Matrix Multisplitting Methods with Applications to the Linear Complementarity Problems

Zhong-Zhi Bai *

State Key Laboratory of Scientific/Engineering Computing
Institute of Computational Mathematics
and Scientific/Engineering Computing
The Academy of Mathematics and Systems Sciences
Chinese Academy of Sciences, P.O.Box 2719, Beijing 100080, P.R.China
Email: bzz@lsec.cc.ac.cn

May 8, 1999

*Project 19601036 supported by the National Natural Science Foundation of China

Contents

Introduction

Given a matrix $M = (m_{kj}) \in R^{n\times n}$ and a vector $q = (q_k) \in R^n$, the linear complementarity problem is to find a vector $z \in R^n$ such that

$$Mz + q \geq 0, \quad z \geq 0 \quad \textbf{and} \quad z^T(Mz + q) = 0. \qquad (1)$$

Based on several splittings of the system matrix $M \in R^{n\times n}$, the linear complementarity problem (1) can be decomposed into independent linear complementarity problems of smaller sizes.

Through solving these sub-problems in parallel on the multiprocessor system without any communication barrier, we presented synchronous, chaotic and asynchronous multisplitting iterative method.

The asynchronous multisplitting iterative methods were established in accordance with the principle of using sufficiently and communicating flexibly the known information. Hence, they have the potentials to achieve high parallel computing efficiency in actual computations.

Numerical examples showed that the relaxed asynchronous multisplitting methods are quite efficient for solving the large sparse linear complementarity problems on the high-speed multiprocessor systems.

The Elementary Matrix Multisplitting Methods for System of Linear Equations

"***Turn large into small***" and "***divide and conquer***" are elementary principles for constructing various parallel algorithms. The matrix multisplitting idea is just an actual application of these principles to the large sparse system of linear equations

$$Mx = b, \quad M \in R^{n\times n} \text{ nonsingular }, \quad b \in R^n. \qquad (2)$$

A multisplitting of the matrix $M \in R^{n\times n}$ is a collection of triples $(B_i, C_i, E_i)(i = 1, 2, \ldots, \alpha)$ satisfying

(a) $M = B_i + C_i$, $i = 1, 2, \ldots, \alpha$;

(b) B_i **is nonsingular, i.e.,** $\det(B_i) \neq 0$, $i = 1, 2, \ldots, \alpha$; **and**

(c) $E_i(i = 1, 2, \ldots, \alpha)$ **are nonnegative diagonal matrices such that**

$$\sum_{i=1}^{\alpha} E_i = I(\textbf{the } n \times n \textbf{ identity matrix}).$$

Here:

α **is the number of processors of the referred multiprocessor system**

$E_i(i = 1, 2, \ldots, \alpha)$ **are the weighting matrices.**

The matrix multisplitting method for the system of linear equations (2) can then be described as follows:

Method 1 (O'Leary and White 1985).
Given an initial vector $x^0 \in R^n$.
For $p = 0, 1, 2, \ldots$ until $\{x^p\}$ convergence, compute

$$x^{p+1} = \sum_{i=1}^{\alpha} E_i x^{p,i},$$

where

$$x^{p,i} = B_i^{-1}(b - C_i x^p), \qquad i = 1, 2, \ldots, \alpha.$$

Several typical properties of this method are:

(a) For a fixed iterate index p, the computations of $x^{p,i} (i = 1, 2, \dots, \alpha)$ are independent of each other for various i, and therefore, they can be fulfilled in parallel on a high-speed multiprocessor system;

(b) The weighting matrices $E_i (i = 1, 2, \dots, \alpha)$ can be used to coordinate the computations of the overlapping and non-overlapping variables so that the distributions among the processors of the task are possibly well balanced;

(c) If some diagonal elements of the weighting matrix E_i are zero, then the corresponding elements of $x^{p,i}$ need not be computed. Hence, considerable savings on the computational workloads are possible;

(d) Suitable choices of the α splittings can result in α lower-dimensional systems, and therefore, each processor only needs to solve a system of linear equations of smaller size; and

(e) In practical implementations, suitable choices of the splittings and the weightings can greatly improve the convergence properties and the parallel efficiency of this method.

Method 1 is the original of the below reviewed synchronous, chaotic, and asynchronous multisplitting iterative methods for the large sparse linear complementarity problem (1).

Some Basic Properties of the Linear Complementarity Problem

Some useful notations and concepts:

$C = (c_{kj}) \in R^{n\times n}$: **an real** $n \times n$ **matrix, with its (k,j)-th element being** c_{kj} **(or** $(C)_{kj}$ **or** $[C]_{kj}$**)**

$\mathrm{diag}(C)$: **the** $n \times n$ **diagonal matrix coinciding in its diagonal with** C

$A \leq B$: **if** $a_{kj} \leq b_{kj}$

A **nonnegative: if** $a_{kj} \geq 0$

$\rho(A)$: **the spectral radius of the matrix** A

This definition carries immediately over to vectors by identifying them with $n \times 1$ **matrices**

$|A| = (|a_{kj}|) \in R^{n\times n}$: **the absolute value of the matrix** $A = (a_{kj}) \in R^{n\times n}$

$\langle A\rangle = (\langle a_{kj}\rangle) \in R^{n\times n}$: **the comparison matrix of** $A \in R^{n\times n}$**, where** $\langle a_{kk}\rangle = |a_{kk}|$ **and**

$$\langle a_{kj}\rangle = -|a_{kj}| \quad \textbf{for} \quad k \neq j$$

$A = (a_{kj}) \in R^{n \times n}$ is called

an L-matrix: if it satisfies $a_{kj} \le 0$ for $k \ne j$, and $a_{kk} > 0$

an M-matrix: if it is nonsingular with $a_{kj} \le 0$ for $k \ne j$, and with $A^{-1} \ge 0$

an H-matrix: if $\langle A \rangle$ is an M-matrix

an H_+-matrix: if A is an H-matrix with all diagonal elements being positive

If $M \in R^{n \times n}$ is an H_+-matrix, then the linear complementarity problem (1) has a unique solution for any $q \in R^n$

For a given matrix $M \in R^{n\times n}$, let $B, C \in R^{n\times n}$ be such that $M = B + C$. Then (B, C) is called a splitting of the matrix M.

The splitting (B, C) is called

a convergent splitting: if the spectral radius of the matrix $(B^{-1}C)$ is less than one

a weak regular splitting: if $B^{-1} \geq 0$ and $B^{-1}C \leq 0$

a regular splitting: if $B^{-1} \geq 0$ and $C \leq 0$

an M-splitting: if B is an M-matrix and $C \leq 0$

an H-splitting: if $\langle B\rangle - |C|$ is an M-matrix

an H-compatible splitting: if $\langle M\rangle = \langle B\rangle - |C|$

a Q-splitting: if B is a Q-matrix

In particular, the splitting (B, C) is called an H_+-splitting and H_+-compatible splitting if it is an H-splitting and H-compatible splitting, respectively, with B an H_+-matrix.

Let the splitting (B, C) be such that $\langle B\rangle$ is nonsingular, and let (F, G) be a matrix pair. Then we call the matrix pair (F, G) a majorizing pair of the splitting (B, C) if F is an H-matrix and it holds that $\langle B\rangle^{-1} \leq \langle F\rangle^{-1}$ and $|C| \leq |G|$. In such a case, we say that (B, C) is majorized by (F, G). Note that here (F, G) is not necessarily a splitting of the matrix M.

Evidently, if $M = B + C$ is an H-splitting, then M and B are H-matrices and

$$\rho(B^{-1}C) \leq \rho(\langle B\rangle^{-1}|C|) < 1;$$

if it is an H-compatible splitting and M is an H-matrix, then it is an H-splitting and thus convergent; and
if (F, G) is a majorizing pair of the splitting (B, C) such that $\langle F\rangle - |G|$ is an M-matrix, then (B, C) is a convergent splitting

More Notations:

$N := \{1, 2, \ldots, n\}$
$N_0 := \{0, 1, 2, \ldots\}$
$\Lambda := \{1, 2, \ldots, \alpha\}$
$J_i (i = 1, 2, \ldots, \alpha)$ is called a decomposition of the number set N, if $J_i (i = 1, 2, \ldots, \alpha)$ are nonempty subsets of N such that $\cup_{i=1}^{\alpha} J_i = N$

Synchronous Multisplitting Iterative Methods

**Based on several splittings of the system matrix $M \in R^{n\times n}$, and through weighting combination of the solutions of the linear complementarity problems resulted from these splittings,
Machida, Fukushima and Ibaraki and Bai
presented a class of synchronous multisplitting iterative method for solving the linear complementarity problem (1).**

This method is a technical extension of the matrix multisplitting method for the system of linear equations to the linear complementarity problem.

Method 2 (Machida, Fukushima and Ibaraki 1995 and Bai 1999).

Given a starting vector $z^0 \in R^n$.
For $p = 0, 1, 2, \ldots$ until $\{z^p\}_{p \in N_0}$ convergence, compute

$$z^{p+1} = \sum_{i=1}^{\alpha} E_i z^{p,i},$$

where, for each $i \in \Lambda$, $z^{p,i}$ is an arbitrary solution of the following linear complementarity problem LCP$(B_i, q_{p,i})$:

$$z \geq 0, \quad B_i z + q_{p,i} \geq 0 \quad \text{and} \quad z^T (B_i z + q_{p,i}) = 0,$$

with

$$q_{p,i} = C_i z^p + q.$$

Here, $(B_i, C_i, E_i)(i = 1, 2, \ldots, \alpha)$ is a multisplitting of the matrix $M \in R^{n \times n}$.

Note that the

$$\textbf{LCP}(B_i, q_{p,i})(i = 1, 2, \ldots, \alpha, p \in N_0)$$

may not have a solution. Hence, it is necessary for us to assume in this method that the splittings

$$M = B_i + C_i(i = 1, 2, \ldots, \alpha)$$

are Q-splittings, so that the iterative sequence $\{z^p\}_{p \in N_0}$ is well defined.

Analogous to Method 1, Method 2 has quite good parallel properties:

- **At every iteration step p, each of the sub-problems LCP$(B_i, q_{p,i})(i = 1, 2, \ldots, \alpha)$ is solved independently on one processor of the multiprocessor system, and therefore, Method 2 can be implemented in parallel.**
- **The α splitting matrices $B_i(i = 1, 2, \ldots, \alpha)$ and the α weighting matrices $E_i(i = 1, 2, \ldots, \alpha)$ may be chosen in such a way that the tasks distributed on the α processors of the multiprocessor system are evenly balanced so that Method 2 achieves high parallel efficiency.**
- **Considerable savings on the computational workloads are available because the entries of $z^{p,i}$ corresponding to the zero-diagonal elements of the weighting matrix E_i need not be computed.**

When the linear complementarity problem (1) is symmetric, we have the following convergence theorem for Method 2.

Theorem 1 ***Let*** $M \in R^{n\times n}$ ***be a symmetric matrix, and*** $(B_i, C_i, E_i)(i = 1, 2, \ldots, \alpha)$ ***be its multisplitting such that***

(a) $M = B_i + C_i (i = 1, 2, \ldots, \alpha)$ ***are Q-splittings;***

(b) $B_i - C_i (i = 1, 2, \ldots, \alpha)$ ***are positive definite matrices; and***

(c) $f(\sum_{i=1}^{\alpha} E_i z^{p,i}) \leq \max_{1\leq i\leq \alpha} f(z^{p,i})$***, where***

$$f(z) = \frac{1}{2} z^T M z + z^T q.$$

Assume that $f(z)$ ***is bounded below on*** $z \geq 0$***, and that***

$$0 \neq z \geq 0, \quad Mz \geq 0 \quad \textbf{\textit{and}} \quad z^T M z = 0$$
$$\Longrightarrow \quad q^T z > 0.$$

Then for any starting vector $z^0 \in R^n$***, the iterative sequence*** $\{z^p\}_{p\in N_0}$ ***generated by Method 2 is bounded and any accumulation point of it solves the linear complementarity problem (1).***

Theorem 1 does not require that the diagonal elements of the weighting matrices $E_i (i = 1, 2, \ldots, \alpha)$ be nonnegative.

Moreover, various choices of the weighting matrices $E_i (i = 1, 2, \ldots, \alpha)$ satisfy condition (c) in Theorem 1.

One of the possibilities is given by the choice of $E_i = \alpha_i I \ (i = 1, 2, \ldots, \alpha)$, where $\alpha_i (i = 1, 2, \ldots, \alpha)$ are nonnegative real numbers satisfying $\sum\limits_{i=1}^{\alpha} \alpha_i = 1$.

In this case, condition (c) is automatically satisfied if either of the following two classes of restrictions are further imposed:

(1) for a positive integer sequence $\{i_p\} \in \Lambda$,

$$\alpha_i = \begin{cases} 1, & \text{for } i = i_p, \\ 0, & \text{for } i \neq i_p, \end{cases} \quad i = 1, 2, \ldots, \alpha, \quad p \in N_0,$$

where the indices $i_p (p \in N_0)$ are chosen either randomly at every iteration, or in a certain predetermined order such as the cyclic rule, or based on the function values $f(z^{p,i}) (i = 1, 2, \ldots, \alpha)$ such that, for $p \in N_0$,

$$f(z^{p,i_p}) = \min_{1 \le i \le \alpha} f(z^{p,i});$$

and

(2) $M \in R^{n \times n}$ is a positive semidefinite matrix.

For the nonsymmetric linear complementarity problem and for general nonnegative diagonal weighting matrices, we have the following convergence theory for Method 2.

Theorem 2 ***Let*** $M \in R^{n\times n}$ ***be an*** H_+***-matrix, and*** $(B_i, C_i, E_i)(i = 1, 2, \ldots, \alpha)$ ***be its multisplitting such that***

(a) $B_i(i = 1, 2, \ldots, \alpha)$ ***are*** H_+***-matrices; and***

(b) for each i***, there exists a majorizing pair*** $(\widehat{B}_i, \widehat{C}_i)$ ***of the splitting*** (B_i, C_i)***, satisfying***

$$\widehat{H}_p u \le \theta u, \qquad \widehat{H}_p \equiv \sum_{i=1}^{\alpha} E_i \langle \widehat{B}_i \rangle^{-1} |\widehat{C}_i|, \qquad \forall p \in N_0, \tag{3}$$

for some nonnegative constant $\theta \in [0, 1)$ ***and some positive vector*** $u \in R^n$***.***

Then for any starting vector $z^0 \in R^n$***, the iterative sequence*** $\{z^p\}_{p\in N_0}$ ***generated by Method 2 converges to the unique solution of the linear complementarity problem (1).***

In this theorem, no restriction is imposed on the weighting matrices $E_i (i = 1, 2, \ldots, \alpha)$. Hence, we can choose them suitably such that Method 2 achieves high parallel computing efficiency.

The restriction on the matrix splittings $M = B_i + C_i (i = 1, 2, \ldots, \alpha)$ in Theorem 2 is quite different from that in Theorem 1.

However, both these classes of conditions naturally originate from the standard conditions for the parallel matrix multisplitting iterative methods of the systems of linear equations.

Two concrete choices of the multiple splittings satisfying condition (3) are given in the following examples.

Example 1 ***Assume that*** $M = (m_{kj}) \in R^{n\times n}$ ***is an*** H_+***-matrix, and***

$$D = diag(M), \quad B = M - D.$$

For a given positive constant $\beta \geq 1$***, let*** $D_i (i = 1, 2, \dots, \alpha)$ ***be positive diagonal matrices satisfying***

$$D \leq D_i \leq \beta D, \quad i = 1, 2, \dots, \alpha.$$

Take

$$B_i = D_i, \quad C_i = M - D_i, \quad i = 1, 2, \dots, \alpha, p = 0, 1, 2, \dots.$$

Then condition (3) is satisfied, because

$$\widehat{B}_i = D_i, \quad \widehat{C}_i = D_i - D + |B|, \qquad i = 1, 2, \dots, \alpha,$$

$u = \langle M \rangle^{-1} e$ ***with*** $e = (1, 1, \dots, 1)^T \in R^n$***, and***

$$\theta = \max\left\{0, 1 - \frac{1}{\max\limits_{1\leq k\leq n}\{\beta m_{kk} u_k\}}\right\}.$$

Example 2 ***Assume that*** $M = (m_{kj}) \in R^{n\times n}$ ***is an*** H_+***-matrix, and***

$$D = diag(M), \quad B = M - D.$$

For each $i \in \Lambda$***, take***

$$B_i = (b_{kj}^{(i)}) \in R^{n\times n} \quad \textbf{\textit{and}} \quad C_i = (c_{kj}^{(i)}) \in R^{n\times n}$$

in accordance with the following rule:

$$b_{kj}^{(i)} = \begin{cases} m_{kj}, & \textbf{\textit{for}}\ k = j, \\ b_{kj}^{(i)} \in [0, m_{kj}], & \textbf{\textit{for}}\ k \neq j\ \textbf{\textit{and}}\ m_{kj} \geq 0, \\ b_{kj}^{(i)} \in [m_{kj}, 0], & \textbf{\textit{for}}\ k \neq j\ \textbf{\textit{and}}\ m_{kj} < 0, \end{cases}$$

$$c_{kj}^{(i)} = \begin{cases} 0, & \textbf{\textit{for}}\ k = j, \\ m_{kj} - b_{kj}^{(i)}, & \textbf{\textit{for}}\ k \neq j. \end{cases}$$

Then condition (3) is satisfied, because

$$\widehat{B}_i = \langle B_i\rangle, \widehat{C}_i = |C_i|, \quad i = 1, 2, \ldots, \alpha, p = 0, 1, 2, \ldots,$$

$u = \langle M\rangle^{-1}e$ ***with*** $e = (1, 1, \ldots, 1)^T \in R^n$***, and***

$$\theta = \max\left\{0, 1 - \frac{1}{\max\limits_{1\leq k\leq n}\{m_{kk}u_k\}}\right\}.$$

The following result is a direct and concrete application of Theorem 2.

Theorem 3 ***Let*** $M \in R^{n\times n}$ ***be an*** H_+***-matrix, and*** $(B_i, C_i, E_i)(i = 1, 2, \ldots, \alpha)$ ***be its multisplitting such that***

(a) $B_i(i = 1, 2, \ldots, \alpha)$ ***are*** H_+***-matrices;***

(b) for each $i \in \Lambda$***, there exists a majorizing pair*** $(\widehat{B}_i, \widehat{C}_i)$ ***of the splitting*** (B_i, C_i)***, satisfying***

$$\begin{cases} \langle \widehat{B}_i \rangle - |\widehat{C}_i| \geq \widehat{M}_i, \\ u > 0, \widehat{M}_i u > 0 \\ \qquad \Longrightarrow \langle \widehat{B}_i \rangle u \leq \dfrac{1}{1-\theta} \widehat{M}_i u \textit{ for some } \theta \in [0, 1), \end{cases} \tag{4}$$

for some matrices $\widehat{M}_i \in R^{n\times n}(i = 1, 2, \ldots, \alpha)$***, for which there exists a positive vector*** $u \in R^n$ ***satisfying*** $\widehat{M}_i u > 0$***.***

Then for any starting vector $z^0 \in R^n$***, the iterative sequence*** $\{z^p\}_{p\in N_0}$ ***generated by Method 2 converges to the unique solution of the linear complementarity problem (1).***

Theorem 2 and Theorem 3 immediately give the following results.

Theorem 4 ***Let*** $M \in R^{n\times n}$ ***be an*** H_+***-matrix. Assume that for each*** $i \in \Lambda$***,*** $M = B_i + C_i$ ***is a splitting with*** B_i ***having positive diagonal elements, and there exist matrices*** $M_i \in R^{n\times n}(i = 1, 2, \ldots, \alpha)$***, a constant*** $\theta \in [0, 1)$ ***and a positive vector*** $u \in R^n$ ***such that*** $M_i u > 0$ ***and***

$$\langle B_i \rangle - |C_i| \geq M_i, \qquad \langle B_i \rangle u \leq \frac{1}{1-\theta} M_i u.$$

Then for any starting vector $z^0 \in R^n$***, the iterative sequence*** $\{z^p\}_{p\in N_0}$ ***generated by Method 2 converges to the unique solution of the linear complementarity problem (1).***

Theorem 5 ***Let*** $M \in R^{n\times n}$ ***be an*** H_+***-matrix. Assume that for each*** $i \in \Lambda$***,*** $M = B_i + C_i$ ***is an*** H_+***-compatible splitting. Then for any starting vector*** $z^0 \in R^n$***, the iterative sequence*** $\{z^p\}_{p\in N_0}$ ***generated by Method 2 converges to the unique solution of the linear complementarity problem (1).***

Chaotic Multisplitting Iterative Methods

The synchronous multisplitting iterative methods discussed in the last section are much profitable for solving the large sparse linear complementarity problems in parallel in the synchronous computing environments, since the α subproblems can be independently solved on the α processors of the multiprocessor system.

In actual computations, however, to start the next iterate, some of the processors may need to wait until all of them have completed their calculations of the current local iterates, in particular, when there is load imbalance or processing difference among the processors.

Hence, if the splittings and the weightings are chosen so that the workloads carried by all processors are exactly equally distributed, these synchronous multisplitting iterative methods can attain maximum efficiency.

When such a balance can be achieved, then the individual processors are ready to contribute towards their updates of the current global iterate at the same time, which, in turn, minimizes idle time.

Nevertheless, there are many applications that the concrete physical properties of the original problems lead to linear complementarity problems which are quite naturally divided into subproblems of unequal sizes.

Hence, the synchronous multisplitting iterative methods are intrinsically not efficient for solving this kind of linear complementarity problems.

To avoid loss of time and efficiency in processor utilization, we studied the chaotic variants of the aforementioned synchronous multisplitting methods based on the implicit splittings of the matrix, for which each processor can carry out a varying number of local iterations until a mutual phase time is reached when all processors are ready to contribute towards the global iteration.

Hence, the synchronous waits among different processors may be greatly decreased and the efficient numerical computation on each processor may be largely increased.

This, therefore, makes these chaotic multisplitting iterative methods be possible to achieve high parallel computing efficiency.

Method 3 (Bai 1999).
Given a starting vector $z^0 \in R^n$.
For $p = 0, 1, 2, \ldots$ until $\{z^p\}_{p \in N_0}$ convergence, compute

$$z^{p+1} = \sum_{i=1}^{\alpha} E_i z^{p,i},$$

where, for each $i \in \Lambda$,

$$z^{p,i,0} := z^p, \quad z^{p,i} := z^{p,i,\mu(p,i)},$$

and for $k = 1, 2, \ldots, \mu(p,i)$, $z^{p,i,k}$ is an arbitrary solution of the linear complementarity problem LCP$(B_i, q_{p,i,k})$:

$$z \geq 0, \qquad B_i z + q_{p,i,k} \geq 0 \quad \text{and} \quad z^T(B_i z + q_{p,i,k}) = 0,$$

with

$$q_{p,i,k} = C_i z^{p,i,k-1} + q$$

and $\mu(p,i)$ the composite numbers.

To ensure the existence of the solution of LCP$(B_i, q_{p,i,k})$ **so that the iterative sequence** $\{z^p\}_{p \in N_0}$ **is well defined, we need to assume in Method 3 that the splittings** $M = B_i + C_i (i = 1, 2, \ldots, \alpha)$ **are Q-splittings.**

The composite numbers $\mu(p,i)$ **in Method 3 can be determined either precedently before the starting of the iteration, or dynamically during the implementation of the method.**

If $\mu(p,i) = 1$ $(i \in \Lambda, p \in N_0)$**, then Method 3 reduces to Method 2.**

The following convergence theorems hold for Method 3 for the cases that the system matrix $M \in R^{n\times n}$ is symmetric and nonsymmetric, respectively.

Theorem 6 ***Let*** $M \in R^{n\times n}$ ***be a symmetric matrix, and*** $(B_i, C_i, E_i)(i = 1, 2, \ldots, \alpha)$ ***be its multisplitting such that***

(a) $M = B_i + C_i (i = 1, 2, \ldots, \alpha)$ ***are Q-splittings;***

(b) $B_i - C_i (i = 1, 2, \ldots, \alpha)$ ***are positive definite matrices; and***

(c) $f(\sum_{i=1}^{\alpha} E_i z^{p,i}) \le \max_{1\le i\le\alpha} f(z^{p,i})$***, where***

$$f(z) = \frac{1}{2} z^T M z + z^T q.$$

Assume that $f(z)$ ***is bounded below on*** $z \ge 0$***, and that***

$$0 \ne z \ge 0, \quad Mz \ge 0 \quad \textit{and} \quad z^T M z = 0$$
$$\Longrightarrow q^T z > 0.$$

Then for any starting vector $z^0 \in R^n$***, independently of the positive integer sequences*** $\{\mu(p,i)\}_{p\in N_0}$ $(i = 1, 2, \ldots, \alpha)$***, the iterative sequence*** $\{z^p\}_{p\in N_0}$ ***generated by Method 3 is bounded and any accumulation point of it solves the linear complementarity problem (1).***

Theorem 7 ***Let*** $M \in R^{n\times n}$ ***be an*** H_+***-matrix.***

Assume that for each $i \in \Lambda$***,*** $M = B_i + C_i$ ***is an*** H_+***-compatible splitting, satisfying***

$$diag(B_i) \leq \beta \ diag(M)$$

for some positive constant $\beta \geq 1$***.***

Then for any starting vector $z^0 \in R^n$***, the iterative sequence*** $\{z^p\}_{p\in N_0}$ ***generated by Method 3 converges, independently of the positive integer sequences*** $\{\mu(p,i)\}_{p\in N_0}$ $(i = 1, 2, \ldots, \alpha)$***, to the unique solution of the linear complementarity problem (1).***

Z. Z. Bai

Asynchronous Multisplitting Iterative Methods

Due to the intrinsic superiority of the parallel multiprocessor system to the single computer system, asynchronous iterative methods are much more efficient than their synchronous and chaotic counterparts, in particular, when there are load imbalances among the tasks distributed on the processors.

To this end, in accordance with the principle of using sufficiently the available information, exchanging flexibly the current message, increasing largely the useful computation and decreasing possibly the useless communication, we studied various asynchronous multisplitting iterative methods for solving the large sparse linear complementarity problems on the high-speed multiprocessor systems.

These asynchronous multisplitting iterative methods not only avoid simultaneous waits among the processors, but also exploit the parallel efficiency of the multiprocessor system.

Therefore, they are quite suitable for high-performance computing in actual applications.

The descriptions of the asynchronous multisplitting iterations need the following elementary notations.

(a) for $p \in N_0$, $J(p)$ is used to denote a nonempty subset of the number set Λ;

(b) for $i \in \Lambda$, $\{s_i(p)\}_{p\in N_0}$ are used to represent α infinite nonnegative integer sequences, with

$$s_i(p) \in N_0, \quad \forall i \in \Lambda \quad \textbf{and} \quad \forall p \in N_0.$$

We assume that the subsets $J(p)$ and the sequences $\{s_i(p)\}_{p\in N_0}(i \in \Lambda)$ satisfy the following properties:

(1) for $i \in \Lambda$, the sets $\{p \in N_0 : i \in J(p)\}$ are infinite;

(2) for $i \in \Lambda$ and $p \in N_0$, $s_i(p) \le p$ hold; and

(3) for $i \in \Lambda$, $\lim_{p\to\infty} s_i(p) = \infty$ hold.

If we define

$$s(p) = \min_{i\in\Lambda}\{s_i(p)\},$$

then it obviously holds that

$$s(p) \le p \quad \textbf{and} \quad \lim_{p\to\infty} s(p) = \infty.$$

The above-described synchronous and chaotic multisplitting methods can achieve high parallel computing efficiency provided the task is evenly distributed onto all processors.

However, such an assumption of the balanced distribution does not always hold in many applications, probably due to some special physical properties of the original problem.

To exploit the parallel computing efficiency of the multiprocessor system as far as possible, we presented a class of asynchronous multisplitting relaxation methods for solving the linear complementarity problem (1).

They are quite suitable for implementing in the asynchronous parallel computing environments, since no mutual wait among the processors is demanded.

Method 4 (Bai and Huang 1999).
Given a starting vector $z^0 \in R^n$.
For $p = 0, 1, 2, \ldots$ until $\{z^p\}_{p \in N_0}$ convergence, compute

$$z^{p+1} = \sum_{i=1}^{\alpha} E_i z^{p,i},$$

where, for each $i \in \Lambda$,

$$z^{p,i} = \begin{cases} \omega \hat{z}^{p,i} + (1-\omega) z^{s_i(p)}, & \text{for } i \in J(p), \\ z^p, & \text{for } i \notin J(p), \end{cases} \quad i = 1, 2, \ldots, \alpha,$$

and $\hat{z}^{p,i}$ is an arbitrary solution of the linear complementarity problem LCP$(B_i, q_{p,i})$:

$$z \geq 0, \quad B_i z + q_{p,i} \geq 0 \quad \text{and} \quad z^T (B_i z + q_{p,i}) = 0,$$

with

$$q_{p,i} = C_i z^{s_i(p)} + q.$$

Here, $\omega \in (0, \infty)$ is a relaxation factor, and $(B_i, C_i, E_i)(i = 1, 2, \ldots, \alpha)$ is a multisplitting of the matrix $M \in R^{n \times n}$.

In Method 4, each processor is allowed to update or retrieve the global approximate solution residing in the host processor at any time.

Hence, currently available information can be used promptly to renew the dated one so that Method 4 achieves high parallel computing efficiency in actual applications.

On the other hand, If $J(p) = \Lambda$ and $s_i(p) = p$ hold for $p \in N_0$ and $i \in \Lambda$, then Method 4 reduces to Method 2 when $\omega = 1$, and it gives an extrapolated variant of Method 2 when $\omega \neq 1$.

We have the following asymptotic convergence theorem for Method 4.

Theorem 8 ***Let*** $M \in R^{n\times n}$ ***be an*** H_+***-matrix,***

$$D = diag(M) \quad \textbf{\textit{and}} \quad B = M - D.$$

Assume that $(B_i, C_i, E_i)(i = 1, 2, \ldots, \alpha)$ ***is a multisplitting of the matrix*** M***, and for each*** $i \in \Lambda$***,*** $M = B_i + C_i$ ***is an*** H_+***-compatible splitting satisfying*** $diag(B_i) = diag(M)$***.***

Then for any starting vector $z^0 \in R^n$***, the iterative sequence*** $\{z^p\}_{p\in N_0}$ ***generated by Method 4 converges to the unique solution of the linear complementarity problem (1), provided the relaxation parameter*** ω ***satisfies***

$$0 < \omega < \frac{2}{1 + \rho(D^{-1}|B|)}.$$

Multi-parameter Generalizations

The above-described methods are implicit iterative methods, in the sense that they need the solutions of α linear complementarity problems of smaller sizes. This makes these methods less convenient in actual computations.

The introduction of relaxation parameters not only can produce various applicable iterative methods, but also can improve the convergence properties of these methods as well.

If a multisplitting method is constructed by using one forward and one backward relaxation sweeps, and if each processor is allowed to update or retrive any piece of the global iterate residing in the host processor at any time, then we can formulate multi-parameter relaxed synchronous, chaotic, and asynchronous multisplitting methods for solving the linear complementarity problem (1).

In this section, we will emphatically discuss the multi-parameter relaxed asynchronous multisplitting iterative methods and their convergence theories.

The synchronous and the chaotic multisplitting iterative methods can be treated as special cases.

We first give the concept of the generalized block triangular multisplitting.

For a given matrix $M = (m_{kj}) \in R^{n\times n}$**, we denote** $D = \mathrm{diag}(M)$**. For** $i = 1, 2, \ldots, \alpha$**, let** $L_i \in R^{n\times n}$ **and** $U_i \in R^{n\times n}$ **be strictly lower-triangular and strictly upper-triangular matrices, respectively,** $W_i \in R^{n\times n}$ **be zero-diagonal matrices,** $E_i \in R^{n\times n}$ **be nonnegative diagonal matrices, and** $J_i (i = 1, 2, \ldots, \alpha)$ **be a decomposition of the number set** N**, such that they have the following particular structures:**

$$\begin{cases} L_i = ([L_i]_{kj}), \quad [L_i]_{kj} = \begin{cases} l_{kj}^{(i)}, & \text{for } k, j \in J_i \text{ and } k > j, \\ 0, & \text{otherwise,} \end{cases} \\ U_i = ([U_i]_{kj}), \quad [U_i]_{kj} = \begin{cases} u_{kj}^{(i)}, & \text{for } k, j \in J_i \text{ and } k < j, \\ 0, & \text{otherwise,} \end{cases} \\ W_i = ([W_i]_{kj}), \quad [W_i]_{kj} = \begin{cases} 0, & \text{for } k = j, \\ w_{kj}^{(i)}, & \text{otherwise,} \end{cases} \\ E_i = \mathrm{diag}([E_i]_{kk}), \quad [E_i]_{kk} = \begin{cases} e_k^{(i)} \geq 0, & \text{for } k \in J_i, \\ 0, & \text{otherwise,} \end{cases} \end{cases}$$

and satisfy the following properties:

(a) D **is nonsingular, i.e.,** $\det(D) \neq 0$;

(b) $M = D + L_i + U_i + W_i, \quad i = 1, 2, \ldots, \alpha$; **and**

(c) $\sum_{i=1}^{\alpha} E_i = I$ **(the** $n \times n$ **identity matrix),**

Then the collection $(D+L_i, D+U_i, W_i, E_i)(i = 1, 2, \ldots, \alpha)$

is called a generalized block triangular multisplitting of the matrix $M \in R^{n\times n}$.

The following notations are elementary for the description of the asynchronous multisplitting iteration.

(a) for $i \in \Lambda$ and $p \in N_0$, $J_i(p)$ is used to denote a subset of the number set J_i;

(b) for $k \in N$ and $p \in N_0$, denote

$$N_k(p) = \{i : k \in J_i(p), \quad i = 1, 2, \dots, \alpha\};$$

and

(c) for $i \in \Lambda$ and $k \in N$, $\{s_k^{(i)}(p)\}_{p\in N_0}$ are used to represent infinite nonnegative integer sequences, with

$$s_k^{(i)}(p) \in N_0, \quad \textbf{for} \quad i \in \Lambda \quad \textbf{and} \quad p \in N_0.$$

We assume that the subsets $J_i(p)$ and the sequences $\{s_k^{(i)}(p)\}_{p\in N_0}$ satisfy the following properties:

(1) for $i \in \Lambda$ and $k \in N$, the sets $\{p \in N_0 : k \in J_i(p)\}$ are infinite;

(2) for $p \in N_0$, $\cup_{i=1}^{\alpha} J_i(p) \neq \emptyset$ hold;

(3) for $i \in \Lambda$, $k \in N$ and $p \in N_0$, $s_k^{(i)}(p) \le p$ hold; and

(4) for $i \in \Lambda$ and $k \in N$, $\lim\limits_{p\to\infty} s_k^{(i)}(p) = \infty$ hold.

If we define

$$s(p) = \min_{i\in\Lambda} \min_{k\in N} \{s_k^{(i)}(p)\},$$

then it obviously holds that

$$s(p) \leq p \quad \textbf{and} \quad \lim_{p\to\infty} s(p) = \infty.$$

Note the difference between these notations and the ones introduced at the beginning of the last Section.

Here:

α **is the number of the processors**

J_i **is the subset of variables assigned to processor** i

$J_i(p)$ **is the subset of variables updated by processor** i **at iteration** p

$N_k(p)$ **is the subset of processors updating the k-th entry of the global variable at iteration** p

$s_k^{(i)}(p)$ **is the iteration index for the k-th entry of the global variable received by processor i from the host processor at iteration** p**.**

Method 5 (Bai and Huang 1999).
Given a starting vector $z^0 \in R^n$.
For $p = 0, 1, 2, \ldots$ until $\{z^p\}_{p \in N_0}$ convergence, successively compute

$$[z^{p+1}]_k = \sum_{i \in N_k(p)} e_k^{(i)} [z^{p+1,i}]_k + \sum_{i \notin N_k(p)} e_k^{(i)} [z^p]_k, \qquad k = 1, 2, \ldots, n,$$

where $z^{p+1,i} (i \in \Lambda)$ are determined by the following two relaxation sweeps:

(I) The forward relaxation sweep:

Successively compute $[z^{p+1/2,i}]_k (k \in J_i(p))$ element by element through

$$[z^{p+1/2,i}]_k = \begin{cases} 0, & \text{if } \gamma_1 \sum_{j=1}^{k-1} l_{kj}^{(i)} ([z^{p+1/2,i}]_j - [z^{s^{(i)}(p)}]_j) \\ & \quad + \omega_1 [M z^{s^{(i)}(p)} + q]_k \\ & \quad > m_{kk} [z^{s^{(i)}(p)}]_k, \\ [z^{s^{(i)}(p)}]_k + \dfrac{\gamma_1}{m_{kk}} \sum_{j=1}^{k-1} l_{kj}^{(i)} ([z^{s^{(i)}(p)}]_j - [z^{p+1/2,i}]_j) \\ \quad - \dfrac{\omega_1}{m_{kk}} [M z^{s^{(i)}(p)} + q]_k, & \\ \quad \text{otherwise.} & \end{cases}$$

(II) The backward relaxation sweep:

Successively compute $[z^{p+1,i}]_k (k \in J_i(p))$ element by element through

$$[z^{p+1,i}]_k = \begin{cases} 0, & \text{if } \gamma_2 \sum_{j=1}^{k-1} u_{kj}^{(i)} ([z^{p+1,i}]_j - [z^{p+1/2,i}]_j) \\ & \quad +\omega_2 [M z^{p+1/2,i} + q]_k \\ & \quad > m_{kk} [z^{p+1/2,i}]_k, \\ [z^{p+1/2,i}]_k + \frac{\gamma_2}{m_{kk}} \sum_{j=1}^{k-1} u_{kj}^{(i)} ([z^{p+1/2,i}]_j - [z^{p+1,i}]_j) \\ \quad -\frac{\omega_2}{m_{kk}} [M z^{p+1,i} + q]_k, & \\ & \text{otherwise,} \end{cases}$$

with

$$z^{s^{(i)}(p)} = \left(z_1^{s_1^{(i)}(p)}, z_2^{s_2^{(i)}(p)}, \ldots, z_n^{s_n^{(i)}(p)} \right)^T, \quad \text{for } i \in \Lambda \text{ and } p \in N_0.$$

Here, $\gamma_j \in [0, \infty) (j = 1, 2)$ are relaxation factors, and $\omega_j \in (0, \infty) (j = 1, 2)$ are acceleration factors.

We have the following asymptotic convergence theorem for Method 5.

Theorem 9 ***Assume that*** $M \in R^{n\times n}$ ***is an*** H_+***-matrix,***

$$D = diag(M) \quad \textbf{\textit{and}} \quad B = M - D.$$

Let

$$(D + L_i, D + U_i, W_i, E_i)(i = 1, 2, \ldots, \alpha)$$

be a generalized block triangular multisplitting of the matrix $M \in R^{n\times n}$***, with***

$$\langle M\rangle = D - |L_i| - |U_i| - |W_i|, \qquad i = 1, 2, \ldots, \alpha.$$

Then the iterative sequence $\{z^p\}_{p\in N_0}$ ***generated by Method 5 converges to the unique solution of the linear complementarity problem (1), provided the relaxation parameters*** γ_k ***and*** ω_k***,*** $k = 1, 2$***, satisfy***

$$0 \leq \gamma_k \leq \omega_k \quad \textbf{\textit{and}} \quad 0 < \omega_k < \frac{2}{1 + \rho(D^{-1}|B|)}, \quad k = 1, 2.$$

Numerical Experiments

We remark that Method 5 is quite suitable for both the tightly coupled multiprocessor and the multi-computer having a shared global memory.

Method 5, its synchronous and sequential counterparts, together with some of their typical cases resulted from special choices of the relaxation parameters, will be the experiment methods of this paper.

The Experimental Problems

We consider the following three practical problems of the LCP(M, q).

Problem 1 ***The linear complementarity problem with the system matrix*** $M \in \mathbb{R}^{n\times n}$ ***corresponding to the Laplacian 5-point finite difference operator:***

$$M = \begin{pmatrix} T & -I & & & \\ -I & T & -I & & \\ & \ddots & \ddots & \ddots & \\ & & -I & T & -I \\ & & & -I & T \end{pmatrix} \in R^{\tilde{n}\times\tilde{n}},$$

$$T = \begin{pmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 4 & -1 \\ & & & -1 & 4 \end{pmatrix} \in R^{\tilde{n}\times\tilde{n}},$$

and the known vector $q \in R^n$ ***suitably chosen, e.g.,***

$$q = (1, -1, \ldots, (-1)^{n-1}, (-1)^n)^T \in R^n,$$

where $n = \tilde{n}^2$***.***

Note that $M \in R^{n\times n}$ ***is an*** H_+***-matrix.***

Therefore, the LCP(M, q) ***has a unique solution.***

Problem 2 ***The linear complementarity problem with the system matrix $M \in R^{n\times n}$ corresponding to the Journal Bearing finite difference operator:***

$$M = \begin{pmatrix} T_1 & -\beta_1 I & & & \\ -\beta_1 I & T_2 & -\beta_2 I & & \\ & \ddots & \ddots & \ddots & \\ & & -\beta_{\tilde{n}-2} I & T_{\tilde{n}-1} & -\beta_{\tilde{n}-1} I \\ & & & -\beta_{\tilde{n}-1} I & T_{\tilde{n}} \end{pmatrix} \in R^{\tilde{n}\times\tilde{n}},$$

$$T_k = \begin{pmatrix} \eta_k+\eta_1 & -\beta_1 & & & \\ -\beta_1 & \eta_k+\eta_2 & -\beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & -\beta_{\tilde{n}-2} & \eta_k+\eta_{\tilde{n}-1} & -\beta_{\tilde{n}-1} \\ & & & -\beta_{\tilde{n}-1} & \eta_k+\eta_{\tilde{n}} \end{pmatrix} \in R^{\tilde{n}\times\tilde{n}},$$

and the known vector $q \in R^n$ suitably chosen, e.g.,

$$q = (50\pi, 50\pi, \ldots, 50\pi)^T \in R^n,$$

where π denotes the ratio of the circumference of a circle to its diameter,

$$\beta_k = (k+\frac{1}{2})^3, \quad \eta_k = (k+\frac{1}{2})^3+(k-\frac{1}{2})^3, \qquad k = 1, 2, \ldots, \tilde{n}.$$

Note that $M \in R^{n\times n}$ is a K-matrix.

Therefore, the LCP(M, q) has a unique solution.

Problem 3 ***The linear complementarity problem with the system matrix*** $M \in R^{n\times n}$ ***corresponding to the Laplacian 9-point finite difference operator:***

$$M = \begin{pmatrix} T & S & & & \\ S & T & S & & \\ & \ddots & \ddots & \ddots & \\ & & S & T & S \\ & & & S & T \end{pmatrix} \in R^{\tilde{n}\times\tilde{n}},$$

$$T = \begin{pmatrix} 20 & -4 & & & \\ -4 & 20 & -4 & & \\ & \ddots & \ddots & \ddots & \\ & & -4 & 20 & -4 \\ & & & -4 & 20 \end{pmatrix} \in R^{\tilde{n}\times\tilde{n}},$$

$$S = \begin{pmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 4 & -1 \\ & & & -1 & 4 \end{pmatrix} \in R^{\tilde{n}\times\tilde{n}},$$

and the known vector $q \in R^n$ ***suitably chosen, e.g.,***

$$q = (1, -1, \ldots, (-1)^{\tilde{n}-1}, (-1)^{\tilde{n}})^T \in R^n.$$

Note that $M \in R^{n\times n}$ ***is an*** H_+***-matrix.***

Therefore, the LCP(M, q) ***has a unique solution.***

We remark that the finite difference discretizations at equidistant grids of a free boundary value problem about the flow of water through a porous dam may result in linear complementarity problems of the types of Problem 1 and Problem 3, and that of a journal bearing problem may result in linear complementarity problems of the type of Problem 2.

The Experimental Environment

We run our programs on an SGI Power Challenge multiprocessor computer as PVM applications.

This parallel machine consists of four 75 MHz TFP 64-bit RISC processors.

These CMOS processors each delivers a peak theoretical performance of 0.3 GFLOPS.

The data cache size is 16 Kbytes.

The Experimental Methods

The tested methods in our numerical experiments are listed in the following three tables:

(a) the sequential relaxation methods:

Method	γ_1	ω_1	γ_2	ω_2	Description
SOR	ω	ω	0	0	the successive overrelaxation method
SSOR	ω	ω	ω	ω	the symmetric SOR method
USOR	γ	γ	ω	ω	the unsymmetric SOR method
AOR	γ	ω	0	0	the accelerated overrelaxation method
SAOR	γ	ω	γ	ω	the symmetric AOR method

(b) the synchronous multisplitting relaxation methods:

Method	γ_1	ω_1	γ_2	ω_2	Description
MSOR	ω	ω	0	0	the multisplitting SOR method
MSSOR	ω	ω	ω	ω	the multisplitting SSOR method
MUSOR	γ	γ	ω	ω	the multisplitting USOR method
MAOR	γ	ω	0	0	the multisplitting AOR method
MSAOR	γ	ω	γ	ω	the multisplitting SAOR method

(c) the asynchronous multisplitting relaxation methods:

Method	γ_1	ω_1	γ_2	ω_2	Description
AMSOR	ω	ω	0	0	the asynchronous MSOR method
AMSSOR	ω	ω	ω	ω	the asynchronous MSSOR method
AMUSOR	γ	γ	ω	ω	the asynchronous MUSOR method
AMAOR	γ	ω	0	0	the asynchronous MAOR method
AMSAOR	γ	ω	γ	ω	the asynchronous MSAOR method

For the convenient of application and without loss of generality, the block triangular multisplittings $(D+L_{p,i}, D+U_{p,i}, W_{p,i}, E_i)(i=1,2,\ldots,\alpha)$, $p=0,1,2,\ldots$, are now chosen to be stationary ones (i.e., they are independent of the iterate index p), and they have the following structures:

$$\begin{array}{lclcl}
L_{p,i} & = & (\mathcal{L}_{kj}^{(p,i)}), & \mathcal{L}_{kj}^{(p,i)} & = \begin{cases} m_{kj}, & \textbf{for } k,j\in J_i \textbf{ and } k>j, \\ 0, & \textbf{otherwise,} \end{cases} \\
U_{p,i} & = & (\mathcal{U}_{kj}^{(p,i)}), & \mathcal{U}_{kj}^{(p,i)} & = \begin{cases} m_{kj}, & \textbf{for } k,j\in J_i \textbf{ and } k<j, \\ 0, & \textbf{otherwise,} \end{cases} \\
W_{p,i} & = & (\mathcal{W}_{kj}^{(p,i)}), & \mathcal{W}_{kj}^{(p,i)} & = \begin{cases} 0, & \textbf{for } k=j, \\ 0, & \textbf{for } k,j\in J_i, \\ m_{kj}, & \textbf{otherwise,} \end{cases} \\
E_i & = & diag(e_k^{(i)}), & e_k^{(i)} & = \begin{cases} 1, & 1\le k\le \tilde{n}_1\tilde{n}, \quad i=1, \\ 0.5, & \tilde{n}_{i-1}\tilde{n}+1\le k\le \tilde{n}_i\tilde{n}, \\ & \qquad 2\le i\le\alpha, \\ 0.5, & \tilde{n}_i\tilde{n}+1\le k\le \tilde{n}_{i+1}\tilde{n}, \\ & \qquad 1\le i\le\alpha-1, \\ 1, & \tilde{n}_\alpha\tilde{n}+1\le k\le n, \quad i=\alpha, \end{cases}
\end{array}$$

where for $i=1,2,\ldots,\alpha$, $\tilde{n}_i = \textbf{Int}\left(\frac{i\tilde{n}}{\alpha+1}\right)$, and

$$J_i = \{\tilde{n}_{i-1}\tilde{n}+1, \tilde{n}_{i-1}\tilde{n}+2, \ldots, \tilde{n}_{i+1}\tilde{n}\}.$$

Here, Int$(\bullet)$ denotes the integer part of the corresponding real number.

The Starting and the Stopping Criterions

All the computations are started from an initial vector having all components equal to 40.0, and terminated once the current iterations z^p obey

$$\frac{|(z^p)^T(Mz^p+q)|}{|(z^0)^T(Mz^0+q)|} \leq 10^{-7}.$$

Numerical Results

The three problems of the LCP(M,q) of various sizes are tested by the sequential, the synchronous and the asynchronous relaxation methods in Tables (a)-(c) when the processor number α ranges from $\{1,2,3,4\}$, respectively.

For $n = 4900$ and $\alpha = 3$, some of the representative numerical results are listed in the following tables and depicted by the following figures.

We use CPU to denote the CPU time (in seconds) required for an iteration to reach the above stopping criterion, ∞ to denote that an iteration does not satisfy the stopping criterion after 5000 iterations, and SP to denote the speed-up of a parallel execution, which is defined to be the ratio of the CPU times of the sequential to the corresponding parallel runnings.

Performance of Example 1

ω	**0.6**	**0.7**	**0.9**	**1.1**	**1.3**	**1.5**
SOR	**91.92**	**73.14**	**48.09**	**32.51**	**21.38**	**13.23**
SSOR	**45.98**	**37.03**	**24.27**	**16.26**	**10.72**	∞

Tabel (A_1): CPUs for the sequential SOR-like methods

ω		0.6	0.7	0.9	1.1	1.3	1.5
MSOR	CPU	48.12	37.70	24.92	16.82	11.10	6.97
	SP	1.91	1.94	1.93	1.93	1.93	1.90
MSSOR	CPU	23.23	19.06	12.55	8.47	5.62	∞
	SP	1.98	1.94	1.93	1.92	1.91	-

Tabel (A_2): CPUs and SPs for the multisplitting SOR-like methods

ω		0.6	0.7	0.9	1.1	1.3	1.5
AMSOR	CPU	42.41	33.59	22.09	14.99	10.04	6.34
	SP	2.17	2.18	2.18	2.17	2.13	2.09
AMSSOR	CPU	21.30	16.80	11.26	7.70	5.12	∞
	SP	2.16	2.20	2.16	2.11	2.09	-

Tabel (A_3): CPUs and SPs for the asynchronous multisplitting SOR-like methods

Tables (A_2) and Tables (A_3) show that the asynchronous multisplitting SOR and SSOR methods outperform the synchronous multisplitting SOR and SSOR methods, and the multisplitting SSOR-like methods outperform the corresponding multisplitting SOR-like methods, respectively, in terms of both elapsed time and parallel speed-up.

γ	0.4	0.6	0.8	1.0	1.2	1.4	1.6	1.7	1.8	1.9
ω	1.0	1.0	1.0	1.1	1.1	1.1	1.2	1.2	1.2	1.2
AOR	63.03	54.90	47.08	35.65	28.53	21.42	13.06	9.79	6.50	3.03
SAOR	29.69	26.03	22.97	17.79	14.87	11.42	7.21	5.59	3.91	2.04
USOR	39.31	31.74	25.37	18.41	13.94	9.93	∞	∞	∞	∞

Tabel (A_4): CPUs for the sequential AOR-like methods

γ		**0.4**	**0.6**	**0.8**	**1.0**	**1.2**	**1.4**	**1.6**	**1.7**	**1.8**	**1.9**
ω		**1.0**	**1.0**	**1.0**	**1.1**	**1.1**	**1.1**	**1.2**	**1.2**	**1.2**	**1.2**
MAOR	CPU	**33.17**	**28.74**	**24.52**	**18.59**	**15.00**	**11.35**	**7.04**	**5.43**	**3.68**	**2.00**
	SP	**1.90**	**1.91**	**1.92**	**1.92**	**1.90**	**1.89**	**1.86**	**1.80**	**1.77**	**1.52**
MSAOR	CPU	**15.14**	**13.59**	**11.97**	**9.48**	**7.31**	**6.16**	**4.01**	**3.12**	**2.21**	**1.29**
	SP	**1.96**	**1.92**	**1.92**	**1.88**	**2.03**	**1.85**	**1.80**	**1.79**	**1.77**	**1.58**
MUSOR	CPU	**20.45**	**16.53**	**13.19**	**9.76**	**7.46**	**5.25**	∞	∞	∞	∞
	SP	**1.92**	**1.92**	**1.92**	**1.89**	**1.87**	**1.89**	-	-	-	-

Tabel (A_5): CPUs and SPs for the multisplitting AOR-like methods

γ		**0.4**	**0.6**	**0.8**	**1.0**	**1.2**	**1.4**	**1.6**	**1.7**	**1.8**	**1.9**
ω		**1.0**	**1.0**	**1.0**	**1.1**	**1.1**	**1.1**	**1.2**	**1.2**	**1.2**	**1.2**
AMAOR	**CPU**	**28.80**	**25.16**	**21.91**	**16.53**	**13.35**	**10.25**	**6.30**	**4.84**	**3.30**	**1.82**
	SP	**2.19**	**2.18**	**2.15**	**2.16**	**2.14**	**2.09**	**2.07**	**2.02**	**1.97**	**1.66**
AMSAOR	**CPU**	**13.64**	**12.14**	**10.72**	**8.40**	**6.86**	**5.47**	**3.52**	**2.72**	**1.96**	**1.19**
	SP	**2.18**	**2.14**	**2.14**	**2.12**	**2.17**	**2.09**	**2.08**	**2.06**	**1.99**	**1.71**
AMUSOR	**CPU**	**18.15**	**14.67**	**11.78**	**8.73**	**6.63**	**4.76**	∞	∞	∞	∞
	SP	**2.17**	**2.16**	**2.15**	**2.11**	**2.10**	**2.09**	**-**	**-**	**-**	**-**

Tabel (A_6): CPUs and SPs for the asynchronous multisplitting AOR-like methods

Tables (A_4)-(A_6) show that the asynchronous multisplitting AOR, SAOR and USOR methods outperform the synchronous multisplitting AOR, SAOR and USOR methods, respectively, and the two-sweep relaxed multisplitting methods outperform the one-sweep relaxed multisplitting methods, correspondingly, in terms of both elapsed time and parallel speed-up.

Moreover, the two-parameter relaxed multisplitting AOR-like methods have larger convergence domains than the corresponding two-parameter relaxed multisplitting SOR-like methods.

In Figures (A_1)-(A_3), we give the behaviours of the asynchronous multisplitting AOR, SAOR and USOR methods, respectively.

The r and w axes in each figure correspond to the γ and ω axes, respectively.

It is clearly demonstrated that all these methods have good convergence properties over a wide range of the relaxation parameters.

Performance of Example 2

The computing results of this problem are depicted by Figures (B_1)-(B_7).

In these figures, the x-axis corresponds to the relaxation parameter, and the y-axis corresponds to the CPU in logarithmic scale.

Note that here we only investigate a representative curve from the two-dimensional surface of the CPU vs. the relaxation parameters γ and ω for the two-parameter relaxed methods, which was cut off by the hyperplane $\omega = 1.3$.

Obviously, Figures $(B_1)-(B_3)$ show that:

the AMAOR method always outperform the MAOR method, and the MAOR method almost always outperform the AOR method except for the points nearby the optimum.

Note that the AMAOR method has larger convergence domain than the MAOR method.

The AMSAOR method always outperform the MSAOR method, and the MSAOR method almost always outperform the SAOR method except for the points nearby the optimum.

That both synchronous and asynchronous multisplitting relaxation methods perform worse than the corresponding sequential relaxation method in the nearby of the optimal points of the relaxation parameter is probably because the convergence properties of the parallel methods are not better than the sequential method when the relaxation parameters are much close to the optimal values.

Evidently, Figures $(B_4) - (B_7)$ show that:

the MSOR method always outperform the SOR method, and its convergence property is quite comparable with the AMSOR method, except for the points nearby the optimum.

Around the optimal point, the MSOR method has the best numerical behaviour, while the AMSOR method has the worst one.

However, the AMSOR method has larger convergence domain than both MSOR amd SOR methods.

Figure (B_6) depicts the performance of the SSOR, MSSOR and AMSSOR methods.

The MSSOR method always outperform the SSOR method, and its convergence property is quite comparable with the AMSSOR method, except for the points nearby the optimum.

Around the optimal point, the MSSOR method has the best numerical behaviour, while the SSOR method has the worst one.

However, the AMSSOR method has larger convergence domain than both MSSOR and SSOR methods.

Hence, the asynchronous multisplitting relaxation methods almost always outperform the synchronous multisplitting relaxation methods and the sequential relaxation methods.

That the convergence speeds of these three classes of methods are quite different in the nearby of the optimums of the relaxation parameters is probably because the drastic difference of the convergence properties of these methods when the relaxation parameters are close to the optimums.

Performance of Example 3

The computing results of this problem are depicted by Figures (C_1)-(C_7).

In these figures, the x-axis corresponds to the relaxation parameter, and the y-axis corresponds to the CPU in logarithmic scale.

Again, note that here we only investigate a representative curve from the two-dimensional surface of the CPU vs. the relaxation parameters γ and ω for the two-parameter relaxed methods, which was cut off by the hyperplane $\omega = 1.3$.

In Figures $(C_1)-(C_3)$, the classes of USOR, AOR and SAOR methods were compared in both sequential and parallel settings, and in Figures $(C_4)-(C_7)$, the classes of SOR, SSOR and AOR methods were done in these situations, too.

The numerical behaviours of these methods for this example are quite analogous to those for Example 2, correspondingly.

Conclusions

The synchronous, the chaotic, and the asynchronous multisplitting relaxation methods reviewed in this paper afford various choices of parallel algorithms for solving the large sparse linear complementarity problems on the high-speed multiprocessor systems

The asynchronous multisplitting iterative methods are much more efficient than their corresponding synchronous and chaotic alternatives, in terms of both computing efficiency and algorithmic generality

In asynchronous computing environments, even if there are load imbalances, each processor is allowed to update the global iterate, or to retrive any piece of the global iterate residing in the host processor, at any time. This is one of the main advantages of these asynchronous multisplitting iterative methods

The convergence properties of these methods are demonstrated for some typical classes of matrices and under some suitable restrictions on both the multiple splittings and the involved parameters, which, therefore, gives theoretical guarantees for the applications of these methods

These works are not only developments of the classical iterative methods for the system of linear equations to the linear complementarity problem, but also improvements of the existing sequential SOR method and theory for the linear complementarity problem to the synchronous, the chaotic, and the asynchronous multisplitting iterative methods. Hence, they present both systematic algorithmic models in the sense of multisplitting and reliable theoretical guarantees in the sense of asymptotic and (or) monotone convergence for solving the large sparse linear complementarity problems on the modern high-speed multiprocessor systems.

The numerical computations show that: In terms of CPU time and parallel efficiency, the asynchronous multisplitting relaxation methods are superior to the corresponding synchronous multisplitting relaxation methods

The multisplitting accelerated overrelaxation methods are superior to the corresponding multisplitting successive overrelaxation methods

The two-sweep relaxed multisplitting methods are superior to the corresponding one-sweep relaxed multisplitting methods

In particular, the advantages of the AMAOR and AMSAOR methods over the AMSOR, AMSSOR and AMUSOR methods, respectively, are, roughly speaking, that:

(i) when the latter ones diverge, the former ones can still converge;

(ii) when the latter ones converge, the former ones converge faster with higher parallel efficiency; and

(iii) the former ones are less sensitive to the relaxation parameters and they have larger convergence domains than the latter ones

Moreover, the numerical property of the AMUSOR method is almost comparable with that of the AMSAOR method.

Therefore, we can conclude that the two-sweep multi-parameter relaxed asynchronous multisplitting methods have better numerical properties than their corresponding synchronous alternatives, and they should be our choice of methods in actual computations

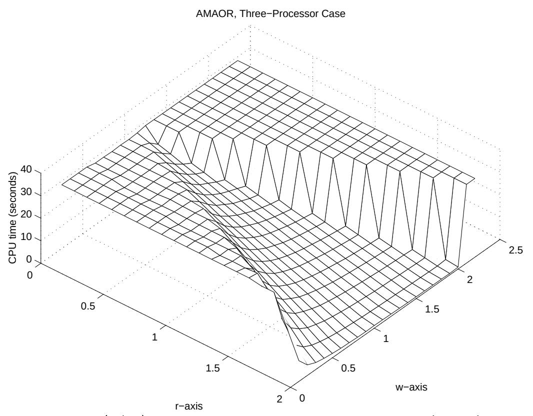

Figure (A_1): The behaviour of AMAOR method for the problem with $n = 4900$. The divergent points are represented in the graph by CPU time being 30.

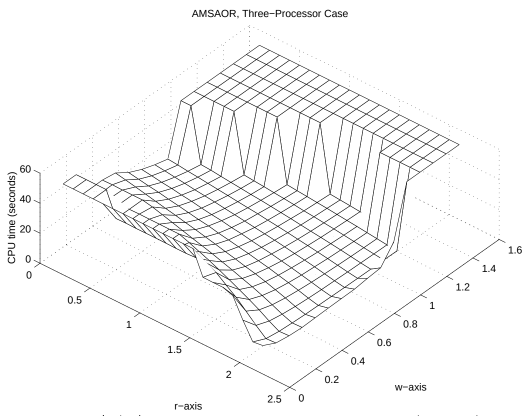

Figure (A_2): The behaviour of AMSAOR method for the problem with $n = 4900$. The divergent points are represented in the graph by CPU time being 40.

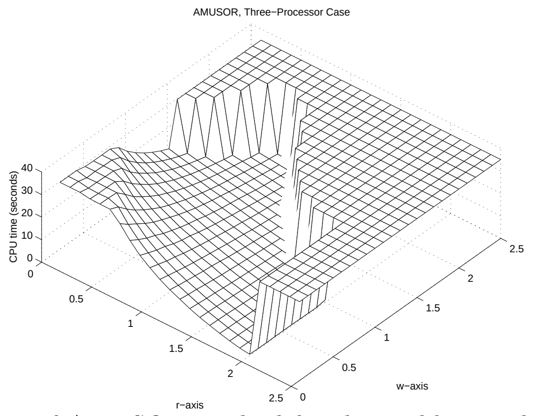

Figure (A_3): The behaviour of AMUSOR method for the problem with $n = 4900$. The divergent points are represented in the graph by CPU time being 30.

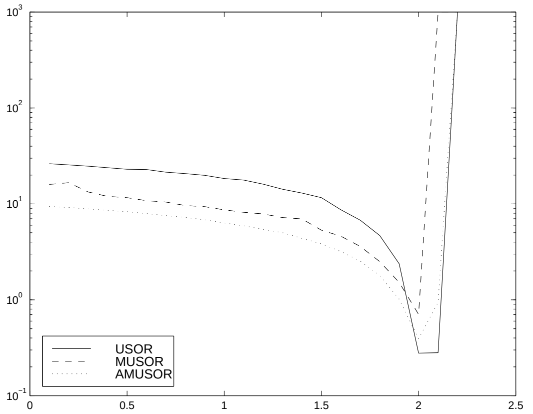

Figure (B_1): CPU vs. parameters curves for the USOR-like methods

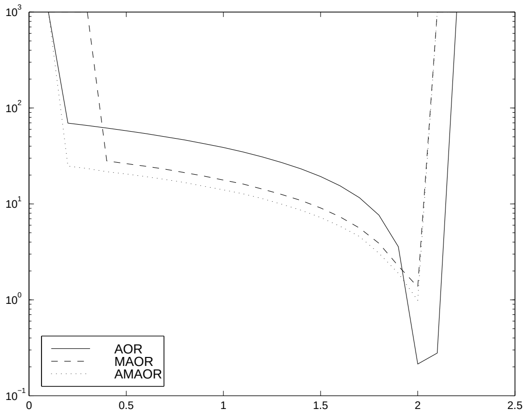

Figure (B_2): CPU vs. parameters curves for the AOR-like methods

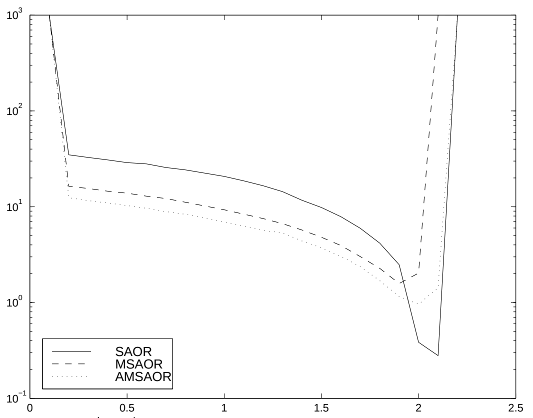

Figure (B_3): CPU vs. parameters curves for the SAOR-like methods

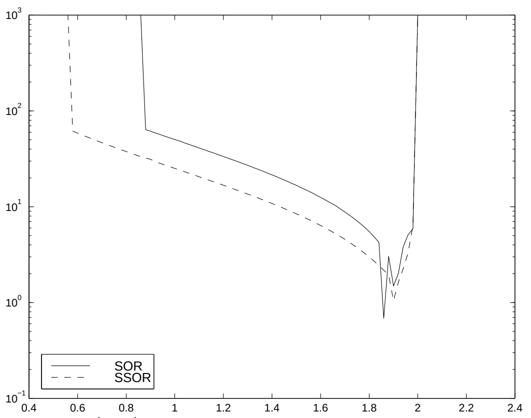

Figure (B_4): CPU vs. parameter curves for the SOR and SSOR methods

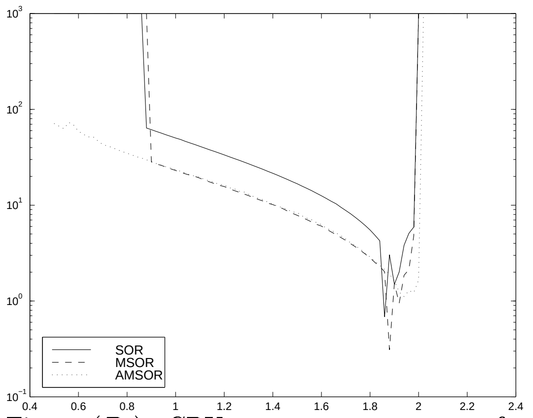

Figure (B_5): CPU vs. parameter curves for the SOR-like methods

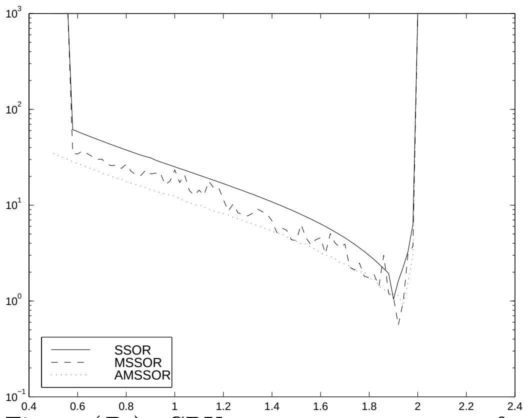

Figure (B_6): CPU vs. parameter curves for the SSOR-like methods

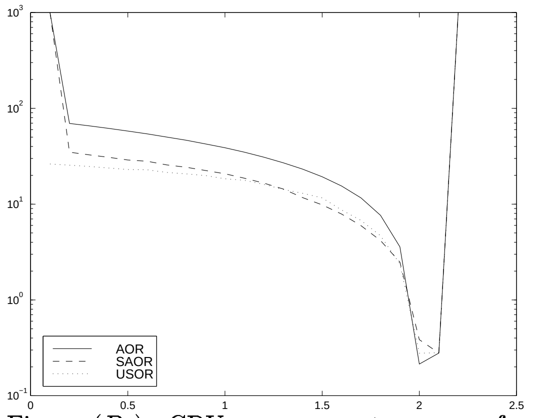

Figure (B_7): CPU vs. parameter curves for the AOR, SAOR and USOR methods

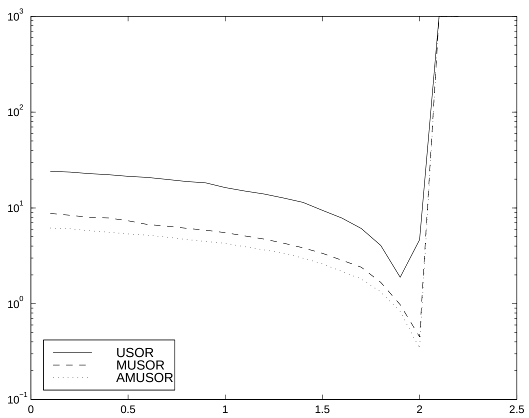

Figure (C_1): CPU vs. parameters curves for the USOR-like methods

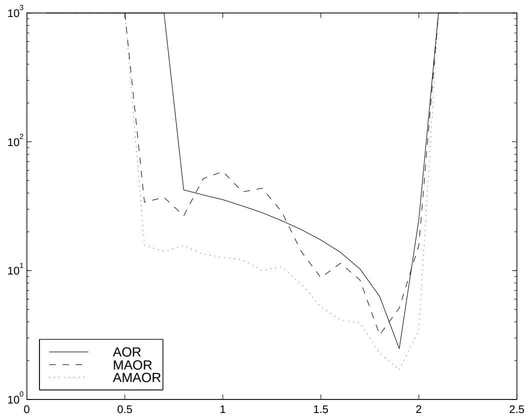

Figure (C_2): CPU vs. parameters curves for the AOR-like methods

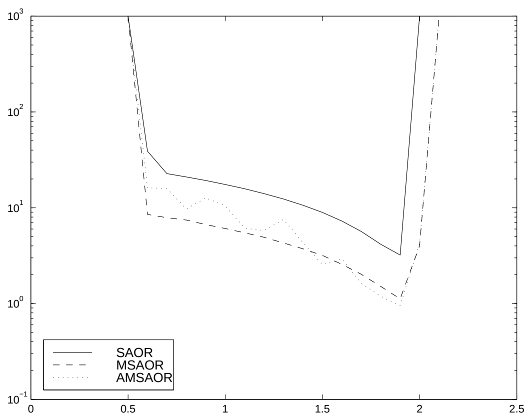

Figure (C_3): CPU vs. parameters curves for the SAOR-like methods

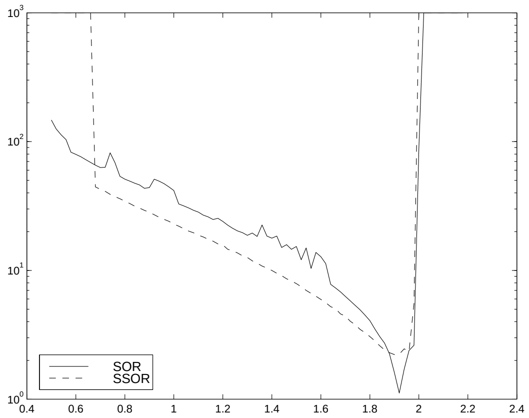

Figure (C_4): CPU vs. parameter curves for the SOR and SSOR methods

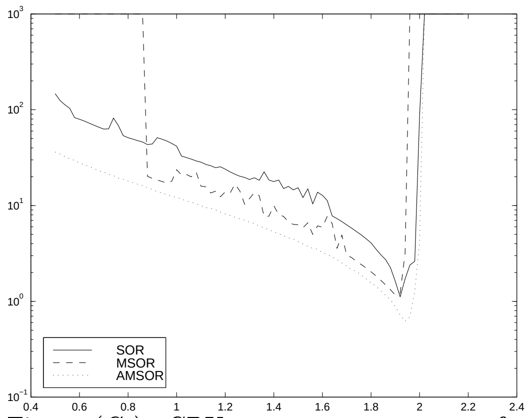

Figure (C_5): CPU vs. parameter curves for the SOR-like methods

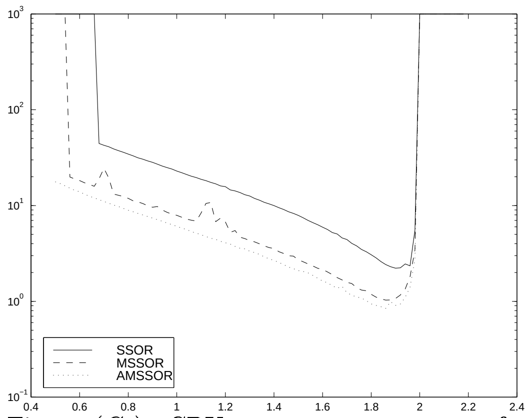

Figure (C_6): CPU vs. parameter curves for the SSOR-like methods

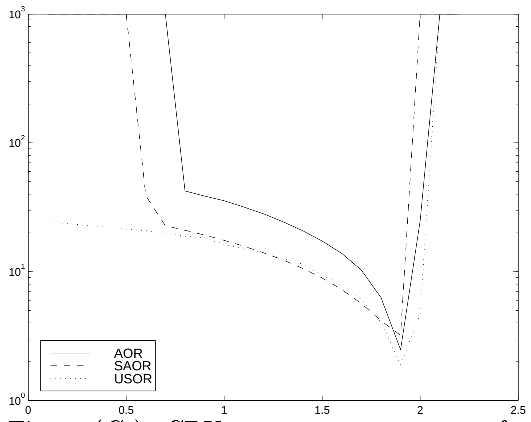

Figure (C_7): CPU vs. parameters curves for the AOR, SAOR and USOR methods