Computational and Applied Mathematics (2024) 43:32
https://doi.org/10.1007/s40314-023-02558-6

On relaxed acceleration of the ADI iteration

Zhongyun Liu[1] · Xiaofei Xu[1] · Yang Zhou[2] · Carla Ferreira[3] · Yulin Zhang[3]

Received: 3 January 2023 / Revised: 23 November 2023 / Accepted: 24 November 2023 /
Published online: 29 December 2023

Abstract
We consider the Alternating Direction Implicit (ADI) method to compute the numerical solution of a continuous Sylvester equation $AX + XB = C$, based on the recently developed inexact ADI iteration, and we propose classical acceleration techniques to enhance its convergence rate. An extrapolated variant (EADI) and a block successive overrelaxation variant (block SOR-ADI) of the ADI iterative method are described. These relaxation approaches are similar to what is used in Gauss-Seidel and Jacobi methods for linear systems, and, to our knowledge, novel, especially the block SOR-ADI scheme. Convergence properties of these two relaxed variants are analyzed when the matrix A is positive definite and the matrix B is positive semi-definite (not necessarily Hermitian matrices), or conversely. Our numerical experiments suggest that these new schemes are computationally attractive. The convergence rate of the ADI method is usually increased, particularly with the block SOR-ADI variant. A comparison with the well-known Hermitian and skew-Hermitian splitting (HSS) method emphasizes the efficiency of the proposed methods.

Keywords Sylvester equation · ADI iteration · Extrapolation · Successive overrelaxation · Positive (semi-)definite matrix

Communicated by Zhong-Zhi Bai.

✉ Yulin Zhang
zhang@math.uminho.pt

Zhongyun Liu
liuzhongyun@263.net

Xiaofei Xu
674204659@qq.com

Yang Zhou
zhouy@hhtc.edu.cn

Carla Ferreira
caferrei@math.uminho.pt

[1] School of Mathematics and Statistics, Changsha University of Science and Technology, Changsha 410076, People's Republic of China

[2] School of Mathematics and Computational Science, Huaihua University, Huaihua 418000, People's Republic of China

[3] Centro de Matemática, Universidade do Minho, 4710-057 Braga, Portugal

Mathematics Subject Classification 15A24 · 15A30 · 65F10 · 65F30

1 Introduction

Given matrices $A \in \mathbb{C}^{m\times m}$, $B \in \mathbb{C}^{n\times n}$ and $C \in \mathbb{C}^{m\times n}$, we will focus on the problem of solving the matrix equation

$$AX + XB = C \tag{1.1}$$

in the variable $X \in \mathbb{C}^{m\times n}$, which is called a continuous Sylvester equation. In the special case $B = A^H$ and $C = C^H$, where K^H denotes the conjugate transpose of K, the continuous Sylvester equation (1.1) reduces to the Lyapunov equation. It is well-known that this equation has a unique solution for X if and only if A and $-B$ do not have common eigenvalues (see, e.g., Horn and Johnson 1991; Lancaster and Tismenetsky 1985).

Throughout this paper, we will assume that A and B in (1.1) are positive definite and positive semi-definite, respectively, or vice versa. Recall the general (weaker) definition of positive (semi-)definiteness, which says that a matrix $K \in \mathbb{C}^{n\times n}$ is said to be positive (semi-)definite if its Hermitian part $\frac{1}{2}(K + K^H)$ is positive (semi-)definite. In general, this condition implies that the real part of $z^H Kz$ is positive (non-negative), for all non-zero vectors $z \in \mathbb{C}^n$.

A Sylvester equation, in general, and a Lyapunov equation, in particular, are formulated in various applications. These arise in systems theory and control, matrix eigen-decompositions, model reduction, numerical solution of matrix differential Riccati equations, image processing, among many others. The importance of these applications motivated the extensive theoretical study of Sylvester equations along with the development of practical algorithms to compute approximate solutions; see, e.g., (Benner et al. 2009; Kürschner et al. 2014; Li et al. 2018; Liu et al. 2020; Smith 1968; Simoncini 2016; Tian et al. 2020; Wachspress 1988; Xiong and Lam 2006) and a large literature therein.

A possible approach to deal with the Sylvester equation (1.1) consists in vectorizing the unknown matrix X and translating the matrix equation into a linear system $\mathscr{A}\mathbf{x} = \mathbf{c}$, where $\mathbf{x}$ and $\mathbf{c}$ are the column-stacking vectors of the matrices X and C, respectively, and $\mathscr{A}$ is the *Kronecker sum* of the matrices A and B^T, that is, $\mathscr{A} = I_m \otimes A + B^T \otimes I_n$, with symbol $\otimes$ denoting the standard Kronecker product. Either direct or iterative methods can be applied to solve this linear system. Another approach is to treat the Sylvester equation (1.1) in its original form using an iterative method directly applied to the matrices A, B and C.

When matrices A and B are large, the dimension of the coefficient matrix $\mathscr{A}$ in the linear system $\mathscr{A}\mathbf{x} = \mathbf{c}$ will be considerably larger and, in general, data storage and computational time become difficult issues. For this reason, the first approach is mainly used in problems of small or medium dimension.

Following the second approach mentioned above, the Alternating Direction Implicit (ADI) method is a popular two-step iterative procedure to obtain a solution to the Sylvester equation (1.1). In this paper we will consider ADI-like methods which, in fact, are analogues to the classical ADI iteration method introduced by Peaceman and Rachford in the context of solving partial differential equations, see (Peaceman and Rachford 1955; Simoncini 2016). A variant of ADI, called factored ADI, was formulated to construct the solution X in factored form, see (Benner et al. 2009; Li and White 2002). The effectiveness of factored ADI depends on whether X is well-approximated by a low rank matrix. This is known to be true under various assumptions about A and B.

In the last two decades, the development of iterative methods based on the concept of matrix splitting have attracted several scholars. In particular, the widely known HSS iteration method proposed in Bai et al. (2003) for solving positive definite, non-Hermitian linear systems, makes use of the Hermitian and skew-Hermitian splitting of the coefficient matrix and it is a resource that motivated several HSS-type iteration methods for solving linear systems, see, e.g., (Bai et al. 2007, 2006, 2007; Benzi 2009; Liu et al. 2018). In Bai (2011), for the first time, a variant of the HSS iteration method, which is an alternating direction implicit method in the spirit analogous to the ADI iteration method, was applied to solve the Sylvester equation (1.1). Numerous other efficient and robust algorithms were proposed in the same spirit, see, e.g., (Li and He 2022; Li et al. 2018; Liu et al. 2022; Wang et al. 2013; Zheng and Ma 2014).

Sylvester equations can also be solved using direct methods like Bartels-Stewart and Hessenberg-Schur methods, see (Bartels and Stewart 1972; Golub et al. 1979). The Bartels-Stewart method is based on the reduction of the matrices A and B to real Schur form (quasi-triangular form) using the QR algorithm for eigenvalues, followed by the use of direct methods to solve several linear systems for the columns of X. In the Hessenberg-Schur method the matrix A is reduced to Hessenberg form and only matrix B is decomposed into the quasi-triangular Shur form. This method is faster than the Bartels-Stewart method. These direct methods are $\mathcal{O}(m^3 + n^3)$ and therefore it is only beneficial to use ADI when matrix–vector multiplication and linear solvers involving A and B can be applied cheaply.

In some particular cases (when A or B is invertible), the Sylvester equation may be rewritten into a Stein matrix equation, after a suitable transformation, e.g., the Caley transformation, $X = \widehat{A}XB + \widehat{C}$, with $\widehat{A} = -A^{-1}$, $\widehat{C} = A^{-1}C$. An extensive amount of iterative methods based on the Smith method were studied and developed for solving the Stein matrix equation in recent years, like (rational) Krylov methods (see Li et al. 2013; Sadkane 2012; Zhou et al. 2009). Despite the lack of a rigorous error analysis, the Smith iteration exhibits very good numerical and computational properties and, when the same shift parameter is used at every iteration, it is equivalent to the ADI iteration.

We propose two relaxation strategies to the classical ADI iterative scheme - an extrapolated variant (EADI) and a block successive overrelaxation variant (block SOR-ADI). These are not strategies for solving inexactly the arising linear systems, but strategies that include in each iteration the usage of a new parameter ω to improve the approximation obtained and thus to accelerate the convergence rate. We based our work in the results presented in Liu et al. (2020).

The new schemes EADI and block SOR-ADI are described in Sections 3 and 4, respectively, after a brief summary of the ADI iteration scheme and some of its convergence properties in Sect. 2. Our numerical experiments, shown in Section 5, suggest that these new schemes can increase the convergence rate of the ADI iterative method and thus be considered attractive solvers.

2 ADI iteration

The method of alternating directions (ADI) is described below. It is a two-step iteration process that alternately updates the column and row spaces of an approximate solution to the Sylvester equation (1.1). An initial guess $X^{(0)}$ is required, as well as shift parameters α and β. An intimate connection between this iterative process and Smith's algorithm (Smith 1968) was mentioned in Wachspress (1988).

ADI iteration. *Given an initial approximation $X^{(0)}$ and two positive constants α and β, repeat the steps*

$$\begin{cases} (\alpha I + A)X^{(k+\frac{1}{2})} = X^{(k)}(\alpha I - B) + C & \left(\text{solve for } X^{(k+\frac{1}{2})}\right) \\ X^{(k+1)}(\beta I + B) = (\beta I - A)X^{(k+\frac{1}{2})} + C & \left(\text{solve for } X^{(k+1)}\right) \end{cases}, \tag{2.1}$$

for $k = 0, 1, 2, \ldots,$ until $\left\{X^{(k)}\right\}$ converges.

The choice of the shift parameters (α, β) strongly determines the convergence rate of the ADI scheme. Some algorithms use different shifts, say $\{(\alpha_k, \beta_k)\}_{k=0}^{N}$ in N iterations, see (Benner et al. 2009; Lu and Wachspress 1991; Wachspress 1988, 2009). Following the strategy adopted in Liu et al. (2020), here we decided to use two fixed shifts (α, β) in every iteration of ADI.

The two steps in (2.1) require the solution of two linear systems involving matrices $\alpha I + A$ and $\beta I + B$. The assumption that matrices A and B are both positive semi-definite, even though one of the matrices is also positive definite, allows us to adequately choose shifts α and β so that the matrices $\alpha I + A$ and $\beta I + B$ are *reasonably* diagonally dominant. This diagonal dominance property guarantees that both incomplete factorization and splitting iteration are applicable, stable and efficient, as mentioned in Bai et al. (2006). Furthermore, the convergence speed of Krylov subspace iteration methods such as GMRES can be considerably accelerated for some adequate preconditioner for these classes of matrices, see (Bai 2016).

When it concerns to analyze the theoretical convergence of the ADI iteration (2.1), it is common to rewrite this iteration in the matrix–vector form,

$$\mathbf{x}^{(k+1)} = \mathscr{H}\mathbf{x}^{(k)} + \mathscr{G}\mathbf{c}, \tag{2.2}$$

where $\mathscr{H}, \mathscr{G} \in \mathbb{C}^{mn\times mn}$ and $\mathbf{x}, \mathbf{c} \in \mathbb{C}^{mn\times 1}$. The matrices $\mathscr{H} = \mathscr{H}(\alpha, \beta)$ and $\mathscr{G} = \mathscr{G}(\alpha, \beta)$ are defined by

$$\begin{cases} \mathscr{H} = \left[(\beta I + B)^{-T}(\alpha I - B)^{T}\right] \otimes \left[(\beta I - A)(\alpha I + A)^{-1}\right] \\ \mathscr{G} = (\alpha + \beta)\left[(\beta I + B)^{-T} \otimes (\alpha I + A)^{-1}\right] \end{cases} \tag{2.3}$$

where $\mathbf{x}$ and $\mathbf{c}$ are the column-stacking vectors of the matrices X and C, respectively, and $\otimes$ is the symbol for the standard Kronecker product. To derive these expressions, use direct substitution in (2.1) and recall that vectorizing UXV column-wise gives $(V^T \otimes U)\mathbf{x}$, for any square matrices U, V and conformable matrix X. The matrix $\mathscr{H}$ is called the *iteration matrix* of the scheme (2.2) and we will denote its spectral radius by $\rho(\mathscr{H})$.

Suppose that A is a positive definite matrix and B is a positive semi-definite matrix, in the weaker sense (recall the definition from the Introduction). We now present a sufficient condition for convergence of the ADI iterative process (2.1) in this case. Assume that equation (1.1) has a unique solution X^* and let $\left\{X^{(k)}\right\}$, $k = 0, 1, 2, \ldots$, be the sequence of approximations to X^* obtained with the iterative scheme (2.1). Let λ_j, $j = 1, \ldots, m$, be the eigenvalues of A and μ_i, $i = 1, \ldots, n$, the eigenvalues of B. Then, by the assumptions of definiteness,

$$\mathrm{Re}(\lambda_j) > 0, \ j = 1, \ldots, m, \qquad \text{and} \qquad \mathrm{Re}(\mu_i) \geq 0, \ i = 1, \ldots, n,$$

where $\mathrm{Re}(x)$, for $x \in \mathbb{C}$, denotes the real part of x. According to (Liu et al. 2020, Theorem 1), for positive constants α and β, if

$$-\min_j \mathrm{Re}(\lambda_j) < \frac{\alpha - \beta}{2} < \min_i \mathrm{Re}(\mu_i),$$

then $\rho(\mathscr{H}) < 1$, or, equivalently, the sequence $\left\{X^{(k)}\right\}$ converges to the exact solution X^*.

3 Extrapolated ADI iteration

Extrapolation is a commonly used acceleration technique in the study of iterative methods. In this section, we introduce an extrapolated ADI iteration (EADI) for solving the Sylvester equation (1.1). To improve the approximation computed in each iteration of the ADI method, this EADI iteration incorporates an extra step (refinement step), using a new parameter ω, at a low computational cost. Following similar ideas to what is used in stationary iterative methods, the parameter ω is fixed in every iteration, the same way that shifts (α, β) are also fixed. The EADI iteration is defined as follows.

EADI iteration. *Given an initial approximation $X^{(0)}$, positive constants α, β (shift parameters) and ω (extrapolation parameter), repeat the steps*

$$\begin{cases} (\alpha I + A)X^{(k+\frac{1}{2})} = X^{(k)}(\alpha I - B) + C & \left(\text{solve for } X^{(k+\frac{1}{2})}\right) \\ \widetilde{X}^{(k)}(\beta I + B) = (\beta I - A)X^{(k+\frac{1}{2})} + C & \left(\text{solve for } \widetilde{X}^{(k)}\right) \\ X^{(k+1)} = (1-\omega)X^{(k)} + \omega \widetilde{X}^{(k)} \end{cases}, \tag{3.1}$$

for $k = 0, 1, 2, \ldots,$ until $\left\{X^{(k)}\right\}$ converges.

According to the formulation (2.2), the first two steps of the EADI iteration (3.1) can be rewritten in the matrix–vector form

$$\widetilde{\mathbf{x}}^{(k)} = \mathscr{H}\mathbf{x}^{(k)} + \mathscr{G}\mathbf{c}, \tag{3.2}$$

where $\mathscr{H}$ and $\mathscr{G}$ are the matrices defined in (2.3). Then, from the third step of (3.1), we obtain

$$\mathbf{x}^{(k+1)} = [(1-\omega)I + \omega\mathscr{H}]\mathbf{x}^{(k)} + \omega\mathscr{G}\mathbf{c}. \tag{3.3}$$

Thus, the iteration matrix of the EADI single-step iterative formula (3.3), denoted by $\mathscr{H}_\omega = \mathscr{H}_\omega(\alpha, \beta, \omega)$, is defined by

$$\mathscr{H}_\omega = (1-\omega)I + \omega\mathscr{H}. \tag{3.4}$$

The following result gives inclusion intervals for the parameters α, β and ω for which the convergence of the EADI iterative procedure (3.1) is ensured.

Theorem 3.1 *Let matrices A and B in equation (1.1) be positive definite and positive semi-definite, respectively, or vice versa (using the weaker definition), and let α, β, ω be three positive constants. Let λ_j, $j = 1, \ldots, m$, be the eigenvalues of A, μ_i, $i = 1, \ldots, n$, the eigenvalues of B and $\rho(\mathscr{H})$ the spectral radius of $\mathscr{H}$ in (2.3). If*

$$-\min_j \operatorname{Re}(\lambda_j) < \frac{\alpha - \beta}{2} < \min_i \operatorname{Re}(\mu_i) \quad \text{and} \quad 0 < \omega < \frac{2}{1 + \rho(\mathscr{H})}, \tag{3.5}$$

then the sequence $\left\{X^{(k)}\right\}$, $k = 0, 1, 2 \ldots,$ generated by the EADI iteration (3.1) converges to the exact solution X^* of (1.1).

Proof We will show that the given conditions imply that $\rho(\mathscr{H}_\omega) < 1$, a necessary and sufficient condition for the convergence of the sequence $\left\{X^{(k)}\right\}$.

As it has been observed at the end of Sect. 2, the first condition on the shift parameters α and β guarantees the convergence of the ADI iterative expression (2.2), which means that $\rho(\mathscr{H}) < 1$.

Now notice that given an eigenvalue λ of $\mathscr{H}$, $(1-\omega)+\omega\lambda$ is an eigenvalue of $\mathscr{H}_\omega$ and thus

$$\rho(\mathscr{H}_\omega)=\max_{\lambda\in\lambda(\mathscr{H})}|(1-\omega)+\omega\lambda|. \tag{3.6}$$

For $\lambda=a+ib, a,b\in\mathbb{R}, i^2=-1$, we have $|a|\le|\lambda|<1$, $|b|\le|\lambda|<1$ and

$$\begin{aligned}|1-\omega+\omega\lambda|^2&=|1-\omega(1-a)+\omega bi|^2=1-2\omega(1-a)+\omega^2(1-a)^2+\omega^2b^2\\&=1-\left[2\omega(1-a)-\omega^2\left(1-2a+|\lambda|^2\right)\right].\end{aligned}$$

So, if $2\omega(1-a)>\omega^2(1-2a+|\lambda|^2)$, we will have $|1-\omega+\omega\lambda|^2<1$ and thus $\rho(\mathscr{H}_\omega)<1$. Observe that, since $\omega>0$, by assumption, and $1-2a+|\lambda|^2>0$, that condition is equivalent to

$$0<\omega<\frac{2(1-a)}{1-2a+|\lambda|^2}. \tag{3.7}$$

To complete the proof it only remains to verify that $0<\omega<\dfrac{2}{1+\rho(\mathscr{H})}$, the condition stated in the theorem, implies (3.7). In fact,

$$\frac{2}{1+\rho(\mathscr{H})}\le\frac{2}{1+|\lambda|}\le\frac{2(1-a)}{1-2a+|\lambda|^2}.$$

given that the inequality $1-2a+|\lambda|^2\le(1+|\lambda|)(1-a)$ is equivalent to $(a+|\lambda|)(1-|\lambda|)\ge0$ which is always true when $|a|\le|\lambda|<1$, since this condition implies $a+|\lambda|\ge0$ and $1-|\lambda|>0$.

We have proved that the restrictions (3.5) on the range of the values α, β and ω imply that $\rho(\mathscr{H}_\omega)<1$ and, therefore, ensure the convergence of the EADI iteration, under the stated assumptions on positiveness of the matrices A and B. □

Observe that when $0<\rho(\mathscr{H})<1$ we will have $1<\dfrac{2}{1+\rho(\mathscr{H})}<2$ and thus, in practice, the choice $0<\omega<1$ will always permit convergence (although it might not be the optimal choice).

We would like to acknowledge that the convergence condition $0<\omega<\dfrac{2}{1+\rho(\mathscr{H})}$ in Theorem 3.1 can also be directly derived from Theorem 1 in Cao (1998). We thank an anonymous referee for this useful observation.

4 Block SOR-ADI iteration

In this section, motivated by the work presented in Bai et al. (2007); Bai and Ng (2012) by Bai, Golub and Ng, we will describe a block successive overrelaxation version of the ADI iterative scheme (block SOR-ADI) and derive convergence conditions for this new version.

First we notice that if the sequence $\left\{X^{(k)}\right\}, k=0,1,2\ldots$, obtained using (2.1), converges to the exact solution X^* of the equation (1.1), then we have

$$\begin{cases}(\alpha I+A)X^*-X^*(\alpha I-B)=AX^*+X^*B=C\\X^*(\beta I+B)-(\beta I-A)X^*=AX^*+X^*B=C\end{cases}.$$

From (2.1) we are able to define the fixed-point matrix equations

$$\begin{cases} (\alpha I + A)X = Y(\alpha I - B) + C \\ Y(\beta I + B) = (\beta I - A)X + C \end{cases} \tag{4.1}$$

and can establish that if X^* is the exact solution of the equation (1.1), then it is the fixed point of both equations above. The reverse is also true. Thus, analogously to (Bai et al. 2007, Theorem 3.1) we have the following result.

Theorem 4.1 *The fixed-point matrix equations in* (4.1) *and the original matrix equation* (1.1) *are*
equivalent equations.

Adopting a matrix–vector formulation, the two equations in (4.1) can be represented by

$$W\mathbf{z} = \mathbf{b} \tag{4.2}$$

where

$$W = \begin{bmatrix} I \otimes (\alpha I + A) & -(\alpha I - B)^T \otimes I \\ -I \otimes (\beta I - A) & (\beta I + B)^T \otimes I \end{bmatrix}, \quad \mathbf{z} = \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \mathbf{c} \\ \mathbf{c} \end{bmatrix}, \tag{4.3}$$

for $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{c}$ representing the column-stacking vectors of the matrices X, Y and C, respectively. We will show that, under the conditions considered in Theorem 3.1, the coefficient matrix $W = W(\alpha, \beta)$ is nonsingular and, thus, the use of the Gauss-Seidel method can be considered.

Theorem 4.2 *Let matrices A and B and parameters α and β satisfy the assumptions and conditions given in Theorem* 3.1. *The coefficient matrix W in the equation* (4.3) *is nonsingular.*

Proof The positive definiteness or semi-definiteness assumptions on A and B and the fact that $\alpha, \beta > 0$ imply that matrices $I \otimes (\alpha I + A)$ and $(\beta I + B)^T \otimes I$ are positive definite and hence nonsingular. Recall the eigenvalue property of the Kronecker product.

Applying the mixed-product and inversion properties of the Kronecker operator, we can decompose W into a matrix product,

$$W = \begin{bmatrix} I & O \\ -I \otimes \left[(\beta I - A)(\alpha I + A)^{-1}\right] & I \end{bmatrix} \begin{bmatrix} I \otimes (\alpha I + A) & -(\alpha I - B)^T \otimes I \\ O & S \end{bmatrix}$$

with matrix S, the Schur complement of W, given by

$$S = \left[(\beta I + B)^T \otimes I\right] - \left\{(\alpha I - B)^T \otimes \left[(\beta I - A)(\alpha I + A)^{-1}\right]\right\}.$$

Thereby, we obtain

$$\begin{aligned} S &= \left[(\beta I + B)^T \otimes I\right] \left\{I - \left[(\beta I + B)^{-T} \otimes I\right] \left[(\alpha I - B)^T \otimes \left[(\beta I - A)(\alpha I + A)^{-1}\right]\right]\right\} \\ &= \left[(\beta I + B)^T \otimes I\right] \left\{I - \left[(\beta I + B)^{-T}(\alpha I - B)^T\right] \otimes \left[(\beta I - A)(\alpha I + A)^{-1}\right]\right\} \\ &= \left[(\beta I + B)^T \otimes I\right] (I - \mathscr{H}), \end{aligned}$$

where $\mathscr{H}$ is the iteration matrix of the ADI scheme, defined in (2.3). Under the given assumptions, $\rho(\mathscr{H}) < 1$. Thus, since both matrices $(\beta I + B)^T \otimes I$ and $(I - \mathscr{H})$ are nonsingular, W is also nonsingular. □

In what follows, we will describe a block SOR-ADI iterative method to solve the block 2×2 linear system (4.2) and establish its convergence properties. First, it is suitable to present the block Jacobi and the block Gauss-Seidel iterations. Let the decomposition

$$\begin{aligned} W &= \begin{bmatrix} I\otimes(\alpha I+A) & O \\ O & (\beta I+B)^T\otimes I \end{bmatrix} - \begin{bmatrix} O & (\alpha I-B)^T\otimes I \\ I\otimes(\beta I-A) & O \end{bmatrix} \\ &= M_J - N_J \end{aligned}$$

be the block Jacobi splitting of W. Thus,

$$M_J^{-1} = \begin{bmatrix} I\otimes(\alpha I+A)^{-1} & O \\ O & (\beta I+B)^{-T}\otimes I \end{bmatrix}$$

and the block Jacobi iterative scheme is defined by

$$\mathbf{z}^{(k+1)} = \mathscr{T}_J\mathbf{z}^{(k)} + M_J^{-1}\mathbf{b}, \tag{4.4}$$

where the iteration matrix $\mathscr{T}_J = \mathscr{T}_J(\alpha,\beta)$ obtained is

$$\mathscr{T}_J = M_J^{-1}N_J = \begin{bmatrix} O & (\alpha I-B)^T\otimes(\alpha I+A)^{-1} \\ (\beta I+B)^{-T}\otimes(\beta I-A) & O \end{bmatrix}. \tag{4.5}$$

The block Gauss-Seidel splitting of W,

$$\begin{aligned} W &= \begin{bmatrix} I\otimes(\alpha I+A) & O \\ -I\otimes(\beta I-A) & (\beta I+B)^T\otimes I \end{bmatrix} - \begin{bmatrix} O & (\alpha I-B)^T\otimes I \\ O & O \end{bmatrix} \\ &= M_{GS} - N_{GS}, \end{aligned}$$

leads to the block Gauss-Seidel iterative scheme

$$\mathbf{z}^{(k+1)} = \mathscr{T}_{GS}\mathbf{z}^{(k)} + M_{GS}^{-1}\mathbf{b}, \tag{4.6}$$

where the iteration matrix $\mathscr{T}_{GS} = \mathscr{T}_{GS}(\alpha,\beta)$ is defined by $\mathscr{T}_{GS} = M_{GS}^{-1}N_{GS}$. Since,

$$M_{GS}^{-1} = \begin{bmatrix} I\otimes(\alpha I+A)^{-1} & O \\ (\beta I+B)^{-T}\otimes\left[(\beta I-A)(\alpha I+A)^{-1}\right] & (\beta I+B)^{-T}\otimes I \end{bmatrix},$$

we obtain

$$\mathscr{T}_{GS} = \begin{bmatrix} O & (\alpha I-B)^T\otimes(\alpha I+A)^{-1} \\ O & \mathscr{H} \end{bmatrix}, \tag{4.7}$$

where $\mathscr{H}$ is the iteration matrix of the ADI method.

In practice, to improve the convergence rate of the Gauss-Seidel iteration, it is common to consider a variant of the method which incorporates a successive overrelaxation technique. Using a new parameter $\omega \neq 0$ (relaxation parameter), in the block Gauss-Seidel iteration we consider the linear system

$$\omega W\mathbf{z} = \omega\mathbf{b},$$

equivalent to (4.3), and the coefficient matrix decomposition

$$\begin{aligned} \omega W &= \begin{bmatrix} I\otimes(\alpha I+A) & O \\ -\omega I\otimes(\beta I-A) & (\beta I+B)^T\otimes I \end{bmatrix} - \begin{bmatrix} (1-\omega)I\otimes(\alpha I+A) & \omega(\alpha I-B)^T\otimes I \\ O & (1-\omega)(\beta I+B)^T\otimes I \end{bmatrix} \\ &= M_{SOR} - N_{SOR}. \end{aligned}$$

The iterative equation obtained, called the block SOR iteration, is

$$\mathbf{z}^{(k+1)} = \mathscr{T}_{SOR}\mathbf{z}^{(k)} + \omega M_{SOR}^{-1}\mathbf{b}, \tag{4.8}$$

where

$$M_{SOR}^{-1} = \begin{bmatrix} I \otimes (\alpha I + A)^{-1} & O \\ \omega(\beta I + B)^{-T} \otimes \left[(\beta I - A)(\alpha I + A)^{-1}\right] & (\beta I + B)^{-T} \otimes I \end{bmatrix}$$

and the iteration matrix $\mathscr{T}_{SOR} = \mathscr{T}_{SOR}(\alpha, \beta, \omega)$ is given by

$$\mathscr{T}_{SOR} = M_{SOR}^{-1}N_{SOR} = \begin{bmatrix} (1-\omega)I \otimes I & \omega(\alpha I - B)^T \otimes (\alpha I + A)^{-1} \\ \omega(1-\omega)(\beta I + B)^{-T} \otimes (\beta I - A) & (1-\omega)I \otimes I + \omega^2\mathscr{H} \end{bmatrix}. \tag{4.9}$$

Observe that with the choice $\omega = 1$ the block SOR method reduces to the block Gauss-Seidel method with $\mathscr{T}_{SOR} = \mathscr{T}_{GS}$.

Recasting $\mathbf{z}^{(k)} = \begin{bmatrix} \mathbf{x}^{(k)} \\ \mathbf{y}^{(k)} \end{bmatrix}$ and $\mathbf{b} = \begin{bmatrix} \mathbf{c} \\ \mathbf{c} \end{bmatrix}$ from (4.2), we can express the block SOR iteration (4.8) as

$$\begin{cases} \mathbf{x}^{(k+1)} = (1-\omega)\mathbf{x}^{(k)} + \omega\left[(\alpha I - B)^T \otimes (\alpha I + A)^{-1}\right]\mathbf{y}^{(k)} + \omega\left[I \otimes (\alpha I + A)^{-1}\right]\mathbf{c} \\ \mathbf{y}^{(k+1)} = \omega(1-\omega)\left[(\beta I + B)^{-T} \otimes (\beta I - A)\right]\mathbf{x}^{(k)} + (1-\omega)\mathbf{y}^{(k)} + \omega^2\mathscr{H}\mathbf{y}^{(k)} + \\ \qquad +\omega^2(\beta I + B)^{-T} \otimes \left[(\beta I - A)(\alpha I + A)^{-1}\right]\mathbf{c} + \omega\left[(\beta I + B)^{-T} \otimes I\right]\mathbf{c} \end{cases} \tag{4.10}$$

and the following step is to go back to the matrix-matrix formulation with the matrices X and Y in (4.1). Prior to this transformation, notice that we can write $\mathscr{H}$ as a mixed-product,

$$\mathscr{H} = \left[(\beta I + B)^{-T} \otimes (\beta I - A)\right]\left[(\alpha I - B)^T \otimes (\alpha I + A)^{-1}\right],$$

and $\mathbf{y}^{(k+1)}$ in (4.10) can be simplified to

$$\begin{aligned} \mathbf{y}^{(k+1)} &= (1-\omega)\mathbf{y}^{(k)} + \omega\left[(\beta I + B)^{-T} \otimes (\beta I - A)\right]\Big\{(1-\omega)\mathbf{x}^{(k)} + \omega\left[(\alpha I - B)^T \otimes (\alpha I + A)^{-1}\right]\mathbf{y}^{(k)} \\ &\quad + \omega\left[I \otimes (\alpha I + A)^{-1}\right]\mathbf{c}\Big\} + \omega\left[(\beta I + B)^{-T} \otimes I\right]\mathbf{c} \\ &= (1-\omega)\mathbf{y}^{(k)} + \omega\left[(\beta I + B)^{-T} \otimes (\beta I - A)\right]\mathbf{x}^{(k+1)} + \omega\left[(\beta I + B)^{-T} \otimes I\right]\mathbf{c}. \end{aligned}$$

The scheme (4.10) can be transformed into the following matrix-matrix version which is called the block SOR-ADI iteration. Recall that for square matrices U, V and vector $\mathbf{x}$, the matrix-stacking of the vector $(V^T \otimes U)\mathbf{x}$ can be given by UXV, where X is the matrix-stacking (column-wise) of the vector $\mathbf{x}$.

Block SOR-ADI iteration. *Given initials approximations $X^{(0)}$ and $Y^{(0)}$ and positive constants α, β (shift parameters) and ω (relaxation parameter), repeat the steps*

$$\begin{cases} X^{(k+1)} = (1-\omega)X^{(k)} + \omega(\alpha I + A)^{-1}\left[Y^{(k)}(\alpha I - B) + C\right] \\ Y^{(k+1)} = (1-\omega)Y^{(k)} + \omega\left[(\beta I - A)X^{(k+1)} + C\right](\beta I + B)^{-1} \end{cases} \tag{4.11}$$

for $k = 0, 1, 2, \cdots$, until $\{X^{(k)}\}$ and $\{Y^{(k)}\}$ converge.

An interesting observation is that this scheme of the block SOR-ADI iteration can be expressed as a certain combination of the ADI iteration and the extrapolated ADI iteration, so it is expectable that the convergence rate will be improved. In this context, observe that (4.11) is equivalent to

$$\begin{cases} (\alpha I + A)X^{(k+\frac{1}{2})} = Y^{(k)}(\alpha I - B) + C & \left(\text{solve for } X^{(k+\frac{1}{2})}\right) \\ X^{(k+1)} = (1-\omega)X^{(k)} + \omega X^{(k+\frac{1}{2})} & \\ Y^{(k+\frac{1}{2})}(\beta I + B) = (\beta I - A)X^{(k+1)} + C & \left(\text{solve for } Y^{(k+\frac{1}{2})}\right) \\ Y^{(k+1)} = (1-\omega)Y^{(k)} + \omega Y^{(k+\frac{1}{2})} & \end{cases}, \tag{4.12}$$

for $k = 0, 1, 2, \cdots$, until $\left\{X^{(k)}\right\}$ and $\left\{Y^{(k)}\right\}$ converge.

For both efficiency and numerical accuracy purposes, the scheme (4.12) is the approach to follow in a computational implementation of the SOR-ADI iteration (4.11), since the explicit computation of a matrix inverse should usually be avoided in favour of the computation of the solution of the equivalent linear system with multiple right-hand sides.

We can establish a relation between the eigenvalues of the block Jacobi iteration matrix $\mathscr{T}_J = \mathscr{T}_J(\alpha, \beta)$, the ADI iteration matrix $\mathscr{H} = \mathscr{H}(\alpha, \beta)$ and the block Gauss-Seidel iteration matrix $\mathscr{T}_{GS} = \mathscr{T}_{GS}(\alpha, \beta)$. In case of convergence, these results allow us to compare the convergence rates. We state these relations in the following theorem.

Theorem 4.3 *Let matrices A and B and parameters α and β satisfy the assumptions and conditions given in Theorem 3.1. The following statements are true:*

(a) $\mathscr{T}_{GS}$ and $\mathscr{H}$ have the same set of non-zero eigenvalues.
(b) If λ is an eigenvalue of $\mathscr{T}_J$, then so is $-\lambda$.
(c) If λ is an eigenvalue of $\mathscr{T}_J$, then λ^2 is an eigenvalue of $\mathscr{H}$.
(d) Conversely, if $\mu \neq 0$ is an eigenvalue of $\mathscr{H}$ and $\lambda^2 = \mu$, then λ is an eigenvalue of $\mathscr{T}_J$.

Thus, as a consequence, since $\rho(\mathscr{H}) < 1$,

$$\rho(\mathscr{H}) = \rho(\mathscr{T}_{GS}) = \left(\rho(\mathscr{T}_J)\right)^2 < 1.$$

This means that block Jacobi iteration (4.4), block Gauss-Seidel iteration (4.6) and ADI iteration (2.1) are all convergent methods. The block Gauss-Seidel iteration is twice as fast as the block Jacobi iteration.

Proof (a) If $\mathscr{T}_{GS}\mathbf{z} = \mu\mathbf{z}$ with $\mathbf{z} = \begin{bmatrix}\mathbf{u}\\ \mathbf{v}\end{bmatrix}$, then from (4.7) we have

$$(\alpha I - B)^T \otimes (\alpha I + A)^{-1}\mathbf{v} = \mu\mathbf{u} \quad \text{and} \quad \mathscr{H}\mathbf{v} = \mu\mathbf{v}.$$

Thus, μ is also an eigenvalue of $\mathscr{H}$ with eigenvector $\mathbf{v}$.

Conversely, if $\mathscr{H}\mathbf{v} = \mu\mathbf{v}$, $\mu \neq 0$, define $\mathbf{u} = (\alpha I - B)^T \otimes (\alpha I + A)^{-1}\mathbf{v}/\mu$ and $\mathbf{z} = \begin{bmatrix}\mathbf{u}\\ \mathbf{v}\end{bmatrix}$. We then obtain $\mathscr{T}_{GS}\mathbf{z} = \mu\mathbf{z}$.

(b) Let λ be an eigenvalue of $\mathscr{T}_J$ and $\mathbf{z} = \begin{bmatrix}\mathbf{u}\\ \mathbf{v}\end{bmatrix}$ the corresponding eigenvector. From the eigenvector equation $\mathscr{T}_J\mathbf{z} = \lambda\mathbf{z}$, we have

$$(\alpha I - B)^T \otimes (\alpha I + A)^{-1}\mathbf{v} = \lambda\mathbf{u} \quad \text{and} \quad (\beta I + B)^{-T} \otimes (\beta I - A)\mathbf{u} = \lambda\mathbf{v}.$$

It is simple to verify that $\mathscr{T}_J\bar{z} = -\lambda\bar{z}$ for $\bar{z} = \begin{bmatrix}\mathbf{u}\\ -\mathbf{v}\end{bmatrix}$ and, thus, $-\lambda$ is an eigenvalue of $\mathscr{T}_J$ with eigenvector $\bar{z}$.

Statements (c) and (d) follow directly from (a), as a corollary of Theorem 4.5 (see below), since
$\mathscr{T}_{SOR} = \mathscr{T}_{GS}$ for $\omega = 1$.

The assumptions on positive definiteness or semi-definiteness of the matrices A and B and the condition on α and β imply that $\rho(\mathscr{H}) < 1$, which says that the ADI iteration is convergent. Thus,

$$\rho(\mathscr{H}) = \rho(\mathscr{T}_{GS}) = \big(\rho(\mathscr{T}_J)\big)^2 < 1$$

and all the three methods are convergent. □

The following concept is an important property in the SOR convergence theory.

Definition 4.4 We say that a matrix T has block Property A if T is a block diagonal matrix or there exists a permutation matrix P such that

$$PTP^T = \begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix}$$

where T_{11} and T_{22} are block diagonal matrices.

The block consistently ordered property generalizes block Property A. See, for instance, (Young 1971).

Matrix W in (4.3) has block Property A. The (1, 1) block is a block diagonal matrix and for the (2, 2) block it is not difficult to verify that there is a permutation matrix P' such that

$$P' \left[(\beta I + B)^T \otimes I\right] P'^T = I \otimes (\beta I + B)^T.$$

So $P = \begin{bmatrix} I & O \\ O & P' \end{bmatrix}$ is a permutation matrix that drives W into the desired block Property A structure. Recall that, in general, if U and V are square matrices, then $U \otimes V$ and $V \otimes U$ are even permutation similar, meaning that there exists a permutation matrix Q such that $V \otimes U = Q(U \otimes V)Q^T$.

It is then possible to apply a classical result in matrix iterative analysis that relates the eigenvalues of the block Jacobi and the block SOR iteration matrices, $\mathscr{T}_J$ and $\mathscr{T}_{SOR}$.

Theorem 4.5 *Let $\omega \neq 0$. If λ is an eigenvalue of $\mathscr{T}_J$ and μ satisfies*

$$(\mu + \omega - 1)^2 = \mu\omega^2\lambda^2, \tag{4.13}$$

then μ is an eigenvalue of $\mathscr{T}_{SOR}$. Conversely, if $\mu \neq 0$ is an eigenvalue of $\mathscr{T}_{SOR}$ and λ satisfies (4.13), then λ is an eigenvalue of $\mathscr{T}_J$.

Proof Since matrix W in (4.3) satisfies Property A, the result follows directly from existing literature. See Theorem 4.5 in Varga (2000). □

The convergence result for the block SOR-ADI iteration (4.11) follows from Theorems 4.3 and 4.5.

Theorem 4.6 *Let matrices A and B and parameters α and β satisfy the assumptions and conditions given in Theorem 3.1. Let $\omega \neq 0$ be the relaxation parameter in the block SOR-ADI iteration (4.11).*

(1) *Suppose that all the eigenvalues of $\mathscr{T}_J$ are real. The block SOR-ADI iteration is convergent if and only if $0 < \omega < 2$.*

(2) *Suppose that some eigenvalues of $\mathscr{T}_J$ are complex. If for some positive number $\tau \in (0, 1)$ and each eigenvalue $\mu = a + ib$ of $\mathscr{T}_J$, the point (a, b) lies in the interior of the ellipse*

$$\mathscr{E}_\tau = \left\{(u, v) : u^2 + \frac{v^2}{\tau^2} = 1\right\}$$

and ω satisfies

$$0 < \omega < \frac{2}{1+\tau}, \tag{4.14}$$

then the block SOR-ADI iteration is convergent. Conversely, if the block SOR-ADI iteration is convergent, then, for some $\tau \in (0, 1)$, all the eigenvalues of $\mathscr{T}_J$ lie inside the ellipse $\mathscr{E}_\tau$. Furthermore, if some eigenvalue lies on $\mathscr{E}_\tau$ and if the block SOR-ADI iteration is convergente, then (4.14) holds.

Proof Observe that the block SOR-ADI iteration (4.11) is convergent if and only if the block SOR iteration (4.8) is convergent. Another important observation here is that the assumptions considered imply that matrices $(\alpha I + A)$ and $(\beta I + B)^T$ are nonsingular and, thus, matrix W in (4.3) not only satisfies Property A, but, in addition, all the diagonal blocks in this special block structure are invertible. Furthermore, block Property A is a sufficient condition for block consistently ordering.

(1) From Theorem 4.3, we have $\rho(\mathscr{T}_J) < 1$ and, by Theorem 2.2 in Young (1971), Chapter 6, this condition is equivalent to $0 < \omega < 2$ and $\rho(\mathscr{T}_{SOR}) < 1$.
(2) This result follows directly from Theorem 4.1 in Young (1971), Chapter 6. Since $\rho(\mathscr{T}_J) < 1$, $\tau \in (0, 1)$.

□

To get the most benefit from overrelaxation, we would like to find the value of ω that minimizes $\rho(\mathscr{T}_{SOR})$. A detailed discussion that the Jacobi spectral radius determines the smallest SOR spectral radius can be found in (Varga 2000, Section 4.3) and in (Young 1971, Section 6.4). The optimal value of ω is

$$\omega_{opt} = \frac{2}{1 + \sqrt{1 - (\rho(\mathscr{T}_J))^2}}$$

which gives the smallest spectral radius $\rho(\mathscr{T}_{SOR}) = \omega_{opt} - 1$.

5 Numerical examples

In this section we compare the computational performance of the three methods, ADI, EADI and block SOR-ADI, exhibiting some numerical examples. We also compare our methods with the Hermitian and skew-Hermitian splitting (HSS) method (Bai 2011), though some numerical examples presented in Liu et al. (2020) already show that the ADI method can outperform the HSS method. All the algorithms were implemented in MATLAB (R2020b) in double precision (unit roundoff $\varepsilon = 2.2\,10^{-16}$) on a LAPTOP-KVSVAUU8 with an Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz and 8 GB RAM, under Windows 10 Home. See Appendix A for details on the implementations (Algorithms 1, 2 and 3 for ADI, EADI and SOR-ADI, respectively, and 4 for HSS). No parallel MATLAB operations were used.

In MATLAB one way to solve the linear system $AX = B$ (multiple right-hand sides) is with $X = \texttt{inv}(A) * B$. A better way, from the point of view of both execution time and numerical accuracy, is to use the matrix backslash operator $X = A\backslash B$. When A is square, this produces the solution using Gaussian elimination, without explicitly forming the inverse A^{-1}. The call $Ad = \texttt{decomposition}(A)$ creates reusable matrix decompositions (LU, LDL, Cholesky, QR, and more) that enables us to solve linear systems more efficiently. The decomposition type is automatically chosen based on the properties of the input matrix A. For example, after computing $Ad = \texttt{decomposition}(A)$ the call $X = Ad\backslash B$ returns the same vector as $X = A\backslash B$, but is typically much faster. This `decomposition` function is well-suited to solving problems that require repeated solutions, since the decomposition of the coefficient matrix does not need to be performed multiple times.

In each iteration, our methods require the solution of multiple linear systems with the same coefficient matrices, $(A + \alpha I)$ and $(B + \beta I)$. Using the above built-in MATLAB functions in our implementations, the efficiency and accuracy will be improved by not forming explicitly the inverses $(A + \alpha I)^{-1}$ and $(B + \beta I)^{-1}$.

In all our numerical experiments we chose as initial approximations the null matrices, that is, $X^{(0)} = O$, for ADI and EADI iterations, and $X^{(0)} = Y^{(0)} = O$, for the block SOR-ADI iteration. The stopping criterion used was

$$\frac{\| R^{(k)} \|_F}{\| C \|_F} \leq tol, \tag{5.1}$$

where $R^{(k)} = C - AX^{(k)} - X^{(k)}B$ is the residual attained at iteration k, tol is the desired accuracy, usually set to 10^{-6}, and $\| \cdot \|_F$ denotes the Frobenius norm.

In our first example, we consider the two-dimensional convection-diffusion equation

$$-(u_{xx} + u_{yy}) + \sigma(x, y)u_x + \tau(x, y)u_y = f(x, y) \tag{5.2}$$

posed on the unit square $(0, 1) \times (0, 1)$ with Dirichlet-type boundary conditions. The coefficients σ and τ represent the velocity components along the x and y directions, respectively, and here we take the case when σ and τ are constant. See (Chen and Sheu 2000, p. 371). Different finite discretization schemes for the Laplacian operator $u_{xx} + u_{yy}$ and for the first derivatives u_x and u_y lead to different linear systems which are naturally equivalent to standard Sylvester equations (with different discretization errors).

In Starke and Niethammer (1991) it is described how to obtain a general Sylvester equation $AX + XB = C$ for any values of the coefficients σ and τ using the second-order finite central differences operator. When σ and τ are constant, A and B are tridiagonal Toeplitz matrices defined by

$$A = \text{tridiag}\left(-1 + \frac{\tau h}{2}, 2, -1 - \frac{\tau h}{2}\right) \text{ and } B = \text{tridiag}\left(-1 + \frac{\sigma h}{2}, 2, -1 - \frac{\sigma h}{2}\right) \tag{5.3}$$

for a uniform $n \times n$ grid with step size $h = 1/(n + 1)$. The matrix A corresponds to the discretization in the y-direction and the matrix B in the x-direction. If $\tau = \sigma$ then $A = B$.

A class of Sylvester equations, which arises from the discretization of various differential equations and boundary value problems using finite differences or Sinc-Galerkin schemes, can be found in Bai (2011); Wang et al. (2013); Zheng and Ma (2014).

Example 5.1 Here we consider the Sylvester equation (1.1) with matrices $A, B \in \mathbb{R}^{n\times n}$ defined by (5.3). The parameters τ and σ depend on the properties of the associated convection-diffusion problem with homogeneous Dirichlet boundary conditions. The function f is defined by $f(x, y) = \boldsymbol{e}^{x+y}$ and different values of τ, σ and $h = \frac{1}{n+1}$ are tested.

Table 1 Performance of ADI, EADI, SOR-ADI and HSS methods for the Example 5.1

	$h = \frac{1}{n+1}$	ADI			EADI			SOR-ADI			HSS		
		$\alpha = \beta$	iter	t_{CPU}	w_{ext}	iter	t_{CPU}	w_{sor}	iter	t_{CPU}	$\alpha = \beta$	iter	t_{CPU}
	0.04	0.5	42	0.003	1.05	37	0.002	0.95	26	0.002	0.75	23	0.05
$\tau = 10;$	0.02	0.5	42	0.01	1.15	32	0.006	0.90	21	0.004	0.5	30	0.02
$\sigma = 100$	0.01	0.25	64	0.03	1.02	30	0.02	0.94	25	0.01	0.31	51	0.09
	0.005	0.15	104	0.17	1.02	50	0.10	0.98	54	0.10	0.15	104	0.87
	0.0025	0.05	273	3.10	1.01	128	1.42	0.98	90	1.10	0.075	210	22.2
	0.04	2.5	35	0.005	1.25	28	0.004	0.92	25	0.003	0.625	26	0.005
$\tau = 1;$	0.02	0.75	34	0.01	1.25	26	0.01	0.90	24	0.008	0.5	30	0.02
$\sigma = 100$	0.01	0.35	49	0.03	1.05	26	0.01	0.97	30	0.01	0.26	51	0.09
	0.005	0.15	104	0.19	1.03	51	0.08	0.98	54	0.12	0.15	104	0.91
	0.0025	0.05	273	3.17	1.01	128	1.46	0.98	90	1.10	0.06	194	14.6
	0.04	1.75	21	0.002	0.95	20	0.001	0.99	20	0.002	0.45	32	0.01
$\tau = 50;$	0.02	0.5	38	0.006	0.95	32	0.008	0.97	28	0.005	0.37	38	0.03
$\sigma = 0.1$	0.01	0.25	70	0.05	0.99	65	0.05	0.99	56	0.04	0.18	77	0.14

The equation in this example was solved using ADI, EADI, block SOR-ADI and HSS methods. The summary of our experiments in terms of the number of iterations (iter) and the CPU time in seconds (t_{CPU}) is presented in Table 1 for different instances of the parameters τ and σ and the order n of the matrices.

The values assigned to the shift parameters α and β, the same for ADI, EADI and SOR-ADI, followed from a few numerical tries starting from the values presented in (Liu et al. 2020, Tables 1, 2, 3) (see $\alpha_{\exp}$ in those tables) which were obtained through a search procedure for similar matrices A and B. The choice of these parameters for the HSS method was made from analogous experiments. The extrapolation parameter in the EADI method and the relaxation parameter in the SOR-ADI method are denoted by ω_{ext} and ω_{sor}, respectively, and the values chosen for these parameters were also settled from some numerical experiments.

In what respects to the number of iterations and the CPU time required by EADI and SOR-ADI methods to converge, given the residual tolerance (5.1), we observe a reduction in both indicators when compared to the ADI method. This reduction is more significative in the block SOR-ADI method, in particular in the case $\tau = 50$ and $\sigma = 0.1$. We can consider that the block SOR-ADI is the method which exhibits the best performance among the three methods (followed by the EADI method) - the number of iterations is sometimes less than a half, or even a third, of the number of iterations needed by the ADI method.

The comparison of the three methods ADI, EADI and block SOR-ADI with the HSS method shows that HSS is the slowest method in terms of the CPU time required. When compared with EADI and SOR-ADI, in addition to the number of iterations required being greater in most cases (about twice as much in half the cases) the computational cost of each HSS iteration is considerably higher and increases with n.

We now present a second example. This example is not connected to a practical application and it was devised only for testing purposes.

Table 2 Performance of ADI, EADI, SOR-ADI and HSS methods for the Example 5.2

$m; n$		ADI			EADI			SOR-ADI			HSS		
		$\alpha = \beta$	iter	t_{CPU}	w_{ext}	iter	t_{CPU}	w_{sor}	iter	t_{CPU}	$\alpha = \beta$	iter	t_{CPU}
	40; 100	0.75	37	0.01	0.85	23	0.008	0.96	23	0.008	1.5	21	0.01
$r = 4$;	500; 10^3	0.75	37	1.62	0.85	23	1.03	0.96	24	1.09	1.5	21	2.28
$s = 3$	10^3; 10^3	0.75	37	2.60	0.85	23	2.65	0.96	23	2.60	1.5	21	5.26
	10^3; 5×10^3	0.75	37	65.03	0.85	23	39.01	0.96	24	41.30	1.5	21	110.5
	40; 100	1.5	40	0.01	0.35	7	0.003	0.92	17	0.007	3.8	12	0.006
$r = 10$;	500; 10^3	1.5	40	1.93	0.35	7	0.38	0.92	17	0.85	3.8	12	1.38
$s = 3$	10^3; 10^3	1.5	40	4.80	0.35	7	1.10	0.92	17	2.12	3.8	12	3.14
	10^3; 5×10^3	1.5	40	70.5	0.35	7	13.08	0.92	17	32.15	3.8	12	87.1

Example 5.2 Consider the problem of finding the solution of the Sylvester equation (1.1) with pentadiagonal Toeplitz matrices $A \in \mathbb{R}^{m \times m}$ and $B \in \mathbb{R}^{n \times n}$ given by

$$A = \text{pentadiag}\,(-0.5, -1, r, -1, -0.5) \;\text{ and }\; B = \text{pentadiag}\,(-1, -0.5, s, -0.5, -1)\,, \tag{5.4}$$

where r and s are two non-zero real parameters, and matrix C is randomly generated from values uniformly distributed in [0, 10].

We considered different values for r and s and, in each case, different orders m and n of the matrices A and B, respectively. Table 2 contains the number of iterations and CPU time required by the four methods ADI, EADI, block SOR-ADI and HSS.

This is an interesting example. In the case $r = 4$ and $s = 3$, regardless of the order of the matrices A and B, the number of iterations is always the same (with residual tolerance $tol = 10^{-6}$) for the four methods. We observe that both EADI and block SOR-ADI methods are more efficient than the ADI method—the number of iterations and the CPU time required are significantly reduced (a reduction of about 40%)—and their performance is approximately the same. The HSS method requires less iterations than the other three methods (only 2 or 3 iterations less than EADI and block SOR-ADI) but it is about twice or three times slower.

In the case $r = 10$ and $s = 3$, again the number of iterations is constant for any order of the matrices A and B, but here we found a significant superiority of the EADI method in relation to block SOR-ADI and ADI methods (although very satisfactory, this contradicts the general pattern which is that block SOR-ADI method is almost always more efficient than EADI). The HSS method requires more iterations than the EADI method and less iterations than the block SOR-ADI method, but in both cases the CPU time is greater.

6 Conclusion

We considered the problem of solving the continuous Sylvester equation $AX + XB = C$ and, combining a classical method with classical acceleration techniques in a way that we have not seen directly presented in the literature before, we developed two iterative methods to solve this equation. These methods are variants of the ADI iterative method - an extrapolated variant (EADI) and a block successive overrelaxation variant (block SOR-ADI). We showed that these variants are closely related to each other, although their formulations did not anticipate this observation. Under certain assumptions of definiteness on matrices A and B, we established sufficient conditions for convergence of these two methods. The preliminary numerical examples provided show, as expected, that these schemes may be advantageous when compared to the basic ADI method. The convergence rate can be improved by the EADI method and, more significantly, by the block SOR-ADI method. Our experiments also show that these new methods are more efficient than the Hermitian and skew-Hermitian splitting (HSS) method. Even in the few cases when the number of iterations required by the HSS method is smaller, the significant higher computational cost of each iteration leads to a greater CPU time.

As future work, instead of using two fixed shifts throughout the whole iteration, we plan to incorporate in the proposed methods EADI and SOR-ADI heuristic shift-parameter strategies to compute different shifts (close to optimal) to be used in every step. We will be able to answer the question if the speed-up by our methods is also obtained when different ADI shifts are used, and compare the performance with existing ADI algorithms that follow the same approach, see (Benner et al. 2009; Li and White 2002; Wachspress 2009). Another interesting and related experiment would be to see if adding the refinement step of EADI to the factored ADI method (Benner et al. 2009) could improve its convergence rate.

Funding The research of the last two authors was partially financed by Portuguese Funds through FCT (Fundação para a Ciência e a Tecnologia) within the Projects UIDB/00013/2020 and UIDP/00013/2020. This work was also supported by NSFC (National Natural Science Foundation of China) under the Grant No.11371075, the Hunan Key Laboratory of Mathematical Modeling and Analysis in Engineering, and the Postgraduate Scientific Research Innovation Project of Hunan Province CX20220953.

Data Availability The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

Appendix A. Implementation details

Algorithm 1 ADI – Alternating Direction Implicit method

```
Input: Matrices $A, B, C$ (orders $m \times m$, $n \times n$ and $m \times n$),
        initial approximations $X_0$, shift parameters $\alpha$ and $\beta$, relative residual tolerance $tol$,
        maximum number of iterations $maxit$
Output: Solution $X$ of the Sylvester equation $AX + XB = C$

  dA = decomposition($A + \alpha * $eye($m$))          ▷ decompositions of $(\alpha I + A)$ and $(\beta I + B)$
  dB = decomposition($B + \beta * $eye($n$))     ▷ reduces the execution time of the backslash operator

  $A_\beta = \beta *$ eye($m$) $- A$
  $B_\alpha = \alpha *$ eye($n$) $- B$

  $X = X_0$
  $R = C - A * X - X * B$                               ▷ initial residual $R = C - AY - YB$
  normR = norm($R$, 'fro')                                ▷ Frobenious norms of $R$ and $C$
  normC = norm($C$, 'fro')

  iter = 0                                              ▷ number of iterations counter
  while ((normR/normC) $> tol$ and iter $< maxit$) do
    $X$ = dA\($X * B_\alpha + C$)
    $X$ = ($A_\beta * X + C$)/dB

    $R = C - A * X - X * B$
    normR = norm($R$, 'fro')
    iter = iter + 1
  end while

  if iter >= $maxit$ then
    disp('Maximum number of iterations exceed.')
  end if
```

Algorithm 2 EADI – Extrapolated Alternating Direction Implicit method

```
Input: Matrices $A$, $B$, $C$ (orders $m \times m$, $n \times n$ and $m \times n$),
       initial approximations $X_0$, shift parameters $\alpha$ and $\beta$, relative residual tolerance $tol$,
       maximum number of iterations $maxit$
Output: Solution $X$ of the Sylvester equation $AX + XB = C$

  dA = decomposition($A + \alpha *$ eye($m$))     ▷ decompositions of $(\alpha I + A)$ and $(\beta I + B)$
  dB = decomposition($B + \beta *$ eye($n$))      ▷ reduces the execution time of the backslash operator

  $\mathrm{A}_\beta = \beta *$ eye($m$) $- A$
  $\mathrm{B}_\alpha = \alpha *$ eye($n$) $- B$
  $\omega_1 = 1 - \omega$

  $X = X_0$
  $R = C - A * X - X * B$                         ▷ initial residual $R = C - AY - YB$
  normR = norm($R$, 'fro')                        ▷ Frobenious norms of $R$ and $C$
  normC = norm($C$, 'fro')

  iter = 0                                        ▷ number of iterations counter
  while ((normR/normC) $> tol$ and iter $< maxit$) do
     $X =$ dA\($X * \mathrm{B}_\alpha + C$)
     $Y =$ ($\mathrm{A}_\beta * X + C$)/dB
     $X = \omega_1 * X + \omega * Y$

     $R = C - A * X - X * B$
     normR = norm($R$, 'fro')
     iter = iter + 1
  end while

  if iter >= $maxit$ then
     disp('Maximum number of iterations exceed.')
  end if
```

Algorithm 3 SOR - ADI – block successive overrelaxation ADI method

Input: Matrices A, B, C (orders $m \times m$, $n \times n$ and $m \times n$), initial approximations X_0 and Y_0, shift parameters α and β, relative residual tolerance tol, relaxation parameter ω, maximum number of iterations $maxit$

Output: Solution Y of the Sylvester equation $AX + XB = C$

```
$\mathtt{dA} = \mathtt{decomposition}(A + \alpha * \mathtt{eye}(m))$     ▷ decompositions of $(\alpha I + A)$ and $(\beta I + B)$
$\mathtt{dB} = \mathtt{decomposition}(B + \beta * \mathtt{eye}(n))$     ▷ reduces the execution time of the backslash operator

$\mathtt{A}_\beta = \beta * \mathtt{eye}(m) - A$
$\mathtt{B}_\alpha = \alpha * \mathtt{eye}(n) - B$
$\omega_1 = 1 - \omega$

$X = X_0; Y = Y_0$
$R = C - A * Y - Y * B$     ▷ initial residual $R = C - AY - YB$
$\mathtt{normR} = \mathtt{norm}(R, \text{'fro'})$     ▷ Frobenious norms of $R$ and $C$
$\mathtt{normC} = \mathtt{norm}(C, \text{'fro'})$

$\mathtt{iter} = 0$     ▷ number of iterations counter
while $\big((\mathtt{normR}/\mathtt{normC}) > tol$ and $\mathtt{iter} < maxit\big)$ do
    $X = \omega_1 * X + \omega * (\mathtt{dA} \backslash (Y * \mathtt{B}_\alpha + C))$
    $Y = \omega_1 * Y + \omega * ((\mathtt{A}_\beta * X + C)/\mathtt{dB})$

    $R = C - A * Y - Y * B$
    $\mathtt{normR} = \mathtt{norm}(R, \text{'fro'})$
    $\mathtt{iter} = \mathtt{iter} + 1$
end while

if $\mathtt{iter} >= maxit$ then
    disp('Maximum number of iterations exceed.')
end if
```

Algorithm 4 HSS – Hermitian and skew-Hermitian splitting iteration

```
Input: Matrices $A$, $B$, $C$ (orders $m \times m$, $n \times n$ and $m \times n$),
       initial approximation $X_0$, shift parameters $\alpha$ and $\beta$, relative residual tolerance $tol$,
       maximum number of iterations $maxit$
Output: Solution $X$ of the Sylester equation $AX + XB = C$

                                                   ▷ Hermitian and skew-Hermitian splittings of $A$ and $B$
  $H_1 = (A + A')/2$
  $S_1 = (A - A')/2$
  $H_2 = (B + B')/2$
  $S_2 = (B - B')/2$
                                                   ▷ schur forms of $H_1$, $H_2$, $S_1$ and $S_2$ (diagonalizable)
  $[Q_1, D_1]$ = schur(full($H_1$))                                     ▷ $H_1 = Q_1 D_1 Q_1^*$
  $[Q_2, D_2]$ = schur(full($H_2$))                                     ▷ $H_2 = Q_2 D_2 Q_2^*$

  $[Q_3, D_3]$ = schur(full($S_1$))
  $[Q_3, D_3]$ = rsf2csf($Q_3$,$D_3$)                                    ▷ $S_1 = Q_3 D_3 Q_3^*$
  $[Q_4, D_4]$ = schur(full($S_2$))
  $[Q_4, D_4]$ = rsf2csf($Q_4$,$D_4$)                                    ▷ $S_2 = Q_4 D_4 Q_4^*$

                          ▷ diagonal elements of $D_1 + \alpha I_n$, $D_2 + \beta I_m$, $D_3 + \alpha I_n$ and $D_4 + \beta I_m$
  $D_1 = \mathtt{diag}(D_1) + \alpha$;  $D_2 = \mathtt{diag}(D_2) + \beta$
  $D_3 = \mathtt{diag}(D_3) + \alpha$;  $D_4 = \mathtt{diag}(D_4) + \beta$

  $X = X_0$
  $R = C - A * X - X * B$

  $normR$ = norm($R$, 'fro');  $normC$ = norm($C$, 'fro')              ▷ Frobenious norms of $R$ and $C$
  iter = 0                                                        ▷ number of iterations counter
  while $\big((normR/normC) > tol$ and iter $< maxit\big)$ do
    % First step
    $R = Q_1{}' * R * Q_2$
    $Z = R./(D1 + D2.')$                                 ▷ solve $(\alpha I + D_1)Z + Z(\beta I + D_2) = R$
    $X = X + Q_1 * Z * Q_2{}'$                                  ▷ update $X$; $X \leftarrow X + Q_1 Z Q_2^*$

    % Second step
    $R = C - A * X - X * B$
    $R = Q_3{}' * R * Q_4$
    $Z = R./(D_3 + D_4.')$                               ▷ solve $(\alpha I + D_3)Z + Z(\beta I + D_4) = R$
    $X = X + Q_3 * Z * Q_4{}'$                                  ▷ update $X$; $X \leftarrow X + Q_3 Z Q_4^*$

    $R = C - A * X - X * B$
    $normR$ = norm($R$, 'fro')
    iter = iter + 1
  end while

  if iter >= maxit then
    disp('Maximum number of iterations exceed.')
  end if
```

References

Bai Z-Z (2011) On Hermitian and skew-Hermitian splitting iterative methods for continuous Sylvester equations. J Comput Math 29:185–198

Bai Z-Z (2016) On SSOR-like preconditioners for non-Hermitian positive definite matrices. Numer Linear Algebra Appl 23:37–60

Bai Z-Z, Ng MK (2012) Erratum. Numer Linear Algebra Appl 19:891

Bai Z-Z, Golub GH, Ng MK (2003) Hermitian and skew-Hermitian splitting methods for non-Hermitian positive definite linear systems. SIAM J Matrix Anal Appl 24:603–626

Bai Z-Z, Yin J-F, Su Y-F (2006) A shift-splitting preconditioner for non-Hermitian positive definite matrices. J Comput Math 24:539–552

Bai Z-Z, Golub GH, Lu L-Z, Yin J-F (2006) Block triangular and skew-Hermitian splitting methods for positive-definite linear systems. SIAM J Sci Comput 26:844–863

Bai Z-Z, Golub GH, Li C-K (2007) Convergence properties of preconditioned Hermitian and skew-Hermitian splitting methods for non-Hermitian positive semidefinite matrices. Math Comput 76:287–298

Bai Z-Z, Golub GH, Ng MK (2007) On successive overrelaxation acceleration of the Hermitian and skew-Hermitian splitting iterations. Numer Linear Algebra Appl 14:319–335

Bartels RH, Stewart GW (1972) Solution of the matrix equation AX + XB = C: Algorithm 432. Commun ACM 15:820–826

Benner P, Li R-C, Truhar N (2009) On the ADI method for Sylvester equations. J Comput Appl Math 233:1035–1045

Benzi P (2009) A generalization of the Hermitian and skew-Hermitian splitting iteration. SIAM J Matrix Anal Appl 31:360–374

Cao Z-H (1998) A convergence theorem on an extrapolated iterative method and its applications. Appl Numer Math 27:203–209

Chen Y-H, Sheu TWH (2000) Two-dimensional scheme for convection-difusion with linear production. Numer Heat Transfer, Part B 37:365–377

Golub GH, Nash SG, Van Loan CF (1979) A Hessenberg-Schur method for the problem AX + XB = C. IEEE Trans Autom Control 24:909–913

Horn RA, Johnson CR (1991) Topics in matrix analysis. Cambridge University Press, Cambridge

Kürschner P, Benner P, Saak J (2014) Self-generating and efficient shift parameters in ADI methods for large Lyapunov and Sylvester equations. Electr Trans Numer Anal 43:142–162

Lancaster P, Tismenetsky M (1985) The theory of matrices, 2nd edn. Academic Press, Orlando

Li X, He N (2022) Shift-splitting iteration method and its variants for solving continuous Sylvester equations, East Asian. J Appl Math 12:367–380

Li J-R, White J (2002) Low rank solution of Lyapunov equations. J Matrix Anal Appl 24(1):260–280

Li T, Weng PC-Y, Chu EK-W, Lin W-W (2013) Large-scale and Lyapunov equations, Smith method, and applications. Numer Algorithm 63(4):727–752

Li X, Huo H-F, Yang A-L (2018) Preconditioned HSS iteration method and its non-alternating variant for continuous Sylvester equations. Comput Math Appl 75:1095–1106

Li P-S, Lam J, Kwok K-W, Lu R-Q (2018) Stability and stabilization of periodic piecewise linear systems: a matrix polynomial approach. Autom J IFAC 94:1–8

Liu Z-Y, Zhou Y, Zhang Y-L (2020) On inexact ADI iteration for continuous Sylvester equations. Numer Linear Algebra Appl 27(5):e2320

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

Liu Z-Y, Zhang F, Ferreira C, Zhang Y-L (2022) On circulant and skew-circulant splitting algorithms for (continuous) Sylvester equations. Comput Math Appl 109:30–43

Lu A, Wachspress EL (1991) Solution of Lyapunov equations by alternating direction implicit iteration. Comput Math Appl 21:43–58

Peaceman DW, Rachford HH Jr (1955) The numerical solution of parabolic and elliptic differential equations. J SIAM 3(1):28–41

Sadkane M (2012) A low-rank Krylov squared Smith method for large-scale discrete-time Lyapunov equations. Linear Algebra Appl 436:2807–2827

Simoncini V (2016) Computational methods for linear matrix equations. SIAM Rev 58:377–441

Smith RA (1968) Matrix equation $XA + BX = C$. SIAM J Appl Math 16:198–201

Starke G, Niethammer W (1991) SOR for $AX - XB = C$. Linear Algebra Appl 154(156):355–375

Tian Z-L, Wang J-X, Dong Y-H, Liu Z-Y (2020) A multi-step Smith-inner-outer iteration algorithm for solving coupled continuous Markovian jump Lyapunov matrix equations. J Frankl Inst 357:3656–3680

Varga R (2000) Matrix iterative analysis, 2nd edn. Springer, New York

Wachspress EL (2009) ADI iteration parameters for solving Lyapunov and Sylvester equations, Technical Report

Wachspress EL (1988) Iterative solution of the Lyapunov matrix equation. Appl Math Lett 107(1):87–90

Wang X, Li W-W, Mao L-Z (2013) On positive-definite and skew-Hermitian splitting iterative methods for continuous Sylvester equations $AX + XB = C$. Comput Math Appl 66:2352–2361

Xiong J, Lam J (2006) Stabilization of discrete-time Markovian jump linear systems via time-delayed controllers. Automatica 42(5):747–753

Young D (1971) Iterative solution of large linear systems. Academic Press, New York

Zheng Q-Q, Ma C-F (2014) On normal and skew-Hermitian splitting iterative methods for large sparse continuous Sylvester equations. J Comput Appl Math 268:145–154

Zhou B, Lam J, Duan G-R (2009) On Smith-type iterative algorithms for the Stein matrix equation. Appl Math Lett 22(7):1038–1044