of (block) H-matrices. *Linear Algebra and its Applications*, 90:119–132, 1987.

[54] Rosemary A. Renaut, Hans D. Mittelmann, and Qing He. Parallel multisplittings: Overview and extensions. In J.G. Lewis, editor, *Proceedings of the Fifth SIAM Conference on Applied Linear Algebra*, pages 34–38. SIAM, Philadelphia, Pa., 1994.

[55] François Robert. Blocs-H-matrices et convergence des méthodes itératives classiques par blocs. *Linear Algebra and its Applications*, 2:223–265, 1969.

[56] F. Robert, M. Charnay, and F. Musy. Itérations chaotiques série-parallèle pour des équations non-linéaires de point fixe. *Aplikace Matematiky*, 20:1–38, 1975.

[57] Samuel Schechter. Relaxation methods for linear equations. *Communications on Pure and Applied Mathematics*, 12:313–335, 1959.

[58] M. E. Sezer and D. D. Šiljak. Nested epsilon decompositions fo linear systems: Weakly coupled and overlapping blocks. *SIAM Journal on Matrix Analysis and Applications*, 12:521–533, 1991.

[59] Barry F. Smith, Petter E. Bjørstad, and William D. Gropp. *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations.* Cambridge University Press, Cambridge - New York - Melbourne, 1996.

[60] John C. Strikwerda. A convergence theorem for chaotic asynchronous relaxation. *Linear Algebra and its Applications*, 253:15–24, 1997.

[61] Daniel B. Szyld and Mark T. Jones. Two-stage and multisplitting methods for the parallel solution of linear systems. *SIAM Journal on Matrix Analysis and Applications*, 13:671–679, 1992.

[62] Paul Tseng, Dimitri P. Bertsekas, and John N. Tsitsiklis. Partially asynchronous, parallel algorithm for network flow and other problems. *SIAM Journal on Control and Optimization*, 28:678–710, 1990.

[63] Richard S. Varga. *Matrix Iterative Analysis.* Prentice-Hall, Englewood Cliffs, New Jersey, 1962.

[64] Richard S. Varga. On recurring theorems on diagonal dominance. *Linear Algebra and its Applications*, 13:1–9, 1976.

[65] Robert E. White. Parallel algorithms for nonlinear problems. *SIAM Journal on Algebraic and Discrete Methods*, 7:137–149, 1986.

[66] Robert E. White. Multisplitting of a symmetric positive definite matrix. *SIAM Journal on Matrix Analysis and Applications*, 11:69–82, 1990.

[67] David M. Young. *Iterative Solution of Large Linear Systems.* Academic Press, New York, 1971.