STATIONARY SPLITTING ITERATIVE METHODS FOR THE MATRIX EQUATION $AXB = C$

ZHONGYUN LIU*, ZHEN LI*, CARLA FERREIRA†, AND YULIN ZHANG†

Abstract. Stationary splitting iterative methods for solving $AXB = C$ are considered in this paper. The main tool to derive our new method is the induced splitting of a given nonsingular matrix $A = \mathcal{M} - \mathcal{N}$ by a matrix $\mathcal{H}$ such that $(I - \mathcal{H})^{-1}$ exists. Convergence properties of the proposed method are discussed and numerical experiments are presented to illustrate its computational efficiency and the effectiveness of some preconditioned variants. In particular, for certain surface-fitting applications our method is much more efficient than the **progressive iterative approximation** (PIA), a conventional iterative method often used in computer aided geometric design (CAGD).

Key words. Hermitian positive definite, H-matrices, stationary splitting iteration, induced splitting,curves fitting.

AMS subject classifications. 15A23, 65D17, 65F10, 65F15.

1. Introduction.

Consider the matrix equation

$$AXB = C, \tag{1.1}$$

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{m\times m}$ and $X, C \in \mathbb{R}^{n\times m}$. This matrix equation plays an important role in many practical applications such as surfaces fitting in computer aided geometric design (CAGD), signal and image processing, photogrammetry, etc., see for example [14, 19, 20, 30, 31] and a large literature therein. Given the existence of many and important applications, the theory and algorithms for matrix equations have intrigued the researchers for decades, see [7, 9, 15, 21, 28, 29, 30, 31, 34, 37]. A recent survey on the major numerical algorithms for the solution of the more general linear matrix equation $AXE + DXB = C$, for given square matrices A, B, D, E and conformable matrix C, and related problems is presented in [33].

The matrix equation (1.1) can be treated as a standard linear system with multiple right-hand sides as pointed out in [33, p.380]. More precisely, it is possible, for instance, to split the initial problem into two matrix equations, $AY = C$ and $B^T X^T = Y^T$, to be considered in succession. In practice, instead of solving each of the linear systems independently, by using the LU decomposition method or some

*School of Mathematics and Statistics, Changsha University of Science and Technology, Changsha 410076, P. R. China (liuzhongyun@263.net). ,

†Centro de Matemática, Universidade do Minho, 4710-057 Braga, Portugal (Carla Ferreira: caferrei@math.uminho.pt, Yulin Zhang: zhang@math.uminho.pt).

iterative method, it is more efficient to use a block solver, especially when the matrices involved are dense. See [13, 32, 38].

Equation (1.1) is also mathematically equivalent to the larger linear system

$$(B^T \otimes A)\mathbf{x} = \mathbf{c}, \tag{1.2}$$

where $\otimes$ denotes the standard Kronecker product symbol, $\mathbf{x} = \mathbf{vec}(X)$ and $\mathbf{c} = \mathbf{vec}(C)$ are two column-stacking vectors of the matrices X and C, respectively. The order of the coefficient matrix in (1.2) is mn, which becomes very large when m, n are large.

It is useful to consider the general linear system (1.2) in a theoretical analysis but it is impractical to solve this system to obtain a numerical solution to (1.1), since it is computationally expensive and it can be ill-conditioned. Thus, it is attractive to get the solution for (1.1) from itself. Moreover, the Level 3 BLAS operations can be performed on high performance computers [8].

In this paper we focus our attention in methods developed directely from the original matrix equation (1.1). There are two main types of methods for solving this problem: direct methods and iterative methods. When the matrices A and B are small and dense, direct methods such as QR-factorization-based algorithms [9, 36] are attractive. However, these direct algorithms are quite costly and suffer from numerical instability when A and B are large and ill-conditioned. Therefore, considerable attention has recently been paid to iterative methods, see for instance [15, 28, 29, 34, 35, 37] and references therein.

In the context of this problem, Tian et al. in [34] developed some Jacobi and Gauss-Seidel-type iterative methods for solving (1.2). Let us roughly summarize their schemes as follows in terms of the original problem (1.1).

We say that the representation $\mathcal{A} = \mathcal{M} - \mathcal{N}$ is a *splitting* of $\mathcal{A}$ if $\mathcal{M}$ is nonsingular. The splitting $\mathcal{A} = \mathcal{M} - \mathcal{N}$ is called *convergent* if $\rho(\mathcal{H}) < 1$, where $\mathcal{H} = \mathcal{M}^{-1}\mathcal{N}$ is the iteration matrix and $\rho(\mathcal{H})$ denotes the spectral radius of the matrix $\mathcal{H}$.

Let $A = F - G$ and $B = \hat{F} - \hat{G}$ be the splittings of A and B, respectively. Then the four basic iteration sequences defined in [34] can be equivalently written as follows:

$$X^{(k+1)} = F^{-1}GX^{(k)} + F^{-1}CB^{-1}, \tag{1.3}$$

$$X^{(k+1)} = X^{(k)}\hat{G}\hat{F}^{-1} + A^{-1}C\hat{F}^{-1}, \tag{1.4}$$

$$X^{(k+1)} = F^{-1}GX^{(k)} + X^{(k)}\hat{G}\hat{F}^{-1} - F^{-1}GX^{(k)}\hat{G}\hat{F}^{-1} + F^{-1}C\hat{F}^{-1} \tag{1.5}$$

and

$$X^{(k+1)} = X^{(k)} + F^{-1}\big(C - AX^{(k)}B\big)\hat{F}^{-1}. \tag{1.6}$$

The iterations (1.3) and (1.4) are referred as modified iterations and some of their convergence theories as well as related algorithms were discussed in [34].

In fact, if A^{-1} or B^{-1} is available or it is easy to obtain, then the matrix equation (1.1) can be reduced into a general linear system with multiple right-hand sides. In this situation, iteration sequence (1.3) or (1.4) can be employed to obtain an approximate solution to (1.1), otherwise they are useless in practice. We therefore focus on iterations (1.5) and (1.6). We refer to iteration (1.5) as the direct splitting iteration and to iteration (1.6) as the residual-updating iteration (cf. [2]).

It is known that the iteration sequences (1.3) and (1.4) converge to the exact solution of (1.1) if and only if $\rho(F^{-1}G) < 1$ and $\rho(\hat{G}\hat{F}^{-1}) < 1$, respectively. In what respects to the iteration sequence (1.5), this condition is not suficient for convergence. It was shown in [21], exhibiting a numerical example, that even if both $\rho(F^{-1}G) < 1$ and $\rho(\hat{G}\hat{F}^{-1}) < 1$ hold simultaneously, the method may not converge. In this paper we recall the sufficent conditions for convergence of (1.5) that were established in [21] (see Theorem 2.10) and develop a new iterative scheme that also deals with the cases when convergence is not ensured immediatley.

The splittings considered in the iterative methods described in [34] were the particular splittings of Jacobi and Gauss-Seidel of the matrix $B^T \otimes A$. The method we propose works with the original matrices A and B, whose size is much smaller if m and n are large, and it can be applied to any convergent splitting of these matrices. Thus, our method is broader in what respects to the type of splitting that may be considered and enables an improvement of the performance speed if m and n are large.

In our numerical experiments we first consider Gauss-Seidel splittings of A and B and antecipate possible divergence, even if conditions $\rho(F^{-1}G) < 1$ and $\rho(\hat{G}\hat{F}^{-1}) < 1$ both hold. These cases where not considered in [34] and, thus, with these numerical results we intend to give a complementary study to [34] rather than to provide a comparison of the different methods.

We present the details of the implementation of our method in Matlab and in our numerical studies we also consider sucessive overrelaxation Gauss-Seidel (SOR) splittings and preconditioned versions of our method.

This paper is organized as follows. In the next section we recall some preliminary results used in the sequel. Then, in Section 3, we introduce stationary splittings and discuss the convergence of the resulting induced splitting iteration. Some preconditioned variants of this iteration are also presented in Section 3. The first numerical experiments are presented in Section 4 to exhibit the effectiveness of our method. An example of an application to surface-fitting is described in Section 5 and more numerical results are shown. A brief conclusion is drawn in Section 6, and the acknowledgements are writen in the last section.

2. Preliminaries.

In this section, we review some definitions, notation and known results needed in the remaining parts of this paper.

Definition 2.1. ([3, 14]) *A matrix $\mathcal{A} = (a_{ij}) \in \mathbb{R}^{n\times n}$ is called*

- *a Z-matrix, if $a_{ij} \leq 0$, for $i, j = 1, \ldots, n$, $i \neq j$;*
- *a monotone matrix, if $\mathcal{A}^{-1} \geq 0$;*
- *an M-matrix, if $\mathcal{A}$ is a monotone Z-matrix.*

Definition 2.2. ([10]) *Let $\mathcal{A} = (a_{ij}) \in \mathbb{C}^{n\times n}$. Let $\langle\mathcal{A}\rangle = (\alpha_{ij}) \in \mathbb{R}^{n\times n}$ be defined by the following conditions:*

- *$\alpha_{ii} = |a_{ii}|$ for $i = 1, \ldots, n$;*
- *$\alpha_{ij} = -|a_{ij}|$ for $i, j = 1, \ldots, n$, $i \neq j$.*

Matrix $\langle\mathcal{A}\rangle$ is called the comparison matrix of $\mathcal{A}$.

Definition 2.3. ([10]) *Let $\mathcal{A} = (a_{ij}) \in \mathbb{C}^{n\times n}$. If its comparison matrix $\langle\mathcal{A}\rangle$ is an M-matrix, then $\mathcal{A}$ is called an H-matrix.*

Definition 2.4. ([3, 14, 27]) *Let $\mathcal{A} \in \mathbb{R}^{n\times n}$. The splitting $\mathcal{A} = \mathcal{M} - \mathcal{N}$ is called*

- *weak regular if $\mathcal{M}^{-1} \geq 0$ and $\mathcal{M}^{-1}N \geq 0$;*
- *regular if $\mathcal{M}^{-1} \geq 0$ and $\mathcal{N} \geq 0$;*
- *P-regular if $\mathcal{M}^* + \mathcal{N}$ is positive definite;*
- *M-splitting if $\mathcal{M}$ is an M-matrix, and $\mathcal{N} \geq 0$;*
- *H-splitting if $\langle\mathcal{M}\rangle - |\mathcal{N}|$ is an M-matrix;*
- *H-compatible splitting if $\langle\mathcal{A}\rangle = \langle\mathcal{M}\rangle - |\mathcal{N}|$.*

The following results about matrix splittings can be found in [3, 10, 14, 24, 27].

Theorem 2.5. *Let $\mathcal{A} = \mathcal{M} - \mathcal{N}$ be an hermitian positive definite matrix, where $\mathcal{M}$ is a invertible hermitian matrix. Then $\rho(\mathcal{M}^{-1}\mathcal{N}) < 1$ if and only if the splitting $\mathcal{A} = \mathcal{M} - \mathcal{N}$ is P-regular.*

Theorem 2.6. *Let $\mathcal{A}$ be hermitian, and let the splitting $\mathcal{A} = \mathcal{M} - \mathcal{N}$ be P-regular. Then $\rho(\mathcal{M}^{-1}\mathcal{N}) < 1$ if and only if $\mathcal{A}$ is positive definite.*

Lemma 2.7. *Let $\mathcal{A} \in \mathbb{R}^{n\times n}$ be a monotone matrix. If $\mathcal{A} = \mathcal{M} - \mathcal{N}$ is a weak regular splitting, then $\rho(\mathcal{M}^{-1}\mathcal{N}) < 1$.*

Lemma 2.8. *Let $\mathcal{A}$ be an $n \times n$ H-matrix.*

- *If the splitting $\mathcal{A} = \mathcal{M} - \mathcal{N}$ is an H-splitting, then $\rho(\mathcal{M}^{-1}\mathcal{N}) < 1$.*
- *If the splitting $\mathcal{A} = \mathcal{M} - \mathcal{N}$ is an H-compatible splitting, then it is an H-splitting and thus convergent.*

We remark here that an M-matrix is certainly an H-matrix but the converse is not true, and that all strictly or irreducibly diagonally dominant matrices are H-matrices.

The following conclusion is often used in the analysis of iterative methods.

LEMMA 2.9. ([17, 24]). *Given a nonsingular matrix $\mathcal{A}$ and $\mathcal{H}$ such that $(I-\mathcal{H})^{-1}$ exists, there exists a unique pair of matrices $\mathcal{M}_{\mathcal{H}}$, $\mathcal{N}_{\mathcal{H}}$, such that $\mathcal{H} = \mathcal{M}_{\mathcal{H}}^{-1}\mathcal{N}_{H}$ and $\mathcal{A} = \mathcal{M}_{\mathcal{H}} - \mathcal{N}_{\mathcal{H}}$, where $\mathcal{M}_{\mathcal{H}}$ is nonsingular. It is said that $\mathcal{A} = \mathcal{M}_{\mathcal{H}} - \mathcal{N}_{\mathcal{H}}$ is an induced splitting of $\mathcal{A}$ by $\mathcal{H}$.*

The following theorem concerns the convergence of iteration sequence (1.5).

THEOREM 2.10. ([21, Theorem 2.8]) *Let $A = F - G$ and $B = \hat{F} - \hat{G}$ be two splittings of A and B, respectively. Let $H = F^{-1}G$ and $\hat{H} = \hat{G}\hat{F}^{-1}$. If*

$$\rho(H) < \sqrt{3} - 1, \quad \rho(\hat{H}) < \sqrt{3} - 1 \tag{2.1}$$

and

$$[\rho(H) + 1]^2 + [\rho(\hat{H}) + 1]^2 < 4, \tag{2.2}$$

then iteration sequences (1.5) and (1.6) converge to the exact solution of (1.1).

3. The induced splitting iteration.

In this section we develop a new iterative method for solving the matrix equation (1.1) and analyze the convergence properties of this new method. A preconditioned variant of this method will also be described.

3.1. Iterative scheme and convergence results.

We first introduce induced splittings of the matrices A and B.

Let the splittings $A = F - G$ and $B = \hat{F} - \hat{G}$ be convergent splittings. Defining

$$H = F^{-1}G, \quad \tilde{H} = \hat{F}^{-1}\hat{G} \quad \text{and} \quad \hat{H} = \hat{G}\hat{F}^{-1},$$

we have $\rho(H) < 1$ and $\rho(\hat{H}) = \rho(\tilde{H}) < 1$, since, for any two square matrices S and T, ST and TS have the same eigenvalues.

Using the above notation, iteration (1.5) may be written as

$$X^{(k+1)} = HX^{(k)} + X^{(k)}\hat{H} - HX^{(k)}\hat{H} + F^{-1}C\hat{F}^{-1}. \tag{3.1}$$

If this iteration sequence does not converge (conditions (2.1) and (2.2) in Theorem 2.10 are not satisfied simultaneously), then we proceed as follows.

We first estimate $\rho = \rho(H)$ and $\hat{\rho} = \rho(\hat{H})$ using the power iteration. If $\rho < 1$, then the sequence ρ^p converges to 0 with p and, thus, for any arbitrarily small $\varepsilon_1 > 0$, there exists a positive integer $\mathbb{N}_1$ such that $\rho^p \le \varepsilon_1$, when $p \ge \mathbb{N}_1$. Let $p \ge \mathbb{N}_1$ be fixed and let $\mathcal{H} = H^p$. Then $\rho(\mathcal{H}) = \rho^p < 1$ and $I - \mathcal{H}$ is invertible. By Lemma 2.9, this $\mathcal{H}$ induces a unique splitting of A, that is,

$$\begin{cases} A = \mathcal{M} - \mathcal{N}, \\ \mathcal{M}^{-1} = \left(\sum\limits_{i=0}^{p-1} H^i\right) F^{-1} \quad \text{with} \quad \mathcal{M}^{-1}\mathcal{N} = \mathcal{H}. \end{cases} \tag{3.2}$$

Observe that, since this splitting of A yields $A^{-1} = (I - \mathcal{H})^{-1}\mathcal{M}^{-1}$ and the first splitting of A gives $A^{-1} = (I - H)^{-1}F^{-1}$, we obtain

$$\mathcal{M}^{-1} = (I - \mathcal{H})A^{-1} = (I - H^p)(I - H)^{-1}F^{-1} = (I + H + \cdots + H^{p-1})F^{-1}.$$

This formula for $\mathcal{M}^{-1}$ can also be directly derived from a two-stage iteration for solving $\mathcal{A}x = b$ with outer splitting $\mathcal{A} = \mathcal{A} - O$, where O denotes the zero matrix, see [6, 10, 17, 24].

We refer to (3.2) as a ***p*-degree induced splitting** of A by H.

Similarly, if $\hat{\rho} < 1$, then, for any arbitrarily small $\varepsilon_2 > 0$, there exists a positive integers $\mathbb{N}_2$ such that $\hat{\rho}^q \le \varepsilon_2$, when $q \ge \mathbb{N}_2$. Let $\tilde{\mathcal{H}} = \tilde{H}^q$ and $\hat{\mathcal{H}} = \hat{H}^q$ for a fixed $q \ge \mathbb{N}_2$. Then $\rho(\tilde{\mathcal{H}}) = \rho(\hat{\mathcal{H}}) = \hat{\rho}^q < 1$ and, by Lemma 2.9, analogously to what was considered for A, matrices $\tilde{\mathcal{H}}$ and $\hat{\mathcal{H}}$ induce unique splittings of B, given, respectively, by

$$\begin{cases} B = \tilde{\mathcal{M}} - \tilde{\mathcal{N}} \\ \tilde{\mathcal{M}}^{-1} = \left(\sum\limits_{i=0}^{q-1} \tilde{H}^i\right) \hat{F}^{-1} \quad \text{with} \quad \tilde{\mathcal{M}}^{-1}\tilde{\mathcal{N}} = \tilde{\mathcal{H}}, \end{cases} \tag{3.3}$$

and

$$\begin{cases} B = \hat{\mathcal{M}} - \hat{\mathcal{N}}, \\ \hat{\mathcal{M}}^{-1} = \hat{F}^{-1} \sum\limits_{i=0}^{q-1} \hat{H}^i \quad \text{with} \quad \hat{\mathcal{N}}\hat{\mathcal{M}}^{-1} = \hat{\mathcal{H}}. \end{cases} \tag{3.4}$$

Observe that the splitting (3.4) is derived from the relations $B^{-1} = \hat{\mathcal{M}}^{-1}(I - \hat{\mathcal{H}})^{-1}$ and $B^{-1} = \hat{M}^{-1}(I - \hat{H})^{-1}$, both combined to obtain the expression for $\hat{\mathcal{M}}^{-1}$. Also observe that $\hat{\mathcal{M}}^{-1} = \tilde{\mathcal{M}}^{-1}$.

The splittings (3.3) and (3.4) are referred to as ***q*-degree induced splittings** of B, by $\tilde{H}$ and $\hat{H}$, respectively.

Now, we can use the induced splittings (3.2) and (3.4) to construct the iteration schemes corresponding to (3.1),

$$X^{(k+1)} = \mathcal{H}X^{(k)} + X^{(k)}\hat{\mathcal{H}} - \mathcal{H}X^{(k)}\hat{\mathcal{H}} + \mathcal{M}^{-1}C\hat{\mathcal{M}}^{-1}, \tag{3.5}$$

and the iteration scheme corresponding to (1.6),

$$X^{(k+1)} = X^{(k)} + \mathcal{M}^{-1}\big(C - AX^{(k)}B\big)\hat{\mathcal{M}}^{-1}. \tag{3.6}$$

We remark here that if $p = q = 1$, then iterations (3.5) and (3.6) reduce to (3.1) and and (1.6), respectively.

We refer to iterations (3.5) and (3.6) as **induced splitting iterations**.

Going to a numerical implementation, we chose to develop the algorithm for the induced splitting iteration (3.6), since it is computationally more efficient. If we let

$$A_1 = \mathcal{M}^{-1}A, \qquad B_1 = B\hat{\mathcal{M}}^{-1}, \qquad C_1 = \mathcal{M}^{-1}C\hat{\mathcal{M}}^{-1}, \tag{3.7}$$

iteration (3.6) becomes

$$X^{(k+1)} = X^{(k)} + \big(C_1 - A_1X^{(k)}B_1\big), \tag{3.8}$$

resulting that only two matrix multiplications are needed per iteration.

We are now in conditions to write the arguments we discussed above as a convergence theorem for iterations (3.6) (and (3.5)).

Theorem 3.1. *Consider convergent splittings $A = F - G$ and $B = \hat{F} - \hat{G}$. Let $H = F^{-1}G$, $\hat{H} = \hat{G}\hat{F}^{-1}$, $\rho = \rho(H)$ and $\hat{\rho} = \rho(\hat{H})$. Then there exist two positive integers $\mathbb{N}_1$ and $\mathbb{N}_2$ such that ρ^p and $\hat{\rho}^q$ satisfy (2.1) and (2.2) for all $p \geq \mathbb{N}_1$ and $q \geq \mathbb{N}_2$. In this case, the iteration (3.6) converges to the exact solution of (1.1), for any given guess $X^{(0)}$.*

Proof. It follows from the hypothesis that $\rho = \rho(H) < 1$ and $\hat{\rho} = \rho(\hat{H}) < 1$. Thus, given any two positive integers p and q, for $\mathcal{H} = H^p$ and $\hat{\mathcal{H}} = \hat{H}^q$, we have $\rho(\mathcal{H}) = \rho^p < 1$, $\rho(\hat{\mathcal{H}}) = \hat{\rho}^q < 1$ and, by Lemma 2.9, $\mathcal{H}$ and $\hat{\mathcal{H}}$ induce the splittings $A = \mathcal{M} - \mathcal{N}$ and $B = \hat{\mathcal{M}} - \hat{\mathcal{N}}$, given by (3.2) and (3.4), respectively. We also know that there always exist positive integers $p = \mathbb{N}_1$ and $q = \mathbb{N}_2$ for which $\rho(\mathcal{H})$ and $\rho(\hat{\mathcal{H}})$ are arbitrarily small, for all $p \geq \mathbb{N}_1$ and $q \geq \mathbb{N}_2$, and then satisfy (2.1) and (2.2). By Theorem 2.10, we therefore conclude that the iteration (3.6) converges to the exact solution of (1.1), for any given guess $X^{(0)}$. $\square$

As a consequence of Theorem 3.1, we have the following result.

Corollary 3.2. *Let A and B be Hermitian positive definite or H-matrices. Let $A = F - G$ and $B = \hat{F} - \hat{G}$ be their Gauss-Seidel splittings*[*]*, let $H = F^{-1}G$, $\hat{H} = \hat{G}\hat{F}^{-1}$, and let $\rho = \rho(H)$ and $\hat{\rho} = \rho(\hat{H})$. Then there exist two positive integers*

[*]The splitting $A = (D - L) - U$ is the Gauss-Seidel splitting of A, where D is the diagonal matrix consisting of the diagonal entries of A, L and U are the strictly lower triangular and the strictly upper triangular parts of A, respectively.

$\mathbb{N}_1$ and $\mathbb{N}_2$ such that ρ^p and $\hat{\rho}^q$ satisfy (2.1) and (2.2) for all $p \geq \mathbb{N}_1$ and $q \geq \mathbb{N}_2$. In this case, the iteration (3.6) converges to the exact solution of (1.1), for any given guess $X^{(0)}$.

Proof. It is sufficient to show that $\rho < 1$ and $\hat{\rho} < 1$. If A and B are Hermitian positive definite and the splittings $A = F - G$ and $B = \hat{F} - \hat{G}$ are the Gauss-Seidel splittings, these are P-regular splittings. Then, by Theorem 2.5 or Theorem 2.6, we have $\rho < 1$ and $\hat{\rho} < 1$. Similarly, If A and B are H-matrices, and the splittings $A = F - G$ and $B = \hat{F} - \hat{G}$ are the Gauss-Seidel splittings, these are H-compatible splittings. Then, by Lemma 2.8, we also have $\rho < 1$ and $\hat{\rho} < 1$. □

We may now present our algorithm based on iteration (3.8). Observe that in this scheme an estimate for the error $E^{(k)} = X^{(k+1)} - X^{(k)}$ is immediately available.

Algorithm 1. *An approximated solution to* (1.1) *is obtained in two stages.*

Preparatory stage:

- *determine the splittings $A = F - G$ and $B = \hat{F} - \hat{G}$ (SOR splittings, for example) and then compute $H = F^{-1}G$ and $\hat{H} = \hat{G}\hat{F}^{-1}$;*
- *obtain estimates $\rho = \rho(H)$ and $\hat{\rho} = \rho(\hat{H})$ and search for two positive integers p and q such that ρ^p and $\hat{\rho}^q$ satisfy (2.1) and (2.2);*
- *once some p and q are determined, compute $\mathcal{M}^{-1}$ and $\hat{\mathcal{M}}^{-1}$ given in (3.2) and (3.4), respectively;*
- *compute A_1, B_1 and C_1 according to* (3.7).

Iterative stage: *Given an initial guess $X^{(0)}$, for $k = 0, 1, 2, \cdots$, until $\{X^{(k)}\}$ converges,*

- *compute $E^{(k)} = C_1 - A_1 X^{(k)} B_1$ and $X^{(k+1)} = X^{(k)} + E^{(k)}$;*
- *compare $\|E^{(k)}\|$ with the desired error tolerance.*

In Section 4 we describe some details considered in our Matlab implementation of Algorithm 1 in which we included the SOR splittings.

We remark here that if A and B are Hermitian positive definite or H-matrices, then a block version of Algorithm 1 with block Gauss-Seidel splittings can be easily developed, which is appealing for solving large matrix equations on parallel architectures, see [1].

3.1.1. Finding p and q. When we find p and q that satisfy (2.1) and (2.2), then we can also consider greater values and the greater p and q are, the smaller the spectral radii $\rho(\mathcal{H})$ and $\rho(\hat{\mathcal{H}})$ are and thus the smaller the number of iterations is (required for Algorithm 1 to converge for a given tolerance). However, increasing p and q means increasing the computational complexity (the number of matrix-matrix multiplications) to obtain $\mathcal{M}^{-1}$ and $\hat{\mathcal{M}}^{-1}$ which will only be compensated by the reduction of the number of iterations up to a certain level, namely if the orders m and n of the matrices A and B are not small.

Thus, it is important to consider the question: for given matrices A and B and convergent splittings of these matrices, are there optimal values for p and q? Certainly, the optimal values, if they exist, will depend on the specific structures and orders of the matrices A and B as well as on the properties of the splittings considered. We regard that a theoretical study of this question is beyond the scope of this paper. We tested different choices for p and q and there is computational evidence that the procedure we next describe gives very good results.

Assuming $\rho = \rho(H) < 1$ and $\rho(\hat{H}) = \hat{\rho} < 1$, we solve

$$\rho^p < \sqrt{3} - 1 \quad \text{and} \quad \hat{\rho}^q < \sqrt{3} - 1, \tag{3.9}$$

for p and q, and take the smallest positive integers p_1 and q_1 that satisfy these inequalities. If condition (2.2) for ρ^{p_1} and $\hat{\rho}^{q_1}$,

$$(\rho^{p_1} + 1)^2 + (\hat{\rho}^{q_1} + 1)^2 < 4, \tag{3.10}$$

is not satisfied, we let p_1 and q_1 be set to $p_1 + 1$ and $q_1 + 1$, respectively, and test condition (3.10) again. We repeat this step until condition (3.10) is satisfied.

Alternatively to this procedure, in the case $\rho \approx \hat{\rho}$ or $\hat{\rho} < \rho$, we could consider solving for p the inequality

$$\left(\rho^p + 1\right)^2 < 2,$$

and take the smallest positive integer solution p_2. Then we would let $p = \max\{p_1, p_2\}$ and $q = \max\{q_1, p_2\}$. Condition (3.10) should be verified (approximately).

Analogously, if $\rho < \hat{\rho}$, we would consider

$$\left(\hat{\rho}^q + 1\right)^2 < 2,$$

find the smallest positive integer solution q_2 and let $p = \max\{p_1, q_2\}$ and $q = \max\{q_1, q_2\}$.

The important observation here is that the difference between p and q will come from conditions (3.9) and not from condition (3.10).

In Section 4 we report some numerical experiments exhibiting the CPU time needed for convergence for different values of p and q in Example 4.2.

3.1.2. Sucessive overrelaxation splittings. In the sucessive overrelaxation Gauss-Seidel (SOR) iterative method, we introduce a parameter ω, the relaxation parameter, which often makes an improvement in the rate of convergence. The value of this parameter should be chosen so that the rate of convergence is maximized, but the optimal value of ω is known only for some special classses of matrices. In the case of positive definiteness it is known that $0 < \omega < 2$ is a necessary and sufficent condition for convergence.

In the preparatory stage of Algorithm 1, for a given value of $0 < \omega < 2$, we consider the matrix equation

$$(\omega A)X(\omega B) = \omega^2 C$$

and the splittings

$$\begin{aligned}\omega A &= F - G = (D + \omega L) - [D - \omega(D + U)],\\ \omega B &= \hat{F} - \hat{G} = (\hat{D} + \omega\hat{L}) - [\hat{D} - \omega(\hat{D} + \hat{U})]\end{aligned}$$

for $A = D + L + U$ and $B = \hat{D} + \hat{L} + \hat{U}$, with $D, \hat{D}$ diagonal matrices, L, $\hat{L}$ strictly lower triangular and U, $\hat{U}$ strictly upper triangular.

For $\omega = 1$, the method obviously reduces to Gauss-Seidel method.

3.2. Preconditioned variants. Here we present a preconditioned variant of the iterative method described above. We assume that A or B in (1.1) is an M-matrix and consider the preconditioned version of the iteration (3.6), by using some of the recently developed preconditioners devised to improve the convergence rate of the classic iteration methods for solving linear systems with M-matrices as coefficient matrices, see for example [16, 18, 25, 26, 34] and references therein.

Two of the popular preconditioners for an M-matrix $A = (a_{ij}) \in \mathbf{R}^{n\times n}$ are

$$P_{1,\boldsymbol{\mu}} = \begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ -\mu_2 a_{21} & 1 & \cdots & \cdots & 0 \\ -\mu_3 a_{31} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \cdots & \cdots & \vdots \\ -\mu_n a_{n1} & 0 & \cdots & \cdots & 1 \end{bmatrix}, \tag{3.11}$$

and

$$P_{2,\boldsymbol{\mu}} = \begin{bmatrix} 1 & -\mu_1 a_{12} & \cdots & \cdots & 0 \\ 0 & 1 & -\mu_2 a_{23} & \cdots & 0 \\ 0 & 0 & \cdots & \cdots & 0 \\ \vdots & \vdots & \cdots & \cdots & -\mu_{n-1} a_{n-1,n} \\ 0 & 0 & \cdots & \cdots & 1 \end{bmatrix}, \tag{3.12}$$

where $\mu_i \in [0,1]$.

If A in (1.1) is an M-matrix, then multiplying (1.1) by P on the left-hand side yields

$$\bar{A}XB = \bar{C} \tag{3.13}$$

where $\bar{A} = PA$ and $\bar{C} = PC$. In this case, we can directly apply Algorithm 1 to (3.13) to get a preconditioned variant of Algorithm 1.

If B in (1.1) is an M-matrix, then we rewrite (1.1) as

$$AY\bar{B} = C, \tag{3.14}$$

where $Y = XP^{-1}$, $\bar{B} = PB$. Again, we can use Algorithm 1 to solve the equation (3.14) for Y and then we obtain $X = YP$. This gives another preconditioned variant.

Similarly, if both A and B are M-matrices, then, combining (3.13) with (3.14) yields the third preconditioned variant of Algorithm 1,

$$\bar{A}Y\bar{B} = \bar{C}. \tag{3.15}$$

Note that either in (3.13) or in (3.14) the preconditioner P denotes $P_{1,\boldsymbol{\mu}}$ or $P_{2,\boldsymbol{\mu}}$ and in (3.15) the preconditioner for A do not have to be the same as for B.

4. Numerical examples.

Our algorithms were implemented in MATLAB (R2018b) in a computer LAPTOP-KVSVAUU8 (Intel(R) Core(TM) i5-8250U CPU, 1.60GHz, 64 bits under Windows 10 Home. In this section we first mention some implementation details of Algorithm 1, concerning the Preparatory stage, and then show some examples of its performance.

- For the Gauss-Seidel splittings, matrices F and $-G$, the lower triangular part of A and the strictly upper triangular part of A, respectively, are obtained using the built-in MATLAB functions `tril` and `triu`, with the commands `F=tril(A)` and `G=-triu(A,1)`. Matrices $\hat{F}$ and $\hat{G}$, related to the same decomposition of B, are obtained in a similar way.

  For the SOR splittings with a given parameter ω, we replace matrices A, B and C with ωA, ωB and $\omega^2 C$, respectively. Then matrices F and $-G$ are obtained with the MATLAB commands `F=diag(diag(A))+w*tril(A,-1)` and `G=(1-w)*diag(diag(A))-w*triu(A,1)`. Similarly for matrices $\hat{F}$ and $\hat{G}$.
- The computation of the matrices H and $\hat{H}$ is done using the *backslash or left matrix divide* operator and the *slash or right matrix divide* operator, respectively, with the MATLAB commands $H = F\backslash G$ and $\hat{H} = \hat{G}/\hat{F}$, which consist of calling the function `mrdivide` to compute the solution of the triangular matrix equations $F * H = G$, for H, and $\hat{F}^T\hat{H}^T = \hat{G}^T$, for $\hat{H}^T$.

- The estimates for $\rho = \rho(H)$ and $\hat{\rho} = \rho(\hat{H})$ are computed using the built-in function `eigs` called in the form ρ=`abs(eigs(H,1))` and $\hat{\rho}$=`abs(eigs(Ĥ,1))`, which returns the largest magnitude eigenvalue of the matrices H and $\hat{H}$, respectively (and then we take its absolute value).
  Assuming that $\rho < 1$ and $\hat{\rho} < 1$, in order for condition (2.1) to be satisfied, we set the values for p and q to be the smallest positive integers such that
  $$p > \max\left\{1, \ln(\sqrt{3}-1)/\ln\rho\right\} \quad \text{and} \quad q > \max\{1, \ln(\sqrt{3}-1)/\ln\hat{\rho}\}.$$
  If condition (2.2) for ρ^p and $\hat{\rho}^q$ is not as well satisfied, for those first values of p and q, we increase p and q by one, in turn, until this condition is satisfied (using a `while` loop).
- Once some p is determined, to compute $\mathcal{M}^{-1}$ we rewrite (3.2) in the form
  $$\mathcal{M}^{-1} = (H^{p-1} + \cdots + H + I)F^{-1} = \big((\cdots((H+I)H+I)H+\cdots)H+I\big)F^{-1}$$
  which is computationally more efficient. Here the inverse F^{-1} is also obtained using the MATLAB left matrix divide command $F\backslash I$.
  We proceed similarly for the computation of $\hat{\mathcal{M}}^{-1}$.

EXAMPLE 4.1. *In this example we have*

$$A_{n^2\times n^2} = \begin{bmatrix} T & -I & & & \\ -I & T & -I & & \\ & \ddots & \ddots & \ddots & \\ & & -I & T & -I \\ & & & -I & T \end{bmatrix}, \quad B_{m^2\times m^2} = \begin{bmatrix} \widehat{T} & -I & & & \\ -I_1 & \widehat{T} & -I & & \\ & \ddots & \ddots & \ddots & \\ & & -I_1 & \widehat{T} & -I \\ & & & -I_1 & \widehat{T} \end{bmatrix}$$

where, for a given value c, $I_1 = (1+c)I$,

$$T = \begin{bmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 4 & -1 \\ & & & -1 & 4 \end{bmatrix}, \quad \widehat{T} = \begin{bmatrix} 4+2c & -1 & & & \\ -1-c & 4+2c & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1-c & 4+2c & -1 \\ & & & -1-c & 4+2c \end{bmatrix}$$

and a random matrix $C_{n^2\times m^2}$. *Matrix* A *is associated with the centered difference scheme for a two-dimension Poisson's equation and matrix* B *is associated with a two-dimension convection-diffusion equation where the centered and backward difference schemes for the second and first order derivatives are used, respectively, see [8, 32].*

In Table 4.1, for the cases $c = 0.5$, $c = 0$ and $c = -0.5$ and for various values of $m = n$, we show the performance of Algorithm 1 with the Gauss-Seidel splitting - the values of p and q required for convergence, the spectral radii $\rho(H^p)$ and $\rho(\hat{H}^q)$, the number of iterations, the total CPU time and the CPU time used in the preparatory stage (in seconds). The required accuracy was 10^{-8}.

	$n = m$	p, q	$\rho(H^p)$, $\rho(\hat{H}^q)$	Iter	CPU	CPU prep
	10	10, 8	0.4374, 0.3722	58	0.02	0.01
	20	29, 20	0.5213, 0.2820	70	0.50	0.17
$c = 0.5$	30	54, 29	0.5738, 0.2271	82	4.86	1.58
	40	84, 37	0.6104, 0.1777	92	26.5	10.73
	50	119, 43	0.6365, 0.1468	103	145.05	65.77
	10	11, 11	0.4026, 0.4026	59	0.03	0.02
	20	40, 39	0.4072, 0.4072	70	0.51	0.26
$c = 0$	30	86, 86	0.4128, 0.4128	76	6.19	3.54
	40	150, 150	0.4141, 0.4141	80	46.4	34.95
	50	233, 232	0.4128, 0.4144	83	248.81	185.50
	10	8, 6	0.5160, 0.3003	66	0.03	0.01
	20	23, 12	0.5965, 0.1858	85	0.58	0.14
$c = -0.5$	30	44, 15	0.6359, 0.1465	106	5.64	2.03
	40	70, 18	0.6627, 0.1080	123	35.9	10.50
	50	101, 21	0.6815, 0.0778	138	149.76	44.34

Table 4.1 Performance of Algorithm 1 with Gauss-Seidel splitting and required accuracy 10^{-8}.

The results in Table 4.1 show that a significative part of the computation time is used in the preparatory stage - in some cases more than 50% and in most cases between 25% and 50%. As expected, for all values of the parameter c, the bigger is $m = n$, the closer to 1 are the spectral radii $\rho(H)$ and $\rho(\hat{H})$ and thus the bigger must be p and q so that the conditions for convergence are satisfied.

We also analysed the performance of Algorithm 1 using SOR splittings with $\omega = 1.7$ for the problem in Example 4.1. We report our results in Table 4.2 for $c = 0.5$ and $c = 0$ and the values of $m = n$ considered in Table 4.1. The required accuracy was 10^{-8}.

We can observe that SOR splittings with $\omega = 1.7$ produced a significative reduction in the values of p and q which means that the inicial spectral radii $\rho(H)$ and $\rho(\hat{H})$ were much smaller than the ones given by the Gauss-Seidel splitting. The total number of iterations increased by a factor of about 2.5 in the case $c = 0.5$ but the total CPU time, when m and n are big, can be considered comparable. For the case $c = 0$, the number of iterations increased by a factor of about 3 but the total CPU time is even shorter when $m = n = 50$ and comparable for the other orders.

For this example we can conclude that when we use Gauss-Seidel splitting, p and q increase and thus more time is needed in the preparatory stage, but the number of iterations is reduced, when compared to the relaxation Gauss-Seidel splitting. This is expected since the bigger are p and q the more advanced is the process of convergence when the iterative part starts. Overall, we can say that both the splittings are equally good since the CPU time is comparable.

	$n = m$	p, q	$\rho(H^p), \rho(\hat{H}^q)$	Iter	CPU
	10	3, 3	0.3430, 0.3430	123	0.12
	20	5, 3	0.4383, 0.3430	183	0.90
$c = 0.5$	30	9, 4	0.5625, 0.2401	211	8.77
	40	14, 5	0.6127, 0.1681	232	43.10
	50	20, 6	0.6416, 0.1177	249	162.21
	10	3, 3	0.3430, 0.3430	129	0.06
	20	6, 5	0.3716, 0.4383	227	1.05
$c = 0$	30	14, 14	0.4086, 0.4086	240	9.80
	40	26, 25	0.4027, 0.4170	252	46.93
	50	40, 40	0.4116, 0.4116	264	188.91

Table 4.2 Performance of Algorithm 1 with SOR splitting with $\omega = 1.7$.

The following example suggests that it is not always true that when we consider greater values for p and q than the minimum values needed for the convergence conditions (2.1) and (2.2) to be satisfied, although the number of iterations is reduced, the total computational efficiency, measured by the CPU time, reduces as well.

Example 4.2. *With this example we wanted to observe how the computational efficiency of Algorithm 1, with Gauss-Seidel splitting, varies with increasing values of p and q for the problem described in Example 4.1.*

In Table 4.3 we present the results for $n = m = 40$ *and* $c = 0$. *In this case* $\rho(H) = \rho(\hat{H}) = 0.99414$ *are close to* 1 *and the minimum values of* p *and* q *that ensure convergence are* $p = q = 150$. *We show the CPU time in seconds and the number of iterations for increasing values of* p *and* q. *The required accuracy was* 10^{-8}.

In the same table we also show the results for the case $c = 0.5$ *and* $n = m = 30$, *for which* $\rho(H) = 0.98976$*,* $\rho(\hat{H}) = 0.95017$*, initial* $p = 54$ *and initial* $q = 29$*, and for the case* $c = 0.5$ *and* $n = m = 40$*, with* $\rho(H) = 0.99414$*,* $\rho(\hat{H}) = 0.95437$*, initial* $p = 84$ *and intial* $q = 37$*. The required accuracy was* 10^{-8}*.*

$c = 0$			$c = 0.5$					
$n = 40$ p, q	Iter	CPU	$n = 30$ p, q	Iter	CPU	$n = 40$ p, q	Iter	CPU
150	80	46.5	$54, 29$	82	4.75	$84, 37$	92	26.77
151	79	55.5	$55, 30$	79	4.64	$85, 38$	90	38.44
152	79	53.2	$56, 31$	77	4.65	$86, 39$	88	41.09
153	78	54.4	$57, 32$	74	4.57	$87, 40$	86	50.65
154	77	50.8	$58, 33$	72	4.87	$88, 41$	84	43.04
155	76	52.9	$59, 34$	70	4.47	$89, 42$	83	37.78
156	76	54.4	$60, 35$	68	4.39	$90, 43$	81	37.21
157	75	52.9	$61, 36$	66	5.47	$91, 44$	80	37.27
158	74	56.0	$62, 37$	65	5.11	$92, 45$	78	36.91
159	74	56.7	$63, 38$	63	4.32	$93, 46$	77	35.25
160	73	55.2	$64, 39$	61	5.02	$94, 47$	75	34.18

Table 4.3 Algorithm 1 for increasing p and q, with $c = 0$, $n = m = 40$ and with $c = 0.5$, $n = m = 30$ and $m = n = 40$.

From the results in Table 4.3 it is not obvious to say that there are optimal values for p and q. Another observation is that the differences in CPU time are not very significative and the results for the minimum values for p and q are already very good and seem to be right away a good choice.

Next we compare the performances of Algorithm 1 and its preconditioned variants described in Section 3.2 for the problem in Example 4.1. Preconditioner $P_{1,\mu}$ in (3.11) produced no improvements and for this reason we don't show any results. This was expected since matrices A and B in Example 4.1 have first column with almost all entries equal to zero and then $P_{1,\mu}$ is close to the identity matrix when $\mu_i \in [0, 1]$. We only consider the preconditioner $P_{2,\mu}$ in (3.12).

Example 4.3. *We studied the problem presented in Example 4.1 for* $c = 0.5$ *and* $c = 0$ *for which matrices* A *and* B *are M-matrices. We tested preconditioner* $P_{2,\mu}$ *for different values of the parameter* $\mu_i \in \{0, 0.1, 0.2, \ldots, 0.9, 1\}$ *and considered the three preconditioned variants of Algorithm 1. See Table 4.4 for the results on variant* (3.15)*. The required accuracy was* 10^{-8}*. It was considered the Gauss-Seidel splitting.*

$P_{2,\mu}A$; $P_{2,\mu}B$						
	$n = m$	$\mu_i; A$	$\mu_i; B$	p, q	Iter	CPU
$c = 0.5$	10	0.8	0.4	3, 3	33	0.02
		0.8	0.3	3, 4	35	0.04
		0.5	0.5	3, 2	46	0.03
	20	0.6	0.7	8, 6	50	0.45
		0.7	0.7	6, 5	51	0.45
		0.5	0.5	8, 5	72	0.35
	30	0.8	0.5	7, 5	72	3.61
		0.8	0.3	9, 8	73	3.78
		0.5	0.5	16, 7	83	4.37
	40	0.8	0.4	12, 9	80	16.01
		0.8	0.3	13, 11	82	17.23
		0.5	0.5	26, 9	90	19.50
	50	0.8	0.5	15, 8	91	81.51
		0.8	0.4	17, 10	92	76.92
		0.5	0.5	37, 11	98	101.39
$c = 0$	10	0.6	0.8	3, 3	33	0.02
		0.8	0.6	3, 3	34	0.02
		0.5	0.5	4, 3	59	0.02
	20	0.6	0.8	7, 5	65	0.46
		0.5	0.6	12, 11	66	0.64
		0.5	0.5	13, 13	69	0.74
	30	0.7	0.7	15, 15	72	4.00
		0.7	0.8	10, 13	73	5.65
		0.5	0.5	29, 28	75	8.16
	40	0.8	0.8	16, 16	77	22.03
		0.7	0.8	23, 18	78	23.73
		0.5	0.5	50, 50	79	34.44
	50	0.8	0.8	25, 25	81	65.49
		0.8	0.7	29, 35	82	76.39
		0.5	0.5	78, 77	82	112.20

Table 4.4 Preconditioned variant (3.15) of Algorithm 1 with preconditioner $P_{2,\mu}$.

For all cases considered in Table 4.4 we show the results for the choice $\mu_i = 0.5$, both in $P_{2,\mu}A$ and $P_{2,\mu}B$. We observe that this choice always produces a significative reduction of the values p and q and, although the number of iterations do not change much, the CPU time is similar and sometimes better, when compared to the results without preconditioning in Table 4.1. For instance, for $c = 0$, $m = n = 40$, where p and q where reduced from 150 to 50, the CPU time decreased by a factor of 25%; and for $c = 0$, $m = n = 50$, where the reduction of p and q was from 233 and 232 to 78 and 77, respectively, the CPU time was shortened in aprroximatly 55%; and for $c = 0.5$, $m = n = 50$, p and q diminished from 119 and 43 to 37 and 11, respectively, and the CPU time dropped by a factor of 30%. These results, once more, reveal that a significative part of the computation in Algorithm 1 is related to the preparatory stage.

In Table 4.4 we also show other choices for μ_i - the choices that lead to the smallest number of iterations and to the shortest time, which coincide for all cases (except for $c = 0.5$, $n = m = 20$ and $n = m = 50$, but the differences are very small). Choices of μ_i that produce similar results to the best ones are also presented. For example, for the case $c = 0$ and $m = n = 40$ the values $\mu_{i,A} = \mu_{i,B} = 0.8$ and $\mu_{i,A} = 0.7$, $\mu_{i,B} = 0.8$ both lead to a reduction of more than 50% of the CPU time.

The most significative difference in efficiency occurred for $c = 0$ and $m = n = 50$ with the CPU time being reduced to a fourth of the initial time. In fact, the greater $n = m$ is, the more gain we see in preconditioning.

We decided not to show any results for the other preconditioned variants (3.13) and (3.14). We did several experiments but, although in most cases there was an improvement in terms of efficiency, it was never so good as using the variant (3.15).

5. Application to surface-fitting.

In this section we consider applications of Algorithm 1 to surfaces-fitting. Let us first recall the details for the conventional iterative methods for tensor product surface-fitting. Let $(\phi_0, \ldots, \phi_n)$ and $(\psi_0, \ldots, \psi_m)$ be two normalized totally positive bases. Then, given a sequence of control points $\{P_{ij}\}$, $i = 0, 1, \ldots, n$ and $j = 0, 1, \ldots, m$. A parametric tensor product surface can be defined as

$$S(\alpha, \beta) = \sum_{i=0}^{n} \sum_{j=0}^{m} P_{ij} \phi_i(\alpha) \psi_j(\beta).$$

Let the control points P_{ij} be parameterized with the real increasing sequences $t_0 < t_1 < \ldots < t_n$ and $s_0 < s_1 < \ldots < s_m$, where the parameter pair (t_i, s_j) is assigned to the control point P_{ij}, for $i = 0, 1, \ldots, n$ and $j = 0, 1, \ldots, m$. Then, an initial tensor product surface can be constructed as follows.

$$S^{(0)}(t,s)=\sum_{i=0}^{n}\sum_{j=0}^{m}P_{ij}^{(0)}\phi_i(t)\psi_j(s),$$

where $P_{ij}^{(0)}=P_{ij}$, for $i=0,1,\ldots,n$ and $j=0,1,\ldots,m$. Then the $(k+1)$-th tensor product surface $S^{(k+1)}(t,s)$ for $k\geq 0$, can be obtained as follows

$$S^{(k+1)}(t,s)=\sum_{i=0}^{n}\sum_{j=0}^{m}P_{ij}^{(k+1)}\phi_i(t)\psi_j(s)$$

with

$$\begin{cases}\Delta_{uv}^{(k)}=P_{uv}-S^{(k)}(t_u,s_v),\\ P_{uv}^{(k+1)}=P_{uv}^{(k)}+\Delta_{uv}^{(k)},\end{cases}\tag{5.1}$$

for $u=0,1,\ldots,n$ and $v=0,1,\ldots,m$.

The iteration (5.1) is the conventional iterative method for tensor product surface-fitting and always converges for all normalized totally positive bases. Therefore, the iteration (5.1) is referred to as the progressive iterative approximation, briefly denoted by PIA in CAGD, see [19, 20] and references therein.

Let $P_{uv}^{(k)}$ be arranged in the following matrix form

$$P^{(k)}=(P_{00}^{(k)},P_{01}^{(k)},\ldots,P_{0m}^{(k)},P_{10}^{(k)},P_{11}^{(k)},\ldots,P_{1m}^{(k)},\ldots,P_{n0}^{(k)},P_{n1}^{(k)},\ldots,P_{nm}^{(k)})^T,$$

which is a $[(n+1)(m+1)\times 3]$ matrix. If we denote $P^{(k)}=(P_x^{(k)},P_y^{(k)},P_z^{(k)})$, then the first, the second and last columns consist of x-coordinates, y-coordinates and z-coordinates of all points $P_{uv}^{(k)}$, for $u=0,1,\ldots,n$ and $v=0,1,\ldots,m$.

Furthermore, let A and B be the collocation matrices of the bases $(\phi_0,\ldots,\phi_n)$, and $(\psi_0,\ldots,\psi_m)$, respectively. Then Equation (5.1) can be written as follows.

$$P^{(k+1)}=P^{(k)}+[P-(B\otimes A)P^{(k)}],\tag{5.2}$$

or

$$\begin{cases}P_x^{(k+1)}=P_x^{(k)}+[P_x-(B\otimes A)P_x^{(k)}],\\ P_y^{(k+1)}=P_y^{(k)}+[P_y-(B\otimes A)P_y^{(k)}],\\ P_z^{(k+1)}=P_z^{(k)}+[P_z-(B\otimes A)P_z^{(k)}].\end{cases}\tag{5.3}$$

The iteration sequences of (5.3) is mathematically equivalent to Richardson iterations (see [4, 5]) for solving the following linear systems

$$(B\otimes A)\mathbf{x}=P_x,\quad (B\otimes A)\mathbf{y}=P_y\quad\text{and}\quad (B\otimes A)\mathbf{z}=P_z.\tag{5.4}$$

From the viewpoint of numerical linear algebra, it is preferable to solve

$$AXB^T = C_1, \quad AYB^T = C_2 \quad \text{and} \quad AZB^T = C_3, \tag{5.5}$$

where $\mathbf{vec}(X) = \mathbf{x}$, $\mathbf{vec}(Y) = \mathbf{y}$, $\mathbf{vec}(Z) = \mathbf{z}$, $\mathbf{vec}(C_1) = P_x$, $\mathbf{vec}(C_2) = P_y$ and $\mathbf{vec}(C_3) = P_z$.

Example 5.1. *In this example we compare Algorithm 1, using Gauss-Seidel splitting, with PIA for the tensor product surface-fitting problem described in this section. Here we consider the cubic B-spline basis whose collocation matrices are H-matrices and have the following form*

$$A_{n\times n} = \begin{bmatrix} 1 & 0 & & & \\ \frac{4-\lambda}{24} & \frac{8+\lambda}{12} & \frac{4-\lambda}{24} & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{4-\lambda}{24} & \frac{8+\lambda}{12} & \frac{4-\lambda}{24} \\ & & & 0 & 1 \end{bmatrix}, \qquad B = A^T,$$

for $-2 \le \lambda \le 1$. We solve the three matrix equations in (5.5) *for $\lambda = -1$ and for several values of n. In all cases, we use the minimum values for p and q, $p = 2$ and $q = 1$, and require the accuracy of 10^{-3}. The results are shown in Table 5.1.*

n	PIA	Algorithm 1
10	0.009	0.004
20	0.093	0.004
30	0.321	0.026
40	1.433	0.034
50	4.883	0.045
60	15.73	0.051
70	34.24	0.056
80	102.51	0.073
90	183.61	0.077
100	315.88	0.111

Table 5.1 CPU time in seconds for Algorithm 1 an PIA.

Our results in Example 5.1 clearly show that Algorithm 1 converges significatively faster than PIA. In Figure 5.1 we show the interpolating surface for the case $\lambda = -1$ and $n = 20$.

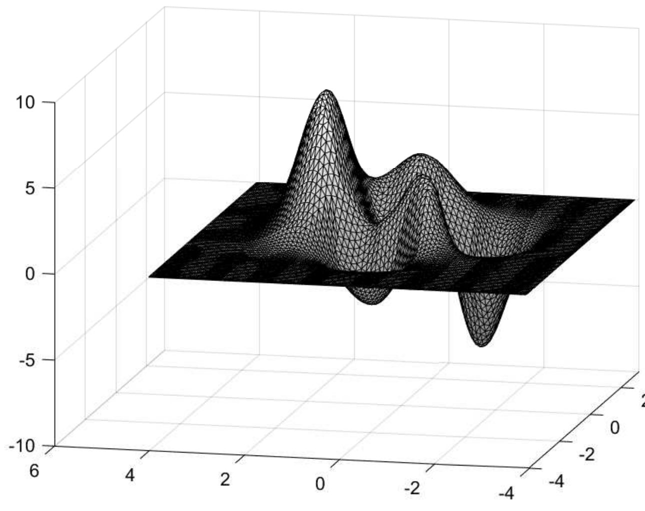

Fig. 5.1 Interpolating surface for Example 5.1 when $\lambda = -1$ and $n = 20$.

6. Conclusions. In this paper we introduced an algorithm to solve the matrix equation $AXB = C$, based on a p-degree induced splitting of A and on a q-degree induced splitting of B. We discussed possible choices for the positive integers p and q so that not only the conditions for convergence are satisfied but also the convergence is fast. We implemented our method in MATLAB, with several concerns with respect to computational costs, and preliminary numerical results show the robustness of our method. In our numerical experiments we also considered preconditioned variants for the case when A and/or B are M-matrices. The value in the interval $[0, 1]$ for the parameter defining the preconditioner considered, even when chosen at random to be 0.5, produced very satisfying improvements. Furthermore, we used Algorithm 1 to solve tensor product surface-fitting problems with the cubic B-spline basis. For our examples the numerical tests show that our method is significatively faster than PIA.

7. Acknowledgement. The authors would like to thank the supports of the National Natural Science Foundation of China under Grant No. 11371075, the Hunan Key Laboratory of mathematical modeling and analysis in engineering, and the Portuguese Funds through FCT–Fundacão para a Ciência e a Tecnologia, within the Project UID/MAT/00013/2013.

REFERENCES

[1] P. Amodio and F. Mazzia, *A parallel Gauss-Seidel method for block tridiagonal linear systems,* SIAM J. Sci. Comput., 16 (1995), pp. 1451-1461.

[2] Z.-Z. Bai and M. Rozloznik, On the numerical behavior of matrix splitting iteration methods for solving linear systems, *SIAM J. Numer. Anal.*, 53 (2015), pp. 1716-1737.

[3] A. Berman and R. Plemmons, *Nonnegative Matrices in Mathematical Sciences*, SIAM, Philadelphia, PA, USA, 1994.

[4] J. Carnicer, J. Delgado and J. Peña , *Richardson method and totally nonnegative linear systems*, Linear Algebra Appl., 11(2010):2010-2017.

[5] J. Carnicer and J. Delgado, *On the progressive iterative approximation property and alternative iterations*, Comput. Aided Geom. Design, 28 (2011): 523-526.

[6] Z.-H. Cao, *On convergence of nested stationary iterative methods*, Linear Algebra Appl., 221 (1995), pp. 159-170.

[7] D. S. Cvetkovic-Ilic, *The reflexive solutions of the matrix equation $AXB = C$*, Comput. Math. Appl., 51 (2006), pp. 897-902.

[8] J. W. Demmel, *Applied Numerical Linear Algebra,* SIAM, Philadelphia, PA, USA, 1997.

[9] D. W. Fausett and C. T. Fulton, *Large least squares problems involving Kronecker products,* SIAM J. Matrix Anal. Appl., 15 (1994), pp. 219-227.

[10] A. Frommer and D. B. Szyld, *H-Splitting and two-stage iterative methods*, Numer. Math., 63 (1992), pp. 345-356.

[11] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore and London, 3rd edition, 1996.

[12] A. Hadjidimos, D. Noutsos and M. Tzoumas, *More on modifications and improvements of classical iterative schemes for M-matrices*, Linear Algebra Appl., 364 (2003), pp. 253-279.

[13] M. Heyouni and A. Essai, *Matrix Krylov subspace methods for linear systems with multiple right-hand sides*, Numerical Algorithms, 40 (2005), pp. 137-156.

[14] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge, UK, 1985.

[15] G.-X. Huang, F. Yin and K. Guo, *An iterative method for the skew-symmetic solution and the optimal approximate solution of the matrix equation $AXB = C$*, J. Comput. Appl. Math., 212 (2008), pp. 231-244.

[16] T. Kohno and H. Niki, *Letter to the Editor: A note on the preconditioned Gauss-Seidel (GS) method for linear systems,* J. Comput. Appl. Math., 233 (2010), 2413-2421.

[17] P. J. Lanzkron, D. J. Rose and D. B. Syzld, *Convergence of nested classical iterative methods for linear systems,* Numer. Math. 58 (1991), pp. 685-702.

[18] W. Li, *The convergence of the modified Gauss-Seidel methods for consistent linear systems,* J. Comput. Appl. Math., 154 (2003), pp. 97-105.

[19] H.-W. Lin, T. Maekawa and C.-Y. Deng, *Survey on geometric iterative methods and their applications*, Comput. Aided Design, 95(2017), pp. 40-51.

[20] M.-Z. Liu, B.-J. Li, Q.-J. Guo, C.-G. Zhu, P. Hu and Y.-H. Shao, *Progressive iterative approximation for regularized least square bivariate B-spline surface fitting,* J. Comput. Appl. Math., 327 (2018), pp. 175-187.

[21] Z.-Y. Liu, Y. Zhou, Y.-L. Zhang, L. Lin and D.-X. Xie, *Some remarks on Jacobi and Gauss-Seidel-type iteration methods for the matrix equation $AXB = C$,* Applied Math. Comput., 354 (2019), pp. 305-307.

[22] Z.-Y. Liu, X.-R. Qin, N.-C. Wu and Y.-L. Zhang, *The shifted classical circulant and skew circulant splitting iteration methods for Toeplitz matrices,* Canad. Math. Bull., 60 (2017), pp. 807-815.

[23] Z.-Y. Liu, N.-C. Wu, X.-R. Qin and Y.-L. Zhang, *Trigonometric transform splitting methods for real symmetric Toeplitz systems, Comput. Math. Appl., 75 (2018), pp. 2782-2794.*

[24] Z.-Y. Liu, H.-B. Wu and L. Lin, *The two-stage iterative methods for symmetric positive definite matrices*, Applied Math. Comput., 114 (2000), pp. 1-12.

[25] J. P. Milaszewicz, *Improving Jacobi and Gauss-Seidel iterations,* Linear Algebra Appl., 93 (1987), pp. 161-170.

[26] H. Niki, K. Harada, M. Morimoto and M. Sakakihara, *The survey of preconditioners used for accelerating the rate of convergence in the Gauss-Seidel method,* J. Comput. Appl. Math., 164-165 (2004), pp. 587-600.

[27] J. M. Ortega, *Numerical Analysis: A Second Course*, Academic Press, New York, 1972.

[28] Z.-Y. Peng, *An iterative method for the least squares symmetric solution of the linear matrix equation $AXB = C$*, Appl. Math. Comput. 170 (2005), 711-723.

[29] Z.-Y. Peng, X.-Y. Hu and L. Zhang, *An iteration method for the symmetric solutions and the optimal approximation solution of matrix equation $AXB = C$*, Appl. Math. Comput., 160 (2005), pp. 763-777.

[30] U. A. Rauhala, *Introduction to array algebra*, Photogramm. Eng. Remote Sens., 46 (1980), pp. 177-192.

[31] P. A. Regalia and S. K. Mitra, *Kronecker products, unitary matrices and signal processing applications*, SIAM Rev., 31 (1989), pp. 586-613.

[32] Y. Saad, *Iterative methods for sparse Linear systems*, 2nd edition, SIAM, Philadelphia, PA, USA, 2000.

[33] V. Simoncini, *Computational Methods for Linear Matrix Equation,* SIAM Review, 58(3) (2016), pp. 377–441.

[34] Z.-L. Tian, et al., *The Jacobi and Gauss-Seidel-type iteration methods for the matrix equation $AXB = C$,* Appl. Math. Comput., 292 (2017), pp. 63-75.

[35] X. Wang, Y. Li and L. Dai, *On Hermitian and skew-Hermitian splitting iteration methods for the linear matrix equation $AXB = C$*. Comput. Math. Appl., 65 (2013), pp. 657-664.

[36] H.-Y. Zha, *Comments on large least squares problems involving Kronecker products*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 1172-1172.

[37] F.-X. Zhang, Y. Li, W.-B. Guo and J.-L. Zhao, *Least squares solutions with special structure to the linear matrix equation $AXB = C$,* Appl. Math. Comput., 217 (2011), pp. 10049-10057.

[38] J. H. Zhang, H. Dai and J. Zhao, *A new family of global methods for linear systems with multiple right-hand sides,* J. Comput. Appl. Math., 236(6) (2011), pp. 1562-1575.