Received: 8 August 2019 | Revised: 19 May 2020 | Accepted: 31 May 2020
DOI: 10.1002/nla.2319

RESEARCH ARTICLE

WILEY

Modulus-based synchronous multisplitting methods for solving horizontal linear complementarity problems on parallel computers

Francesco Mezzadri

Department of Engineering "Enzo Ferrari", University of Modena and Reggio Emilia, Modena, Italy

Correspondence
Francesco Mezzadri, Department of Engineering "Enzo Ferrari", University of Modena and Reggio Emilia, via P. Vivarelli 10/1, building 26, Modena I-41125, Italy.
Email: francesco.mezzadri@unimore.it

Summary
In this article, we generalize modulus-based synchronous multisplitting methods to horizontal linear complementarity problems. In particular, first we define the methods of our concern and prove their convergence under suitable smoothness assumptions. Particular attention is devoted also to modulus-based multisplitting accelerated overrelaxation methods. Then, as multisplitting methods are well-suited for parallel computations, we analyze the parallel behavior of the proposed procedures. In particular, we do so by solving various test problems by a parallel implementation of our multisplitting methods. In this context, we carry out parallel computations on GPU with CUDA.

KEYWORDS
horizontal linear complementarity problems, matrix multisplitting, modulus-based methods, parallel computing

1 | INTRODUCTION

Linear complementarity problems (LCPs) consist in determining two vectors $\boldsymbol{z}, \boldsymbol{w} \in \mathbb{R}^n$ that satisfy

$$A\boldsymbol{z} - \boldsymbol{w} = \boldsymbol{q}; \quad \boldsymbol{w} \geq \boldsymbol{0}; \quad \boldsymbol{z} \geq \boldsymbol{0}; \quad \boldsymbol{z}^T\boldsymbol{w} = 0,$$

where $\boldsymbol{q} \in \mathbb{R}^n$ is a known term and $A \in \mathbb{R}^{n\times n}$. Such problems have several applications and have been widely studied.[1-3] For a complete description of LCPs and of their applications, the reader is referred to Reference 1.

The popularity of LCPs is testified also by the many solution techniques that have been proposed during the years. Classical methods include, for instance, interior point methods (see, eg, References 4,5), projected splitting methods,[6-9] and matrix multisplitting iteration methods.[2,3] More recently, modulus-based matrix splitting methods have been proposed.[10] This latter class of iterative methods is based on rewriting the LCP as an implicit fixed-point system (in this regard, see also References 11-13), which is then solved by a matrix splitting iteration method. The efficiency of modulus-based matrix splitting methods led to the formulation of many other modulus-based iterative procedures, such as the generalized[14] and the accelerated variants.[15,16] Furthermore, new methods of this kind are still being proposed and analyzed, especially in the sense of matrix multisplitting and of multigrid, as in the recent articles.[17-19] Other recent works on modulus-based matrix splitting methods include also References 20-22. Finally, modulus-based iterations have been recently extended to various generalizations of LCPs, like in References 23-28.

In particular, an interesting generalization of LCPs is given by horizontal linear complementarity problems (HLCPs), which have applications in structural mechanics, in mechanical and electrical engineering and in many other fields. In particular, the HLCP(A,B,$\boldsymbol{q}$) consists in finding a pair of vectors $\boldsymbol{z}, \boldsymbol{w} \in \mathbb{R}^n$ such that

$$A\boldsymbol{z} - B\boldsymbol{w} = \boldsymbol{q}, \qquad \boldsymbol{z} \geq \boldsymbol{0},\ \boldsymbol{w} \geq \boldsymbol{0}, \qquad \boldsymbol{z}^T\boldsymbol{w} = 0, \tag{1}$$

with $A, B \in \mathbb{R}^{n \times n}$. Solution techniques for these problems include reduction to LCP,[29,30] interior point methods,[31,32] homotopy approaches,[33] neural networks,[34] and projected splitting methods.[35] Recently, HLCPs have been reformulated as an implicit fixed-point system and solved by generalizations of modulus-based matrix splitting methods,[28] demonstrating the effectiveness of such strategies for solving HLCPs. The convergence of these methods has been further extended in Reference 36 and modulus-based parametrized nonsmooth Newton methods for HLCPs have been proposed in Reference 37.

However, these procedures are generally not very well-suited for parallel computations. Nonetheless, parallelization techniques could greatly increase the efficiency of the solution of HLCPs. This would be particularly interesting in modern computers, where multiple cores and, especially, GPU computing make the parallel use of tens of computational threads possible even on common desktop equipment.

In this context, it is interesting to analyze modulus-based iterations where the matrices A and B are split into multiple splittings. Indeed, synchronous multisplitting iterations are well-suited for parallel computations and are commonly used for solving linear systems[38,39] and LCPs (see, eg, References 2,3,40-42). In particular, modulus-based multisplitting (MM) iterations for LCPs[17,18] exhibited good parallel efficiency.

Hence, in this work, our aim is to generalize the modulus-based synchronous multisplitting iteration methods for LCPs[17] to the solution of HLCPs. In order to do so, in Section 2 we first recall some results on modulus-based methods for HLCPs. Then, we formulate our MM methods and provide some remarks on how they generalize both the MM iterations for LCPs[17] and the modulus-based matrix splitting iterations for HLCPs.[28] In this context, we also provide the splitting matrices and the iteration of modulus-based multisplitting AOR (MMAOR) methods for HLCPs, which are particularly interesting for parallel implementations. In Section 3, we then analyze the convergence of the methods, considering both the general and AOR multisplittings. In Section 4, we then provide and solve several numerical experiments. The considered test problems are solved in parallel on GPU by modulus-based multisplitting Jacobi (MMJ), multisplitting Gauss-Seidel, and multisplitting SOR methods, carrying out parallel computations on CUDA.[43] In the analysis of results, we devote particular attention to the parallel efficiency of the algorithms. Finally, concluding remarks are provided in Section 5.

2 | THE MM ITERATION FOR HLCPS

In this section, we recall some results on modulus-based matrix splitting methods for HLCPs[28] and formulate the MM iteration for HLCPs.

2.1 | Notation and preliminaries

Regarding notation, in the following, given two matrices $C, D \in \mathbb{R}^{m \times n}$, $C \geq D$ means that every entry of C is larger than or equal to the respective component of D. Similarly, $|C|$ means that every element of C is taken in absolute value. Furthermore, $\langle C \rangle$ is the comparison matrix of C, which is defined as the matrix of elements $\langle c_{ij} \rangle = |c_{ii}|$ for $i = j$ and $\langle c_{ij} \rangle = -|c_{ij}|$ for $i \neq j$, for $i = 1, \dots, m$ and $j = 1, \dots, n$.

Moreover, we say that C is an M-matrix if it is a Z-matrix (ie, its off-diagonal entries are nonpositive) and if its inverse is nonnegative. We instead say that C is an H-matrix if $\langle C \rangle$ is an M-matrix, and we say that C is an H_+-matrix if it is an H-matrix with positive diagonal entries.[42] Finally, in the following we also use the concepts of H-splitting and of H-compatible splitting. In particular, the splitting $A = M - N$ is called an H-splitting if $\langle M \rangle - |N|$ is an M-matrix and an H-compatible splitting if $\langle A \rangle = \langle M \rangle - |N|$.[44]

It is known by Reference 28 that HLCPs can be reformulated as follows:

Theorem 1. 28 *Let $A, B \in \mathbb{R}^{n\times n}$, let $\boldsymbol{q} \in \mathbb{R}^n$, and let $A = M_A - N_A$ and $B = M_B - N_B$ be splittings of A and of B, respectively. Furthermore, let Γ, Ω be two positive diagonal matrices of order n. Then,*

- *if $(\boldsymbol{z},\boldsymbol{w})$ is a solution of the HLCP(A,B,$\boldsymbol{q}$), then $\boldsymbol{x} = 1/2(\Gamma^{-1}\boldsymbol{z} - \Omega^{-1}\boldsymbol{w})$ satisfies*

$$(M_A\Gamma + M_B\Omega)\boldsymbol{x} = (N_A\Gamma + N_B\Omega)\boldsymbol{x} + (B\Omega - A\Gamma)|\boldsymbol{x}| + \boldsymbol{q} \tag{2}$$

- *if $\boldsymbol{x}$ satisfies (2), then*

$$\boldsymbol{z} = \Gamma(|\boldsymbol{x}| + \boldsymbol{x}); \qquad \boldsymbol{w} = \Omega(|\boldsymbol{x}| - \boldsymbol{x}), \tag{3}$$

  is a solution of the HLCP(A,B,$\boldsymbol{q}$).

On the basis of the previous theorem, the following modulus-based matrix splitting iteration for HLCPs was proposed.[28]

Method 1 (Modulus-based matrix splitting iteration for HLCPs[28]). Let $A, B \in \mathbb{R}^{n\times n}$ and $\boldsymbol{q} \in \mathbb{R}^n$. Furthermore, let Ω be a positive diagonal matrix of order n and γ be a positive constant. Starting from an initial guess $\boldsymbol{x}^{(0)} \in \mathbb{R}^n$, compute the $(k+1)$th iterate $\boldsymbol{x}^{(k+1)}$ as the solution of the linear system

$$(M_A + M_B\Omega)\boldsymbol{x} = (N_A + N_B\Omega)\boldsymbol{x}^{(k)} + (B\Omega - A)|\boldsymbol{x}^{(k)}| + \gamma\boldsymbol{q}, \tag{4}$$

with $A = M_A - N_A$ and $B = M_B - N_B$. Then, at the generic $(k+1)$th iteration, the complementarity vectors can be computed as

$$\boldsymbol{z}^{(k+1)} = \frac{1}{\gamma}(|\boldsymbol{x}^{(k+1)}| + \boldsymbol{x}^{(k+1)}); \qquad \boldsymbol{w}^{(k+1)} = \frac{1}{\gamma}\Omega(|\boldsymbol{x}^{(k+1)}| - \boldsymbol{x}^{(k+1)}). \tag{5}$$

2.2 | Formulating the MM method for HLCPs

Let us now formulate a MM iteration. In order to do so, let us consider ℓ splittings of A and B

$$\begin{aligned} A &= M_{A_k} - N_{A_k}, \quad k = 1, 2, \dots, \ell; \\ B &= M_{B_k} - N_{B_k}, \quad k = 1, 2, \dots, \ell, \end{aligned}$$

with $\ell \leq n$. Furthermore, let E_k, $k = 1,2,\dots,\ell$ be nonnegative weighting matrices such that

$$\sum_{k=1}^{\ell} E_k = I. \tag{6}$$

Then, we call the set of splittings

$$\{(M_{A_k}, N_{A_k}, M_{B_k}, N_{B_k}, E_k),\ k = 1, 2, \dots, \ell\}$$

a *multisplitting* of the system matrices A,B of the HLCP(A,B,$\boldsymbol{q}$).

Based on the results in Section 2.1, we can formulate a MM iteration for HLCPs. Indeed, consider Theorem 1 with Ω prescribed positive diagonal matrix and $\Gamma = \frac{1}{\gamma}I$, with $\gamma > 0$. Then, performing a scaling, we know that if $\boldsymbol{x}$ satisfies

$$(M_{A_k} + M_{B_k}\Omega)\boldsymbol{x} = (N_{A_k} + N_{B_k}\Omega)\boldsymbol{x} + (B\Omega - A)|\boldsymbol{x}| + \gamma\boldsymbol{q}, \quad k = 1, 2, \dots, \ell, \tag{7}$$

then

$$\boldsymbol{z} = \frac{1}{\gamma}(|\boldsymbol{x}| + \boldsymbol{x}) \quad \text{and} \quad \boldsymbol{w} = \frac{1}{\gamma}\Omega(|\boldsymbol{x}| - \boldsymbol{x})$$

solve the HLCP(A,B,$\boldsymbol{q}$), for any two splittings $A = M_{A_k} - N_{A_k}$ and $B = M_{B_k} - N_{B_k}$. Hence, we formulate the following MM iteration method.

Method 2 ((MM iteration for HLCPs)). Let $A, B \in \mathbb{R}^{n\times n}$ and $\boldsymbol{q} \in \mathbb{R}^n$. Furthermore, let Ω be a positive diagonal matrix of order n, let γ be a positive constant and let $\{(M_{A_k}, N_{A_k}, M_{B_k}, N_{B_k}, E_k),\ k = 1, 2, \dots, \ell\}$ be a multisplitting of the matrices (A,B). Then, starting from an initial guess $\boldsymbol{x}^{(0)} \in \mathbb{R}^n$, let $\boldsymbol{x}^{(m,k)}$, with $m = 0,1,2,\dots$ and $k = 1,2,\dots,\ell$, be the solution of the linear system

$$(M_{A_k} + M_{B_k}\Omega)\boldsymbol{x} = (N_{A_k} + N_{B_k}\Omega)\boldsymbol{x}^{(m)} + (B\Omega - A)|\boldsymbol{x}^{(m)}| + \gamma\boldsymbol{q}, \tag{8}$$

and set

$$\boldsymbol{x}^{(m+1)} := \sum_{k=1}^{\ell} E_k x^{(m,k)}. \tag{9}$$

At the generic $(m+1)$th iteration, the complementarity vectors can then be computed as

$$\boldsymbol{z}^{(m+1)} = \frac{1}{\gamma}(|\boldsymbol{x}^{(m+1)}| + \boldsymbol{x}^{(m+1)}); \qquad \boldsymbol{w}^{(m+1)} = \frac{1}{\gamma}\Omega(|\boldsymbol{x}^{(m+1)}| - \boldsymbol{x}^{(m+1)}). \tag{10}$$

Remark 1. Method 2 reduces to the standard MM method for LCPs[17] when $B = I$ and the multisplitting $\{(M_{A_k}, N_{A_k}, I, 0, E_k),\ k = 1, 2, \dots, \ell\}$ is used. Furthermore, it reduces to the modulus-based matrix splitting method for HLCPs[28] when we use a single splitting (ie, when $\ell = 1$).

The peculiarity of multisplitting methods is that they are well-suited for parallel computation. Indeed, each of the ℓ splittings is independent of the others and can be computed on a different processor. Furthermore, at the generic mth iteration, $m = 0,1,2,\dots$, we may compute only the entries of $\boldsymbol{x}^{(m,k)}$, $k = 1,2,\dots,\ell$ that we actually need to compute $\boldsymbol{x}^{(m+1)}$. In this context, the choice of the splitting matrices is particularly important in order to reduce the computations performed by each processor.

2.3 | An interesting special case: The MMAOR

In this regard, let D_A be the diagonal part of A and let D_B be the diagonal part of B. Furthermore, let $L_{A_k} \in \mathbb{R}^{n\times n}$ and $L_{B_k} \in \mathbb{R}^{n\times n}$, $k = 1,\dots,\ell$, be strictly lower triangular matrices. However, they need not be the entire lower triangular part of A and B, respectively. For instance, assuming, for simplicity, that n is divisible by ℓ, L_{A_k} may be defined to contain the strictly lower triangular part of A from row $(k-1)n/l$ to kn/l, for $k = 1,\dots,\ell$ (and analogously for L_{B_k}). Finally, let $U_{A_k} = D_A - L_{A_k} - A$ and $U_{B_k} = D_B - L_{B_k} - B$. Notice that, in general, U_{A_k} and U_{B_k} are not upper triangular. We also assume that (D_A, L_{A_k}, U_{A_k}) and (D_B, L_{B_k}, U_{B_k}) have conformable block partitions when A and B are partitioned into blocks. More details on the form of the multisplitting actually used in the implementations are provided at the beginning of Section 4.

Then, based on the triangular multisplitting

$$(D_A - L_{A_k}, U_{A_k}, D_B - L_{B_k}, U_{B_k}, E_k), \quad k = 1, \dots, \ell,$$

we define the multisplitting AOR for HLCPs by

$$\begin{cases} M_{A_k} = \frac{1}{\alpha}(D_A - \beta L_{A_k}) \\ N_{A_k} = \frac{1}{\alpha}[(1-\alpha)D_A + (\alpha-\beta)L_{A_k} + \alpha U_{A_k}] \end{cases} \tag{11}$$

and

$$\begin{cases} M_{B_k} = \frac{1}{\alpha}(D_B - \beta L_{B_k}) \\ N_{B_k} = \frac{1}{\alpha}[(1-\alpha)D_B + (\alpha-\beta)L_{B_k} + \alpha U_{B_k}] \end{cases} \tag{12}$$

for $k=1,\dots,\ell$. Hence, at the generic $(m+1)$th iteration, $m=0,1,2,\dots$, the *MMAOR method* consists in determining $\boldsymbol{x}^{(m,k)}$ as the solution of

$$[D_A-\beta L_{A_k}+(D_B-\beta L_{B_k})\Omega]\boldsymbol{x}=[(1-\alpha)D_A+(\alpha-\beta)L_{A_k}+\alpha U_{A_k}]\boldsymbol{x}^{(m)} \\ +[(1-\alpha)D_B+(\alpha-\beta)L_{B_k}+\alpha U_{B_k}]\Omega\boldsymbol{x}^{(m)}+\alpha(B\Omega-A)|\boldsymbol{x}^{(m)}|+\alpha\gamma\boldsymbol{q}, \tag{13}$$

for $k=1,2,\dots,\ell$, and in building the new vector $\boldsymbol{x}^{(m+1)}$ as in (9). In particular, we obtain the MMJ iteration when $\alpha=1$, $\beta=0$, the modulus-based multisplitting Gauss-Seidel (MMGS) iteration when $\alpha=\beta=1$ and the modulus-based multisplitting SOR (MMSOR) iteration when $\alpha=\beta$.

3 | CONVERGENCE ANALYSIS

We now analyze the convergence of the MM iteration methods for HLCPs. In particular, assuming that the solution of the HLCP(A,B,$\boldsymbol{q}$) is unique, in this section we prove that the iterations converge under assumptions similar to the ones made for the modulus-based matrix splitting method in Reference 28.

Theorem 2. *Let $(\boldsymbol{z}^*,\boldsymbol{w}^*)$ be the solution of the HLCP(A,B,$\boldsymbol{q}$). Let $A,B\in\mathbb{R}^{n\times n}$ be H_+-matrices and let $\{(M_{A_k},N_{A_k},M_{B_k},N_{B_k},E_k),\ k=1,\dots,\ell\}$ be a multisplitting of (A,B). Assume that $A=M_{A_k}-N_{A_k}$ and $B=M_{B_k}-N_{B_k}$ are H-compatible splittings, for $k=1,\dots,\ell$. Furthermore, let $\Omega\geq D_AD_B^{-1}$ and let $|b_{ij}|\omega_{jj}\leq|a_{ij}|$ for $i=1,\dots,n$ and $j\neq i$. Finally, let $\operatorname{sign}(a_{ij})=\operatorname{sign}(b_{ij})$ for any $b_{ij}\neq0$, for $i,j=1,\dots,n$, and let $M_{A_k}+M_{B_k}\Omega$ be an H_+-matrix, for $k=1,\dots,\ell$. Then the sequence $\{\boldsymbol{z}^{(k)},\boldsymbol{w}^{(k)}\}$ generated by Method 2 converges to $(\boldsymbol{z}^*,\boldsymbol{w}^*)$ for any initial iterate $\boldsymbol{x}^{(0)}\in\mathbb{R}^n$.*

Proof. As (7) must be satisfied at the solution $\boldsymbol{x}^*$, by (7) and (8) we can write the error at the iteration (m,k) as

$$|\boldsymbol{x}^{(m,k)}-\boldsymbol{x}^*|=|(M_{A_k}+M_{B_k}\Omega)^{-1}[(N_{A_k}+N_{B_k}\Omega)(\boldsymbol{x}^{(m)}-\boldsymbol{x}^*)+(B\Omega-A)(|\boldsymbol{x}^{(m)}|-|\boldsymbol{x}^*|)]|,$$

for $k=1,\dots,\ell$. Hence, considering also (9), we have

$$\begin{aligned}|\boldsymbol{x}^{(m+1)}-\boldsymbol{x}^*|&\leq\sum_{k=1}^{\ell}E_k|\boldsymbol{x}^{(m,k)}-\boldsymbol{x}^*|\\&\leq\sum_{k=1}^{\ell}E_k\langle M_{A_k}+M_{B_k}\Omega\rangle^{-1}[|N_{A_k}+N_{B_k}\Omega|+|B\Omega-A|]|\boldsymbol{x}^{(m)}-\boldsymbol{x}^*|,\end{aligned}$$

where, in the last passage, we have also exploited that $|(M_{A_k}+M_{B_k}\Omega)^{-1}|\leq\langle M_{A_k}+M_{B_k}\Omega\rangle^{-1}$. This latter inequality holds since $M_{A_k}+M_{B_k}\Omega$ is an H_+-matrix by hypothesis, $k=1,\dots,\ell$.[44,45]

For compactness, set

$$T:=\sum_{k=1}^{\ell}E_k\langle M_{A_k}+M_{B_k}\Omega\rangle^{-1}[|N_{A_k}+N_{B_k}\Omega|+|B\Omega-A|]\geq0.$$

Remembering that $\sum_{k=1}^{\ell}E_k=I$, adding and subtracting the same term $\langle M_{A_k}+M_{B_k}\Omega\rangle$ inside the square bracket, we get

$$T=I-\sum_{k=1}^{\ell}E_k\langle M_{A_k}+M_{B_k}\Omega\rangle^{-1}[\langle M_{A_k}+M_{B_k}\Omega\rangle-|N_{A_k}+N_{B_k}\Omega|-|B\Omega-A|]. \tag{14}$$

Next, notice that M_{A_k} and M_{B_k} have positive diagonal part. Indeed, the facts that A is an H_+-matrix and that we are applying H-compatible splittings implies $m_{ii}^{A_k}-n_{ii}^{A_k}=a_{ii}>0$ and $|m_{ii}^{A_k}|-|n_{ii}^{A_k}|=a_{ii}>0$ for $i=1\dots n$ and for $k=1,\dots,\ell$. Any $m_{ii}^{A_k}\leq0$ cannot satisfy both these conditions. Analogous evaluations can be performed with regard to M_{B_k}. Then, $\langle M_{A_k}+M_{B_k}\Omega\rangle-|N_{A_k}+N_{B_k}\Omega|-|B\Omega-A|$ is larger than or equal to

$$\langle M_{A_k}\rangle+\langle M_{B_k}\Omega\rangle-|N_{A_k}|-|N_{B_k}\Omega|-|B\Omega-A|. \tag{15}$$

Since $A = M_{A_k} - N_{A_k}$ and $B = M_{B_k} - N_{B_k}$ are H-compatible splittings, $k = 1, \dots, \ell$, we have $\langle M_{A_k} \rangle - |N_{A_k}| = \langle A \rangle$ and $\langle M_{B_k}\Omega \rangle - |N_{B_k}\Omega| = \langle B \rangle \Omega$, for $k = 1, \dots, \ell$. Furthermore, by the assumptions $\Omega \geq D_A D_B^{-1}$ and $|b_{ij}|\omega_{jj} \leq |a_{ij}|$ for $i = 1, \dots, n$ and $j \neq i$ with $\operatorname{sign}(a_{ij}) = \operatorname{sign}(b_{ij})$, we also have

$$|B\Omega - A| = D_B\Omega - D_A + |C_A| - |C_B|\Omega = -\langle A \rangle + \langle B \rangle \Omega,$$

where, for compactness, we have set $C_A := D_A - A$ and $C_B := D_B - B$. Then, it easily follows

$$\langle M_{A_k} \rangle - |N_{A_k}| + \langle M_{B_k}\Omega \rangle - |N_{B_k}\Omega| - |B\Omega - A| = 2\langle A \rangle,$$

which, by (14) with (15), implies

$$T \leq I - 2\sum_{k=1}^{\ell} E_k \langle M_{A_k} + M_{B_k}\Omega \rangle^{-1} \langle A \rangle.$$

Defining $V := D_A^{-1}|C_A|$, we can then write

$$\begin{aligned} T &\leq I - 2\sum_{k=1}^{\ell} E_k \langle M_{A_k} + M_{B_k}\Omega \rangle^{-1} (D_A - |C_A|) \\ &= I - 2\sum_{k=1}^{\ell} E_k \langle M_{A_k} + M_{B_k}\Omega \rangle^{-1} (D_A - D_A V) \\ &= I - 2\sum_{k=1}^{\ell} E_k \langle M_{A_k} + M_{B_k}\Omega \rangle^{-1} D_A (I - V). \end{aligned}$$

As A is an H-matrix, we have $\rho(V) = \rho(D_A^{-1}|C_A|) < 1$.[44,46] Let us then define $V_\epsilon = V + \epsilon \boldsymbol{e}\boldsymbol{e}^T$, where $\boldsymbol{e}$ is the unity vector of order n and ϵ is an arbitrary positive number such that $\rho_\epsilon := \rho(V_\epsilon) < 1$. Therefore,

$$T < I - 2\sum_{k=1}^{\ell} E_k \langle M_{A_k} + M_{B_k}\Omega \rangle^{-1} D_A (I - V_\epsilon)$$

Next, applying also the Perron-Frobenius theorem,[47] we know that there exists a vector $\boldsymbol{v}_\epsilon > \boldsymbol{0}$ such that $(I - V_\epsilon)\boldsymbol{v}_\epsilon = (1 - \rho_\epsilon)\boldsymbol{v}_\epsilon$. Hence, multiplying both sides of the previous equality by such a vector $\boldsymbol{v}_\epsilon$, we obtain

$$T\boldsymbol{v}_\epsilon < \boldsymbol{v}_\epsilon - 2(1 - \rho_\epsilon)\sum_{k=1}^{\ell} E_k \langle M_{A_k} + M_{B_k}\Omega \rangle^{-1} D_A \boldsymbol{v}_\epsilon.$$

At this point, notice that $1 - \rho_\epsilon > 0$, $E_k \geq 0$ by the definition of the weighting matrices, D_A is a positive diagonal matrix and $\langle M_{A_k} + M_{B_k}\Omega \rangle^{-1}$ is nonsingular and nonnegative, as it is the inverse of an M-matrix by assumption, for $k = 1, \dots, \ell$. Therefore, we obtain

$$T\boldsymbol{v}_\epsilon < \boldsymbol{v}_\epsilon$$

with $T \geq 0$, $\boldsymbol{v}_\epsilon > \boldsymbol{0}$. Hence, $\rho(T) < 1$ [46, p. 28] and the MM iteration converges to the solution of the HLCP(A,B,$\boldsymbol{q}$). ■

Theorem 3. *Let $(\boldsymbol{z}^*, \boldsymbol{w}^*)$ be the solution of the HLCP(A,B,$\boldsymbol{q}$). Furthermore, let $\{(M_{A_k}, N_{A_k}, M_{B_k}, N_{B_k}, E_k),\ k = 1, \dots, \ell\}$ be the MMAOR of (A,B) in (11) to (12) and let Ω, A and B satisfy the assumptions of Theorem 2. Then,*

(i) the sequences $\{\boldsymbol{z}^{(k)}, \boldsymbol{w}^{(k)}\}$ generated by the MMJ and by the MMGS iteration methods converge to $(\boldsymbol{z}^*, \boldsymbol{w}^*)$ for any initial iterate $\boldsymbol{x}^{(0)} \in \mathbb{R}^n$;

(ii) the sequence $\{\boldsymbol{z}^{(k)}, \boldsymbol{w}^{(k)}\}$ generated by the MMAOR iteration method converges to $(\boldsymbol{z}^*, \boldsymbol{w}^*)$ for any initial iterate $\boldsymbol{x}^{(0)} \in \mathbb{R}^n$ provided that

$$0 < \beta \leq \alpha < \min\,[(D_A\rho + D_B\Omega)^{-1}(D_A + D_B\Omega)\boldsymbol{e}], \tag{16}$$

with ρ spectral radius of $V := D_A^{-1}|C_A|$ and $\boldsymbol{e}$ unity vector of order n.

Proof. Regarding (i), the proof immediately follows from Theorem 2. Indeed, multisplitting Jacobi and multisplitting Gauss-Seidel are based on H-compatible splittings and all other assumptions of Theorem 2 are satisfied.

Regarding (ii), we instead have to analyze the iteration. Indeed, (16) admits values of α for which the used splittings are not H-compatible.

Proceeding as in the first part of Theorem 2 and considering the MMAOR iteration (13), we find that the error can be written as

$$|\boldsymbol{x}^{(m+1)} - \boldsymbol{x}^*| \leq T|\boldsymbol{x}^{(m)} - \boldsymbol{x}^*|$$

with

$$\begin{aligned} T = I - \sum_{k=1}^{\ell} E_k[D_A - \beta|L_{A_k}| + (D_B - \beta|L_{B_k}|)\Omega]^{-1} \{D_A + D_B\Omega - [|1-\alpha|D_A + \alpha|L_{A_k}| + \alpha|U_{A_k}| \\ +(|1-\alpha|D_B + \alpha|L_{B_k}| + \alpha|U_{B_k}|)\Omega + \alpha|B\Omega - A|]\} \geq 0, \end{aligned} \tag{17}$$

where we have also exploited $\beta \leq \alpha$. Setting $C_A := D_A - A$ and $C_B := D_B - B$, by simple passages we then find

$$\begin{aligned} T = I - \sum_{k=1}^{\ell} E_k[D_A - \beta|L_{A_k}| + (D_B - \beta|L_{B_k}|)\Omega]^{-1} [D_A + D_B\Omega \\ -|1-\alpha|D_A - \alpha|C_A| - |1-\alpha|D_B\Omega - \alpha|C_B|\Omega - \alpha|B\Omega - A|]. \end{aligned}$$

Furthermore, with the assumptions of Theorem 2 on A and B, we have seen that

$$|B\Omega - A| = -D_A + |C_A| + D_B\Omega - |C_B|\Omega.$$

Hence, we obtain

$$\begin{aligned} T = I - \sum_{k=1}^{\ell} E_k[D_A - \beta|L_{A_k}| + (D_B - \beta|L_{B_k}|)\Omega]^{-1} \\ \cdot [D_A + D_B\Omega - |1-\alpha|D_A - |1-\alpha|D_B\Omega + \alpha D_A - \alpha D_B\Omega - 2\alpha|C_A|]. \end{aligned} \tag{18}$$

At this point, notice that the method converges if $\alpha \leq 1$. Indeed, in this case, $|1-\alpha| = 1-\alpha$ and the second square bracket reduces to

$$2\alpha D_A - 2\alpha|C_A| = 2\alpha D_A(I - V) > 2\alpha D_A(I - V_\epsilon).$$

Proceeding as in Theorem 2, we then have that there exists a positive vector $\boldsymbol{v}_\epsilon$ such that

$$T\boldsymbol{v}_\epsilon < \boldsymbol{v}_\epsilon - (1-\rho_\epsilon)\sum_{k=1}^{\ell} E_k[D_A - \beta|L_{A_k}| + (D_B - \beta|L_{B_k}|)\Omega]^{-1}\, 2\alpha D_A\, \boldsymbol{v}_\epsilon < \boldsymbol{v}_\epsilon,$$

with $\rho_\epsilon := \rho(V_\epsilon) < 1$. Hence, $1 - \rho_\epsilon > 0$, which, by the positivity of also the other terms, implies $\rho(T) < 1$.

If $\alpha > 1$, we instead have $|1-\alpha| = \alpha - 1$ and the square bracket in (18) becomes

$$2D_A + 2D_B\Omega - 2\alpha D_B\Omega - 2\alpha|C_A| = 2[D_A(I - \alpha V) + (1-\alpha)D_B\Omega] > 2[D_A(I - \alpha V_\epsilon) + (1-\alpha)D_B\Omega].$$

Hence, there exists a positive vector $\boldsymbol{v}_\epsilon$ such that

$$\begin{aligned} T\boldsymbol{v}_\epsilon &< \boldsymbol{v}_\epsilon - 2\sum_{k=1}^{\ell} E_k[D_A - \beta|L_{A_k}| + (D_B - \beta|L_{B_k}|)\Omega]^{-1}[D_A(I - \alpha V_\epsilon) + (1-\alpha)D_B\Omega]\boldsymbol{v}_\epsilon \\ &= \boldsymbol{v}_\epsilon - 2\sum_{k=1}^{\ell} E_k[D_A - \beta|L_{A_k}| + (D_B - \beta|L_{B_k}|)\Omega]^{-1}[D_A(1 - \alpha\rho_\epsilon) + (1-\alpha)D_B\Omega]\boldsymbol{v}_\epsilon. \end{aligned}$$

Next, consider that

$$D_A(1-\alpha\rho)+(1-\alpha)D_B\Omega > 0 \tag{19}$$

along the main diagonal if

$$D_A + D_B\Omega > \alpha(\rho D_A + D_B\Omega).$$

Since the involved matrices are diagonal, it follows that (19) is satisfied if

$$\alpha I < (D_A\rho + D_B\Omega)^{-1}(D_A + D_B\Omega)$$

or, equivalently,

$$\alpha < \min\,[(D_A\rho + D_B\Omega)^{-1}(D_A + D_B\Omega)\boldsymbol{e}],$$

which we assumed to hold true by hypothesis. Then, by continuity of the spectral radius, we can choose an $\epsilon > 0$ that is sufficiently small so that

$$D_A(1-\alpha\rho_\epsilon)+(1-\alpha)D_B\Omega > 0$$

is satisfied as well. Therefore, we again obtain $T\boldsymbol{v}_\epsilon < \boldsymbol{v}_\epsilon$ with $T \geq 0$ and $\boldsymbol{v}_\epsilon > \boldsymbol{0}$, implying $\rho(T) < 1$. Then, the method converges to the solution of the HLCP(A,B,$\boldsymbol{q}$). ∎

Remark 2. Notice that $(D_A\rho + D_B\Omega)^{-1}(D_A + D_B\Omega)$ is a positive diagonal matrix whose diagonal elements are all larger than one, as $\rho < 1$. Then, condition (ii) of Theorem 3 allows for $\alpha > 1$, for which the MMAOR splitting is not H-compatible. Hence, Theorem 3 enlarges the convergence conditions of the MMAOR iteration for HLCPs with respect to Theorem 2.

Remark 3. When $B = I$, the MMAOR iteration for HLCPs does not reduce to the MMAOR iteration for LCPs.[17] Indeed, also in this case, the MMAOR iteration for HLCPs performs a splitting on B as well, so that the used multisplitting is $\{(M_{A_k}, N_{A_k}, \frac{1}{\alpha}I, \frac{1-\alpha}{\alpha}I, E_k),\quad k = 1, 2, \dots, \ell\}$. Thus, Theorem 3 for $B = I$ does not reduce to theorem 4.1 of Reference 17, but it provides the convergence of an "HLCP-like" MMAOR iteration method for LCPs, where also $B = I$ is split.

Nonetheless, the previous results reduce to theorem 4.1 of Reference 17 if we set $B = I$ and consider the multisplitting $\{(M_{A_k}, N_{A_k}, I, 0, E_k),\quad k = 1, 2, \dots, \ell\}$ (see also Remark 1).

We also notice that the convergence domain of the MMAOR method is larger than that of the MAOR method according to theorem 5 of Reference 28, which would require $\alpha \leq 1$.

4 | NUMERICAL EXPERIMENTS

In this section, we evaluate numerically the behavior of the proposed MM methods in the framework of parallel computing.

Numerical experiments have been conducted in Fortran on a desktop computer equipped with Windows 10. Parallel computations have been performed on GPU using CUDA[43] and Fortran codes have been compiled by PGI compiler Community Edition Version 19.4. The computational load has been distributed uniformly among processors by choosing weighting matrices as in Reference 17, which is

$$E_k = \operatorname{diag}(0, \dots, 0, I_{s_k}, 0, \dots, 0),$$

where

$$s_k = \begin{cases} \phi_q + 1 & \text{if } k \leq \phi_r \\ \phi_q & \text{if } k > \phi_r \end{cases}$$

with $\phi_q \geq 0$, $\phi_r \geq 0$ integers such that $n = \ell\phi_q + \phi_r$ and $0 \leq \phi_r < \ell$. Furthermore, L_{A_k} and L_{B_k}, $k = 1, \dots, \ell$, are defined as the strictly lower triangular part of E_kAE_k and of E_kBE_k, respectively. The matrices U_{A_k} and U_{B_k} are defined accordingly.

The graphic card used in the numerical experiments is an NVIDIA GeForce GTX 1050 Ti, which is provided with 6 multiprocessors, each consisting of 128 threads. Hence, the GPU is provided with a total of 768 CUDA threads. The total amount of global memory is 4 GB, the maximum clock rate is 1.62 GHz and the maximum number of threads per multiprocessor is 2048.

At each iteration, we measure the elapsed time needed to run the parallel computations, which include the entire MM iteration (8) and the construction of the vector $\boldsymbol{x}^{(m)}$ as in (9). As commonly done in CUDA, elapsed times are measured by creating two CUDA events: one immediately before starting the parallel procedures and the other immediately after their execution. The elapsed time between the two events is then measured by the CUDA routine `cudaEventElapsedTime`, which has a resolution of about 0.5 microseconds.

In the following, we report the sum of these times at all iterations, which we denote by T_ℓ, where ℓ denotes the number of threads (and of simultaneous splittings) that are being used. We instead denote the computational efficiency by $\mathcal{E}_\ell = T_1/(\ell T_\ell)$ and the number of iterations before convergence by IT_ℓ.

In all cases, the iterative procedures are stopped when the Euclidean norm of the residual

$$res := \|A\boldsymbol{z}^{(m)} - B\boldsymbol{w}^{(m)} - \boldsymbol{q}\|$$

becomes smaller than 10^{-6}. This ensures that the computed vectors satisfy the linear system. The complementarity condition and the nonnegativity of vectors are, instead, necessarily satisfied by the definition of the iterates as in (10).

4.1 | Statement of the test problems

In the following, we solve three sets of HLCPs by MM methods with various choices of ℓ. In all cases, we build test problems of solution

$$\boldsymbol{z}^* = (0,1,0,\ \dots\ ,1)^T; \qquad \boldsymbol{w}^* = (1,0,1,\ \dots\ ,0)^T$$

by computing a suitable known term as $\boldsymbol{q} = A\boldsymbol{z}^* - B\boldsymbol{w}^*$. Furthermore, in all problems we start from the initial iterate $\boldsymbol{x}^{(0)} = (2, 2, \dots, 2)^T$.

In the first problem, which we denote by Example 1, the matrix A is modeled after the first example of Reference 17, while B is a block diagonal matrix of tridiagonal blocks. In particular, defining $S = \text{tridiag}(-1, 4, -1) \in \mathbb{R}^{h\times h}$, the matrices of the problem are

$$A = \begin{pmatrix} S & -I & -I & & \\ & S & -I & -I & \\ & & \ddots & \ddots & \ddots \\ & & & S & -I \\ & & & & S \end{pmatrix}, \quad B = \begin{pmatrix} S & & & & \\ & S & & & \\ & & \ddots & & \\ & & & S & \\ & & & & S \end{pmatrix},$$

with $A, B \in \mathbb{R}^{n\times n}$, with $n = h^2$.

Example 2 and Example 3 are, instead, the problems with symmetric and nonsymmetric system matrices that have been solved in Reference 28 with "standard" modulus-based matrix splitting methods. In particular, defining

$$\hat{A} = \begin{pmatrix} S & -I & & & \\ -I & S & -I & & \\ & \ddots & \ddots & \ddots & \\ & & -I & S & -I \\ & & & -I & S \end{pmatrix}, \quad \hat{B} = \begin{pmatrix} S & & & & \\ & S & & & \\ & & \ddots & & \\ & & & S & \\ & & & & S \end{pmatrix},$$

Example 2 is defined by the matrices $A, B \in \mathbb{R}^{n\times n}$, with $A = \hat{A} + \mu I$ and $B = \hat{B} + \nu I$, where μ, ν real parameters and $n = h^2$. The matrices of Example 3 are, instead, defined as $A = \tilde{A} + \mu I$ and $B = \tilde{B} + \nu I$ with μ, ν real parameters and with

$$\tilde{A} = \begin{pmatrix} \tilde{S} & -0.5I & & & \\ -1.5I & \tilde{S} & -0.5I & & \\ & \ddots & \ddots & \ddots & \\ & & -1.5I & \tilde{S} & -0.5I \\ & & & -1.5I & \tilde{S} \end{pmatrix}, \quad \tilde{B} = \begin{pmatrix} \tilde{S} & & & & \\ & \tilde{S} & & & \\ & & \ddots & & \\ & & & \tilde{S} & \\ & & & & \tilde{S} \end{pmatrix},$$

where $\tilde{S} = \text{tridiag}(-1.5, 4, -0.5)$. In the following, in these latter problems we set $\mu = 0$ and $\nu = 4$. Finally, the parameter Ω of modulus-based methods is set as $\Omega = I$ for Example 1 and as $\Omega = 0.5I$ for Example 2 and Example 3. We also set $\gamma = 2$ in all cases.

4.2 | A remark on grids and blocks of threads in CUDA

In CUDA, threads are organized in blocks, which, in turn, are grouped into a grid of blocks. Multiplying the dimensions of the grid, *ng*, by the dimensions of the blocks, *nb*, we obtain the total number of threads that are being launched.

The constraints on the choice of grid and block dimensions are not problematic in our cases: for instance, grids can be made of up to $65535 \times 65535 \times 65535$ blocks, each consisting of up to 512 (or more, depending on the GPU) threads. Nonetheless, performance is influenced by block and grid dimensions, especially when many threads are used. The dimension of blocks should, indeed, be selected to maximize the occupancy. Some remarks on the choice of *ng* and *nb* in the following experiments are, then, needed.

First, when we use fewer than 32 processors, the measured times do not vary significantly when *ng* and *nb* are changed. This is reasonable, as, in many applications, each block can have 64 to 128 threads before slowdowns occur.

To analyze what happens when we use more processors, let us instead consider an example. In particular, let us consider Example 1 with $h = 1024$ and let us solve it by the MMGS method with $\ell = 512$ on $ng \times nb = 512$ threads. The times with various choices of *ng* and *nb* are reported in Table 1 and plotted in Figure 1. As we solve the same problem with the same method, the final residual is $res = 9.56e-7$ and the number of iterations is $IT_{512} = 76$ in all cases. The difference in times is then solely attributable to the ratio between *ng* and *nb*.

TABLE 1 Times for various dimensions of CUDA grids (*ng*) and blocks (*nb*) for solving Example 1 by MMGS with $\ell = 512$ threads

ng,nb	**1,512**	**2,256**	**4,128**	**8,64**	**16,32**	**32,16**	**64,8**	**128,4**	**256,2**	**512,1**
T_ℓ	0.94	0.57	0.46	0.45	0.43	0.37	0.53	0.90	1.73	2.72

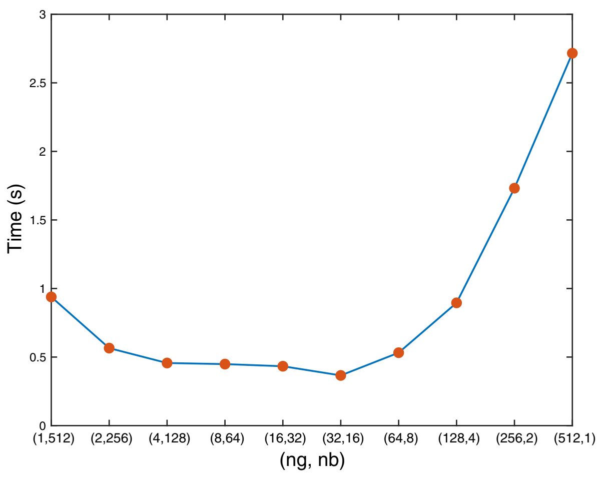

FIGURE 1 Plot of times for various dimensions of CUDA grids (*ng*) and blocks (*nb*) for solving Example 1 by MMGS with $\ell = 512$ threads

Elapsed time remains stable around 0.4 to 0.5 seconds when each block is made of 4 to 64 threads, but it readily increases when blocks have too many or too few threads. In particular, times double when we use a single block and increase more than 6 times when we use only one thread per block.

The above considerations are consistent with what commonly happens in many CUDA applications. Furthermore, they lead to the following operational indications that are followed in all numerical experiments performed hereafter:

- when $\ell \leq 32$, we use a grid of a single block ($ng = 1$) consisting of ℓ threads ($nb = \ell$);
- when $\ell > 32$, we use blocks of 32 threads ($nb = 32$). Consequently, assuming that ℓ is divisible by 32, the dimension of the grid is chosen as $ng = \ell/32$.

Of course, other choices can be easily done when ℓ is not divisible by 32 by exploiting the fact that times do not increase significantly for many choices of nb.

4.3 | Results and analysis

In this subsection, we present and analyze the results obtained by solving the considered test problems by various MM methods. In this context, we consider several choices of n and of ℓ. Evidently, when $\ell = 1$, a single splitting is used and all computations are performed on a single CUDA thread.

Figure 2 reports the parallel efficiency $\mathcal{E}_\ell$ of the MMJ, MMGS, and MMSOR methods for solving Example 1, Example 2, and Example 3, considering $h = 256$, $h = 512$, and $h = 1024$. In MMSOR, se set $\alpha = 1.1$. In figures, we also report T_1, which, combined with $\mathcal{E}_\ell$, allows to infer also all values of T_ℓ. Nonetheless, for completeness, we also report, as an example, the full numerical data obtained by the solution of Example 3 in Table 2. For clearness, in tables we report only the first decimal digits of T_ℓ. The values of $\mathcal{E}_\ell$ in tables and figures are, however, computed by considering all decimal digits, up to the resolution of `cudaEventElapsedTime`.

In all cases, all methods converge to the solution of the HLCPs, as expected. It is also fundamental to notice that in all cases the elapsed time T_ℓ is reduced when the number of processors ℓ is increased. Furthermore, the parallel efficiency remains larger than 0.7 in almost all cases when 64 or fewer threads are used, with some peaks that exceed a parallel efficiency of 1.

More specifically, in Example 1, with all solvers the computational efficiency slowly decreases as ℓ is increased, but it remains always larger than 0.9 when $\ell \leq 16$. Then, depending on the size of the problem, we may have a faster decrease of $\mathcal{E}_\ell$. Nonetheless, $\mathcal{E}_\ell$ does not become smaller than about 0.6 in all problems and for all solvers also when $\ell = 512$ is used. In Example 2 and in Example 3, we instead notice an increase of parallel efficiency, so that $\mathcal{E}_\ell$ exceeds 1 when ℓ is between 2 and 16. The improvement with respect to the efficiency of sequential procedures can be significant, as $\mathcal{E}_\ell$, in some cases, exceeds 1.12. Then, $\mathcal{E}_\ell$ tends to decrease, but it generally remains larger than 0.6 for $\ell \leq 128$ and in the order of $\mathcal{E}_\ell = 0.4$ to 0.5 when $\ell = 512$.

Furthermore, $\mathcal{E}_\ell$ does not present significant differences when the solution method is changed. Indeed, all methods show almost the same values of $\mathcal{E}_\ell$ in many cases.

Instead, the elapsed time before convergence T_ℓ changes significantly when the solution method is changed. In particular, MMJ is slower than the other considered methods, while MMSOR is the fastest. Indeed, in Figure 2 we see that T_1 is significantly larger for MMJ iterations than for the other methods. The MMGS and MMSOR methods remain faster also when ℓ is increased, as it can be easily verified by noticing that MMJ does not present a better parallel efficiency than MMGS and MMSOR. This is also evident in the complete data reported in Table 2 for Problem 3.

We also notice that T_ℓ is smaller for MMSOR than for MMGS, although the difference is generally quite small, especially when ℓ is large. The better efficiency of MMSOR was, nonetheless, expected, as the parameter α can be chosen as to increase the efficiency of MMSOR method.

Table 2 also affords some considerations on the number of iterations needed by the various methods to converge. First, we notice that the MMJ method requires significantly more iterations before convergence, explaining

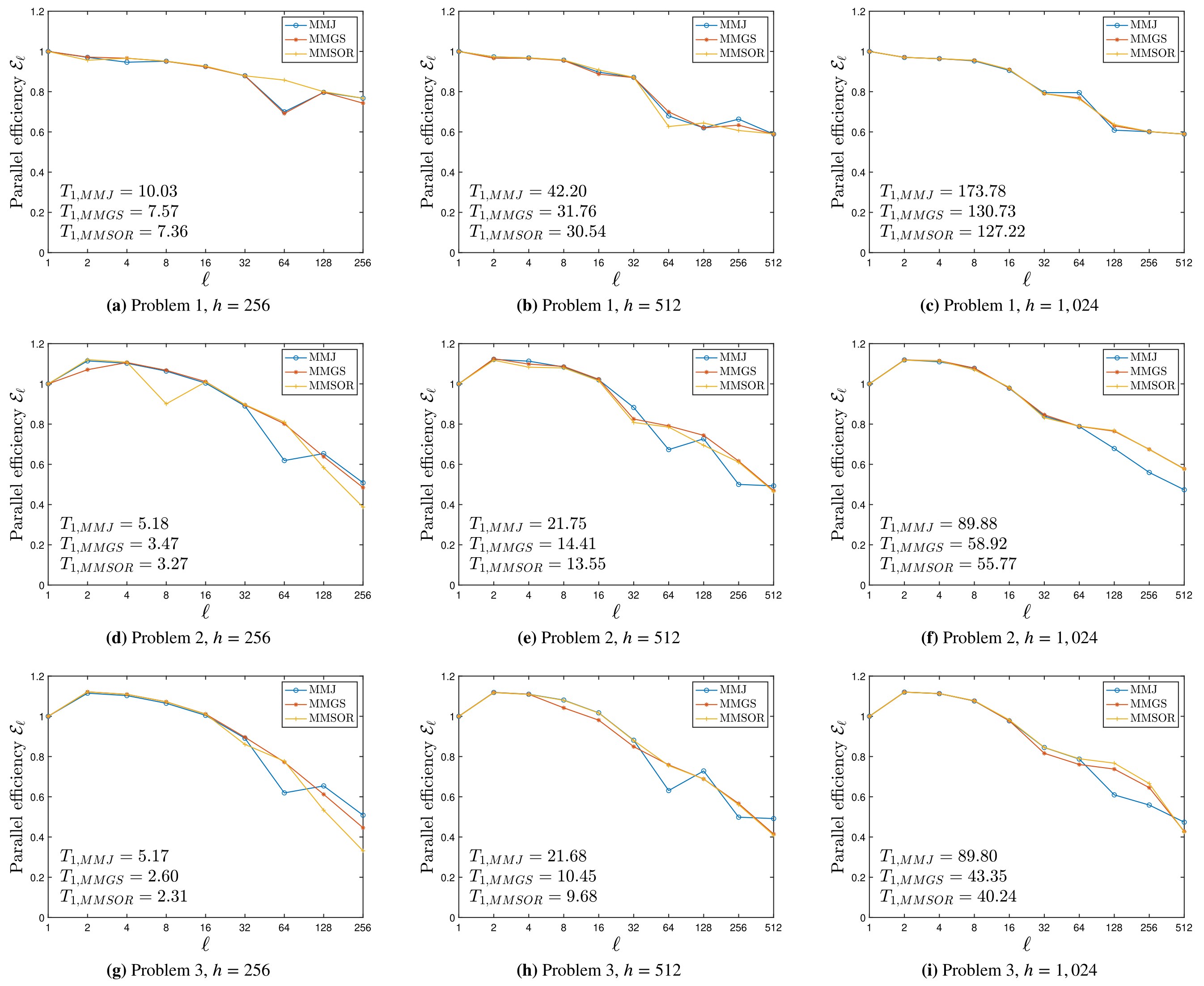

FIGURE 2 Parallel efficiency $\mathcal{E}_\ell$ and values of single-thread time T_1 for solving the considered problems by MMJ, MMGS, and MMSOR with various choices of problem dimension h^2 and of number of multisplittings (and processors) ℓ. MMGS, modulus-based multisplitting Gauss-Seidel; MMJ, modulus-based multisplitting Jacobi; MMSOR, modulus-based multisplitting SOR

why the elapsed time T_ℓ is larger for MMJ than for MMGS and MMSOR. The MMSOR method, in particular, requires less iterations than any other considered method, as expected. Second, we notice that the number of iterations of MMJ remains always constant, while the number of iterations of MMGS and MMSOR slightly increases when ℓ is large. This latter consideration is true also for Example 2, while, in Example 1, T_ℓ remains stable in all methods.

It is also interesting to notice that the above observations are consistent with what was observed in MM methods for LCPs in Reference 17.

Finally, for completeness, let us consider also a case with even larger dimensions. In particular, Table 3 shows an example of the results obtained by applying the MMGS iteration for solving Example 3 with $h = 2048$.

The obtained results demonstrate that, also in this case, we are able to solve the problem with good parallel efficiency (indeed, $\mathcal{E}_\ell$ is larger than 1 in some cases) and with a number of iterations that does not significantly increase with respect to the values reported in Table 2 for MMGS. In this context it is interesting to notice that parallelization allows to easily solve a large HLCP of dimension $n = 2048^2 = 4194304$ on a common desktop computer in about one second of parallel computations. Similar considerations can be applied also to the other problems and methods.

TABLE 2 Complete results for Example 3

h	Method	ℓ	1	2	4	8	16	32	64	128	256	512
256	MMJ	T_ℓ	5.17	2.32	1.17	0.61	0.32	0.18	0.13	0.06	0.04	–
		$\mathcal{E}_\ell$	1.00	1.11	1.10	1.06	1.00	0.89	0.62	0.65	0.51	–
		IT_ℓ	54	54	54	54	54	54	54	54	54	–
256	MMGS	T_ℓ	2.60	1.16	0.59	0.30	0.16	0.09	0.05	0.03	0.02	–
		$\mathcal{E}_\ell$	1.00	1.12	1.11	1.07	1.01	0.90	0.77	0.61	0.45	–
		IT_ℓ	27	27	27	27	27	27	28	29	31	–
256	MMSOR	T_ℓ	2.31	1.03	0.52	0.27	0.14	0.08	0.05	0.03	0.02	–
		$\mathcal{E}_\ell$	1.00	1.12	1.11	1.07	1.01	0.86	0.78	0.53	0.33	–
		IT_ℓ	24	24	24	24	24	25	25	27	29	
512	MMJ	T_ℓ	21.7	9.69	4.88	2.51	1.33	0.77	0.54	0.23	0.17	0.09
		$\mathcal{E}_\ell$	1.00	1.12	1.11	1.08	1.02	0.88	0.63	0.73	0.50	0.49
		IT_ℓ	56	56	56	56	56	56	56	56	56	56
512	MMGS	T_ℓ	10.5	4.67	2.36	1.25	0.67	0.38	0.22	0.12	0.07	0.05
		$\mathcal{E}_\ell$	1.00	1.12	1.11	1.04	0.98	0.85	0.76	0.69	0.57	0.41
		IT_ℓ	27	27	27	28	28	28	28	29	30	32
512	MMSOR	T_ℓ	9.68	4.33	2.18	1.12	0.60	0.34	0.20	0.11	0.07	0.05
		$\mathcal{E}_\ell$	1.00	1.12	1.11	1.08	1.02	0.88	0.76	0.69	0.56	0.41
		IT_ℓ	25	25	25	25	25	25	26	26	28	30
1024	MMJ	T_ℓ	89.8	40.1	20.2	10.4	5.74	3.32	1.78	1.15	0.63	0.37
		$\mathcal{E}_\ell$	1.00	1.12	1.11	1.08	0.98	0.85	0.79	0.61	0.56	0.47
		IT_ℓ	58	58	58	58	58	58	58	58	58	58
1024	MMGS	T_ℓ	43.4	19.3	9.74	5.03	2.77	1.66	0.89	0.46	0.26	0.20
		$\mathcal{E}_\ell$	1.00	1.12	1.11	1.08	0.98	0.82	0.76	0.74	0.65	0.43
		IT_ℓ	28	28	28	28	28	29	29	29	30	31
1024	MMSOR	T_ℓ	40.2	18.0	9.04	4.67	2.56	1.49	0.80	0.41	0.24	0.19
		$\mathcal{E}_\ell$	1.00	1.12	1.11	1.08	0.98	0.85	0.79	0.77	0.67	0.42
		IT_ℓ	26	26	26	26	26	26	26	26	27	29

Abbreviations: MMGS, modulus-based multisplitting Gauss-Seidel; MMJ, modulus-based multisplitting Jacobi; MMSOR, modulus-based multisplitting SOR.

5 | CONCLUSIONS

We have generalized modulus-based synchronous multisplitting methods to HLCPs. In particular, after a few preliminary remarks, we have defined the MM iteration for HLCPs considering a general splitting of the matrices of the problem and the special case of multisplitting AOR. This latter case (which includes multisplitting Jacobi, multisplitting Gauss-Seidel, and multisplitting SOR) is particularly interesting for parallel computations. Then, we have analyzed the convergence of the methods, providing conditions that ensure the global convergence of the procedures.

Finally, we have analyzed the behavior of the proposed methods in parallel computing frameworks by implementing the MMJ, the MMGS, and the MMSOR methods in Fortran with GPU parallelization of multisplitting procedures in CUDA. Numerical experiments have involved test problems of various dimension (always larger than the maximum dimension considered in Reference 28) and have been solved with several choices of the number of processors (and of simultaneous splittings) ℓ.

The obtained results have demonstrated that MM methods for HLCPs preserve good computational efficiency as ℓ increases and that their parallel implementation is able to substantially reduce the time needed for solving HLCPs. This

TABLE 3 Results for Example 3 with $h = 2048$ solved by MMGS

ℓ	1	2	4	8	16	32	64	128	256
T_ℓ	172.4	80.1	40.2	23.0	12.0	6.78	3.45	1.83	1.26
$\mathcal{E}_\ell$	1.00	1.08	1.07	0.94	0.90	0.79	0.78	0.74	0.53
IT_ℓ	29	29	29	29	29	29	29	30	30

Abbreviation: MMGS, modulus-based multisplitting Gauss-Seidel.

allowed to consider significantly larger problems, as well. For instance, we solved problems more than 400 times larger than the largest HLCP considered in Reference 28 ($n = 4194304$ vs $n = 10000$) in about 1 second of parallel computations using 256 GPU threads on an ordinary desktop computer.

CONFLICTS OF INTEREST

This work does not have any conflicts of interest.

ORCID

Francesco Mezzadri https://orcid.org/0000-0002-9779-7647

REFERENCES

1. Cottle RW, Pang J-S, Stone RE. The linear complementarity problem. Classics in applied mathematics. Philadelphia, PA: SIAM, 2009.
2. Bai Z-Z, Evans DJ. Matrix multisplitting methods with applications to linear complementarity problems: Parallel synchronous and chaotic methods. Réseaux et Systèmes Répartis: Calculateurs Parallèles. 2001;13(1):125–154.
3. Bai Z-Z, Evans DJ. Matrix multisplitting methods with applications to linear complementarity problems: Parallel asynchronous methods. Int J Comput Math. 2002;79(2):205–232.
4. Kojima M, Megiddo N, Mizuno S. A primal-dual infeasible-interior-point algorithm for linear programming. Math Program. 1993;61:263–280.
5. Simantiraki EM, Shanno DF. An infeasible-interior-point method for linear complementarity problems. SIAM J Optim. 1997;7(3):620–640.
6. Cryer CW. The solution of a quadratic programming problem using systematic overrelaxation. SIAM J Control. 1971;9:385–392.
7. Mangasarian OL. Solution of symmetric linear complementarity problems by iterative methods. J Optim Theory Appl. 1977;22:465–485.
8. Ahn BH. Solution of nonsymmetric linear complementarity problems by iterative methods. J Optim Theory Appl. 1981;33(2):175–185.
9. Bai Z-Z. On the monotone convergence of the projected iteration methods for linear complementarity problem. Numer Math J Chin Univ (English Ser). 1996;5(2):228–233.
10. Bai Z-Z. Modulus-based matrix splitting iteration methods for linear complementarity problems. Numer Linear Algebra Appl. 2010;17:917–933.
11. van Bokhoven WMG. Piecewise-linear modelling and analysis. Eindhoven, Netherlands: Proefschrift, 1981.
12. Murty K. Linear complementarity, linear and nonlinear programming. Berlin, Germany: Heldermann Verlag, 1988.
13. Dong J-L, Jiang M-Q. A modified modulus method for symmetric positive-definite linear complementarity problems. Numer Linear Algebra Appl. 2009;16:129–143.
14. Li W. A general modulus-based matrix splitting method for linear complementarity problems of H-matrices. Appl Math Lett. 2013;26:1159–1164.
15. Zheng N, Yin J-F. Accelerated modulus-based matrix splitting iteration methods for linear complementarity problems. Numer Algorithms. 2013;64:245–262.
16. Zheng H, Li W, Vong S. A relaxation modulus-based matrix splitting iteration method for solving linear complementarity problems. Numer Algorithms. 2017;74(1):137–152.
17. Bai Z-Z, Zhang L-L. Modulus-based synchronous multisplitting iteration methods for linear complementarity problems. Numer Linear Algebra Appl. 2013;20:425–439.
18. Bai Z-Z, Zhang L-L. Modulus-based synchronous two-stage multisplitting iteration methods for linear complementarity problems. Numer Algorithms. 2013;62:59–77.
19. Bai Z-Z, Zhang L-L. Modulus-based multigrid methods for linear complementarity problems. Numer Linear Algebra Appl. 2017;24:e2105:1–e2105:15.
20. Ren H, Wang X, Tang X-B, Wang T. The general two-sweep modulus-based matrix splitting iteration method for solving linear complementarity problems. Comput Math Appl. 2019;77(4):1071–1081.
21. Dai P-F, Li J, Bai J, Qiu J. A preconditioned two-step modulus-based matrix splitting iteration method for linear complementarity problem. Appl Math Comput. 2019;348:542–551.

22. Mezzadri F. On the equivalence between some projected and modulus-based splitting methods for linear complementarity problems. Calcolo. 2019;56(41):1–28.
23. Xia Z, Li C. Modulus-based matrix splitting iteration methods for a class of nonlinear complementarity problems. Appl Math Comput. 2015;271:34–42.
24. Zheng H, Vong S, Liu L. The relaxation modulus-based matrix splitting iteration method for solving a class of nonlinear complementarity problems. Int J Comput Math. 2019;96(8):1648–1667.
25. Zheng H, Vong S. A modified modulus-based matrix splitting iteration method for solving implicit complementarity problems. Numer Algorithms. 2019;82(2):573–592.
26. Zheng H, Liu L. The sign-based methods for solving a class of nonlinear complementarity problems. J Optim Theory Appl. 2019;180(2):480–499.
27. Li R, Wang Y, Yin J-F. On the convergence of two-step modulus-based matrix splitting iteration methods for a restricted class of nonlinear complementarity problems with H_+-matrices. Numer Math Theory Methods Appl. 2018;11:128–139.
28. Mezzadri F, Galligani E. Modulus-based matrix splitting methods for horizontal linear complementarity problems. Numer Algorithms. 2020;83(1):201–219.
29. Gowda MS. Reducing a monotone horizontal LCP to an LCP. Appl Math Lett. 1995;8(1):97–100.
30. Tütüncü RH, Todd MJ. Reducing horizontal linear complementarity problems. Linear Algebra Appl. 1995;223/224:717–729.
31. Zhang Y. On the convergence of a class on infeasible interior-point methods for the horizontal linear complementarity problem. SIAM J Optim. 1994;4(1):208–227.
32. Mezzadri F, Galligani E. An inexact Newton method for solving complementarity problems in hydrodynamic lubrication. Calcolo. 2018;55(1):1–28.
33. Ralph D. A stable homotopy approach to horizontal linear complementarity problems. Control Cybernet. 2002;31:575–600.
34. Gao X, Wang J. Analysis and application of a one-layer neural network for solving horizontal linear complementarity problems. Int J Comput Intell Syst. 2014;7(4):724–732.
35. Mezzadri F, Galligani E. Splitting methods for a class of horizontal linear complementarity problems. J Optim Theory Appl. 2019;180:500–517.
36. Zheng H, Vong S. On convergence of the modulus-based matrix splitting iteration method for horizontal linear complementarity problems of H_+-matrices. Appl Math Comput. 2020;369:1–6.
37. Mezzadri F, Galligani E. A modulus-based nonsmooth Newton's method for solving horizontal linear complementarity problems. Optim Lett. 2019. https://doi.org/10.1007/s11590-019-01515-9.
38. O'Leary DP, White RE. Multi-splittings of matrices and parallel solution of linear systems. SIAM J Alg Discr Methods. 1985;6:630–640.
39. Bai Z-Z, Sun J-C, Wang D-R. A unified framework for the construction of various matrix multisplitting iterative methods for large sparse system of linear equations. Comput Math Appl. 1996;32(12):51–76.
40. Bai Z-Z, Evans DJ. Matrix multisplitting relaxation methods for linear complementarity problems. Int J Comput Math. 1997;63(3-4):309–326.
41. Machida N, Fukushima M, Ibaraki T. A multisplitting method for symmetric linear complementarity problems. J Comput Appl Math. 1995;62:217–227.
42. Bai Z-Z. On the convergence of the multisplitting methods for the linear complementarity problem. SIAM J Matrix Anal Appl. 1999;21:67–78.
43. Cook S. CUDA programming: A developer's guide to parallel computing with GPUs. San Francisco, CA: Morgan Kaufmann Publishers Inc, 2012.
44. Frommer A, Szyld DB. H-splittings and two-stage iterative methods. Numer Math. 1992;63:345–356.
45. Frommer A, Meyer G. Convergence of relaxed parallel multisplitting methods. Linear Algebra Appl. 1989;119:141–152.
46. Berman A, Plemmons RJ. Nonnegative matrices in the mathematical sciences: Classics in applied mathematics. Philadelphia, PA: SIAM, 1994.
47. Varga RS. Matrix iterative analysis. 2nd ed. Berlin, Germany: Springer, 2000.

How to cite this article: Mezzadri F. Modulus-based synchronous multisplitting methods for solving horizontal linear complementarity problems on parallel computers. *Numer Linear Algebra Appl*. 2020;e2319. https://doi.org/10.1002/nla.2319