ON ITERATIVE METHODS BASED ON SHERMAN-MORRISON-WOODBURY SPLITTING

DIMITRIOS MITSOTAKIS

ABSTRACT. We consider a new splitting based on the Sherman-Morrison-Woodbury formula, which is particularly effective with iterative methods for the numerical solution of large linear systems. These systems involve matrices that are perturbations of circulant or block circulant matrices, which commonly arise in the discretization of differential equations using finite element or finite difference methods. We prove the convergence of the new iteration without making any assumptions regarding the symmetry or diagonal-dominance of the matrix.

To illustrate the efficacy of the new iteration we present various applications. These include extensions of the new iteration to block matrices that arise in certain saddle point problems as well as two-dimensional finite difference discretizations. The new method exhibits fast convergence in all of the test cases we used. It has minimal storage requirements, straightforward implementation and compatibility with nearly circulant matrices via the Fast Fourier Transform. For this reasons it can be a valuable tool for the solution of various finite element and finite difference discretizations of differential equations.

1. Introduction

Classical iterative methods for approximating solutions of linear systems are based on the splitting of matrices $\mathbf{A} = \mathbf{M} - \mathbf{N}$ and converge whenever the iteration matrix $\mathbf{G} = \mathbf{M}^{-1}\mathbf{N}$ has spectral radius $\rho(\mathbf{G}) < 1$. The spectral radius $\rho(\mathbf{G})$ also characterizes the speed of convergence of the iterative method, and the smaller it is, the faster the corresponding iterative method converges. Such splittings rely heavily on the invertibility of M. On the other hand, most commonly used methods for the numerical solution of differential equations such as finite difference and finite element methods lead to discretizations with large and sparse linear systems. When structured rectangular grids are used the coefficient matrix A of such linear systems is usually a perturbation of a circulant matrix M, i.e. we can write A = M - N, with N a sparse matrix. Such systems are usually solved with the help of direct methods for band matrices such as the banded LU or Cholesky decomposition and in the case of circulant matrices with the Sherman-Morrison-Woodbury algorithm [12] or with direct inversion of its diagonalization via the Fast Fourier Transform(FFT) [3]. In addition to the previously mentioned classical discretizations, other H^1 -Galerkin/mixed Finite element discretizations lead to saddle point problems with block nearly circulant matrices [7, 15]. General saddle point problems can be very challenging and various methods have been proposed for their numerical approximation of their solutions [1, 2, 10, 11]. Among these methods we just mention the Uzawa-type iterative methods and other Krylov methods such as the conjugate gradient and GMRES method [2, 17].

In this paper we study iterative methods based on a new splitting $\mathbf{A} = \mathbf{M} - \mathbf{N}$. The analysis is not limited to perturbations of circulant or symmetric matrices: We consider general perturbations of a matrix \mathbf{M} , which we call nearly \mathbf{M} matrices. The definition of a nearly \mathbf{M} matrix is based on the Sherman-Morrison-Woodbury formula, [6]:

Definition 1.1. A matrix $\mathbf{A} \in \mathbb{C}^{n,n}$ is near a matrix \mathbf{M} (we say nearly \mathbf{M} matrix) if there is an invertible matrix \mathbf{M} and a matrix $\mathbf{N} = \mathbf{U}\mathbf{V}^T$ for appropriate \mathbf{U}, \mathbf{V} such that $\mathbf{A} = \mathbf{M} - \mathbf{N}$, and with spectral radius $\rho(\mathbf{V}^T\mathbf{M}^{-1}\mathbf{U}) < 1$.

Note that if the spectral radius $\rho(\mathbf{V}^T\mathbf{M}^{-1}\mathbf{U}) < 1$, then $\det(\mathbf{I} - \mathbf{V}^T\mathbf{M}^{-1}\mathbf{U}) \neq 0$ and the Sherman-Morrison-Woodbury theorem ensures that \mathbf{A} is invertible [6]. An example of significant interest is the case where \mathbf{M} is circulant as we mentioned before. In such a case we will call the matrix \mathbf{A} nearly circulant matrix. We know that circulant matrices are invertible and its inversion can be performed as follows: Suppose that we want to solve the system $\mathbf{M}\mathbf{x} = \mathbf{b}$ where \mathbf{M} is circulant, then the diagonalization of \mathbf{M} is $\mathbf{M} = \mathcal{F}\mathbf{D}\mathcal{F}^{-1}$ where \mathcal{F} is the so-called Fourier matrix and \mathbf{D} a diagonal matrix with entries the discrete Fourier transform of the first row (or

Date: October 17, 2023.

 $^{2020\} Mathematics\ Subject\ Classification.\ 65F10, 65F45, 65N22.$

Key words and phrases. Iterative methods, Sherman-Morrison-Woodbury formula, convergence, nearly circulant matrix.

column) of **M**. Therefore, the solution of the linear system can be expressed as $\mathbf{x} = \mathcal{F}(\mathbf{D}^{-1}\mathcal{F}^{-1}\mathbf{b})$, which has a trivial implementation via the FFT, [3].

Some of the main advantages of the proposed iterative method are the following:

- There is no requirement for symmetric, sparse or diagonaly dominant matrix
- It can become faster using sparse matrix-vector multiplication or parallel computing
- In the case of nearly circulant matrices it can be implemented seamlessly via FFT and it can be very fast.

In Section 2 we study the new iteration and its convergence. Applications with various finite element and finite difference discretizations are presented in Section 3. The numerical experiments were performed using our implementations in Python programming language, and thus the times reported in this paper are indicative. Our implementations can be found in [14].

2. The New splitting

Let **A** be near **M**. Using standard notation of [19] for iterative methods, we consider the splitting $\mathbf{A} = \mathbf{M} - \mathbf{N}$ where $\mathbf{N} = \mathbf{U}\mathbf{V}^T$. For this particular splitting, we define the iterative method as usual

(1)
$$\mathbf{x}^{(k+1)} = \mathbf{M}^{-1}(\mathbf{N}\mathbf{x}^{(k)} + \mathbf{b}) \text{ for } k = 0, 1, \dots,$$

for $\mathbf{x}^{(0)}$ a given initial guess of the solution. We will call this new method Sherman-Morrison-Woodbury (SMW) iterative method because of the particular splitting we used.

Before proving the convergence of the new iterative scheme, and for the sake of completeness, we present a generalization of the so-called *matrix determinant lemma* of [4, 5] for more general rank-r modifications:

Lemma 2.1. If $\mathbf{M} \in \mathbb{C}^{n,n}$ is invertible, and $\mathbf{U}, \mathbf{V} \in \mathbb{C}^{n,n}$ matrices, then

(2)
$$\det(\mathbf{M} - \mathbf{U}\mathbf{V}^T) = \det(\mathbf{I} - \mathbf{V}^T\mathbf{M}^{-1}\mathbf{U})\det(\mathbf{M}).$$

Proof. The proof follows from the identity

$$\begin{pmatrix} \mathbf{M} & \mathbf{U} \\ \mathbf{V}^T & \mathbf{I} \end{pmatrix} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{V}^T \mathbf{M}^{-1} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{M} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} - \mathbf{V}^T \mathbf{M}^{-1} \mathbf{U} \end{pmatrix} \begin{pmatrix} \mathbf{I} & \mathbf{M}^{-1} \mathbf{U} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \;.$$

The determinants of the first and last matrix on the right-hand side are equal to 1, while the determinant in the middle of this formula can be expressed as the desired product.

Using the previous matrix determinant lemma we can prove the convergence of the new iterative method:

Theorem 2.1. If **A** is nearly **M**, then the SMW iteration converges.

Proof. In order to prove that the SMW iteration converges we will show that the spectral radius of the iteration matrix $\mathbf{G} = \mathbf{M}^{-1}\mathbf{N}$ is less than 1, i.e. $\rho(\mathbf{M}^{-1}\mathbf{N}) < 1$. If λ is an eigenvalue of the iteration matrix \mathbf{G} , then

$$0 = \det(\mathbf{M}^{-1}\mathbf{N} - \lambda \mathbf{I}) = \det(\mathbf{M}^{-1}\mathbf{U}\mathbf{V}^{T} - \lambda \mathbf{I}) = \det(-\lambda \mathbf{M}^{-1})\det(\mathbf{M} - \frac{1}{\lambda}\mathbf{U}\mathbf{V}^{T})$$
(By Lemma 2.1) = \det(-\lambda\mathbf{M}^{-1})\det(\mathbf{I} - \frac{1}{\lambda}\mathbf{V}^{T}\mathbf{M}^{-1}\mathbf{U})\det(\mathbf{M}) = \det(\mathbf{V}^{T}\mathbf{M}^{-1}\mathbf{U} - \lambda\mathbf{I}).

Thus, the eigenvalues of $\mathbf{M}^{-1}\mathbf{N}$ coincide with the eigenvalues of $\mathbf{V}^T\mathbf{M}^{-1}\mathbf{U}$. By the Definition 1.1 these eigenvalues are $|\lambda| < 1$. Thus, the new iterative method converges.

Remark 2.1. One can define a matrix \mathbf{A} to be nearly \mathbf{M} as the perturbation $\mathbf{A} = \mathbf{M} - \mathbf{u}\mathbf{v}^T$, where $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$ with $|\mathbf{v}^T\mathbf{M}^{-1}\mathbf{u}| < 1$. In such a case the previous theorem holds true again and we can obtain the eigenvalue of the iteration matrix \mathbf{G} to be $\lambda = \mathbf{v}^T\mathbf{M}^{-1}\mathbf{u}$.

In the previous analysis M does not need to be a circulant matrix but any matrix that can be inverted easily. On the other hand the inversion of a circulant matrix can be implemented trivially using the Fast Fourier Transform. Specifically, if M is circulant, then the iterative method (1) can be written as

(3)
$$\mathbf{x}^{(k+1)} = \mathcal{F}\mathbf{D}^{-1}\mathcal{F}^{-1}(\mathbf{N}\mathbf{x}^{(k)} + \mathbf{b}) ,$$

where $\mathbf{M} = \mathcal{F}\mathbf{D}\mathcal{F}^{-1}$. Recall that the diagonal matrix \mathbf{D} is the FFT of the first row of \mathbf{M} . The computational cost of the inversion of \mathbf{M} is $O(n \log(n))$ while the matrix-vector multiplications on the right-hand side of (3) can be performed using sparse matrix multiplication algorithms. The minimal storage requirements of circulant

matrices and the almost linear complexity of the FFT for very large values of n, makes the iteration (3) very useful in demanding situations.

We can improve the speed of the iteration (3) by introducing an extrapolated parameter $\omega \in \mathbb{C} \setminus \{0\}$ and writing the extrapolated SMW (eSMW) iteration in the form

(4)
$$\mathbf{x}^{(k+1)} = [(1-\omega)\mathbf{I} + \omega\mathbf{M}^{-1}\mathbf{N}]\mathbf{x}^{(k)} + \omega\mathbf{M}^{-1}\mathbf{b}, \quad k = 0, 1, 2 \dots$$

It is known that the extrapolated iteration converges for $\max_{\lambda_i \in \sigma(\mathbf{G})} |1 + \omega(\lambda_i - 1)| < 1$, [9, 8]. The estimation of such parameter ω requires the knowledge of the eigenvalues λ_i of $\mathbf{G} = \mathbf{M}^{-1}\mathbf{N}$. To simplify the calculations we will search for a real optimal parameter ω_{opt} that maximizes the speed of convergence, and also we assume that the eigenvalues λ_i are real with λ_{\min} and λ_{\max} the minimum and maximum eigenvalues respectively. Since $\max_i |\lambda_i| < 1$, the convergence of the extrapolated iteration requires that the eigenvalues $1 + \omega(\lambda_i - 1)$ of the iteration matrix $\mathbf{G}_{\omega} = [(1 - \omega)\mathbf{I} + \omega\mathbf{G}]$ are less than 1. This implies that

$$0 < \omega < \frac{2}{1 - \lambda_{\min}} \ .$$

Following [16] we compute the optimal value $\omega_{\rm opt}$ to be

(5)
$$\omega_{\text{opt}} = \frac{2}{2 - (\lambda_{\min} + \lambda_{\max})}.$$

This value is the optimal real value ω that satisfies the minimization problem

$$\min_{\omega \in (0,2/(1-\lambda_{\min}))} \max_{\lambda \in \rho(G)} |1+\omega(\lambda-1)| \ .$$

Recognizing the difficulty of finding the optimal relaxation parameter, in most of the experiments we estimated the parameter using experimentation, while in some case we estimated the eigenvalues numerically, and then computed the optimal values $\omega_{\rm opt}$.

3. Applications

Although we let the definition of a nearly \mathbf{M} matrices to be very generic, we focus here on linear systems that occur in finite element and finite difference discretizations of differential equations with nearly circulant matrices. In all cases we can write $\mathbf{N} = \mathbf{U}^T \mathbf{V}$ and we can verify numerically that the assumptions on matrices \mathbf{U} and \mathbf{V} are satisfied. In some case we present an indicative comparison of the performance of the new iteration with the classical Gauss-Seidel and GMRES methods, [17]. In all the experiments we considered as initial guess of the solution the zero vector.

3.1. **Applications to finite element methods.** First, we demonstrate the applicability of the previous iterations by solving the classical boundary value problem

$$-u''(x) + u(x) = f(x)$$
 for $x \in (0,1)$ with $u(0) = u(1) = 0$,

using finite element methods with spline elements. In particular we consider a uniform partition $0 = x_0 < x_1 < \cdots < x_n = 1$ of [0,1] with stepsize $h = x_{i+1} - x_i$ and the space

$$S_h^1 = \{ \phi \in C([0,1]) \ : \ \phi(0) = \phi(1) = 0 \text{ and } \phi \in P^1([x_i, x_{i+1}]) \text{ for } i = 0, 1, \dots, n-1 \} \ ,$$

where P^r denotes the space of polynomials of degree r. Using the standard basis splines ϕ_i of continuous piecewise linear elements [13], the discrete problem is based on the computation of the H^1 -projection $u_h \in S_h^1$ such that

$$\int_0^1 u_h' \phi' + u_h \phi \ dx = \int_0^1 f \phi \quad \text{for all } \phi \in S_h^1 \ .$$

Expressing $u_h = \sum_i c_i \phi_i(x)$ and taking $\phi = \phi_j$, the discrete problem becomes equivalent to a banded linear system with matrix

$$\mathbf{A} = \mathcal{K} + \mathcal{M} = \frac{1}{h} \begin{pmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{pmatrix} + \frac{h}{6} \begin{pmatrix} 2 & 1 & & & \\ 1 & 4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 4 & 1 \\ & & & 1 & 2 \end{pmatrix} ,$$

where K is the stiffness matrix and M the mass matrix. For the sake of simplicity here we consider h = 1 for any value of n, and we take appropriate right-hand side **b** that leads to the exact solution $\mathbf{x} = (1, 1, \dots, 1)^T$.

Specifically, we considered the circulant matrix defined by the vector

$$\mathbf{M} = \text{circulant}(-5/6, 8/3, -5/6)$$
,

and $\mathbf{N} = \mathbf{M} - \mathbf{A}$, and we solve the corresponding linear system using the iterations (3) and (4). Table 1 presents a comparison between the classical iterative method Gauss-Seidel, the GMRES and the new one. We observe that the new method requires the least iterations to converge while it is very fast. Taking advantage of parallel features of the FFT one can achieve better results.

	method	Gauss-Seidel	GMRES	SMW(3)	$eSMW(\omega = 1.20)$ (4)
$n = 10^3$	time in sec (iterations)	0.462(25)	0.017(21)	0.007(18)	0.006(13)
$n = 10^4$	time in sec (iterations)	3.472(25)	0.026(20)	0.010(18)	0.008(13)
$n = 3 \times 10^4$	time in sec (iterations)	19.43(25)	0.092(19)	0.038(18)	0.029(13)

Table 1. Number of iterations and CPU time required for the linear finite elements matrix

Note that we report only the time used for the execution of the main loop of the method. Our implementations took into account the sparsity of the matrices **A**, **M** and **N** and we used sparse matrix vector multiplications [13]. It is easy to check (numerically) that the condition of the Definition 1.1 is satisfied by the matrices **M** and **N** that is derived using the $n \times n$ matrices **U** and **V** with $\mathbf{U}_{1,1} = \mathbf{U}_{n,n} = 2$, $\mathbf{U}_{1,n} = \mathbf{U}_{n,1} = 1$, $\mathbf{V}_{1,1} = \mathbf{V}_{n,n} = 7/6$ $\mathbf{V}_{1,n} = \mathbf{V}_{n,1} = -1$, and all the other entries are zero.

In a similar manner we study a more demanding situation. We consider the finite element space of smooth cubic splines

$$S_h^3 = \{ \phi \in C^2[0,1] : \phi(0) = \phi(1) = 0 \text{ and } \phi \in P^3([x_i, x_{i+1}] \text{ for } i = 0, 1, \dots, n-1 \},$$

and we compute the corresponding mass matrix for the computation of the L^2 -projection

$$\int_0^1 u_h \phi \ dx = \int_0^1 f \phi \ dx \quad \text{for all } \phi \in S_h^3 \ .$$

The particular mass matrix is hepta-diagonal

and is not a diagonally dominant matrix. Again, for simplicity we take h = 1 for all values of n, and we choose the right hand side appropriately so as the exact solution is again the vector $\mathbf{x} = (1, 1, ..., 1)^T$. For the new method we consider the circulant matrix \mathbf{M} which is defined as

$$\mathbf{M} = \operatorname{circulant}(1/2240, 3/56, 1191/2240, 151/140, 1191/2240, 3/56, 1/2240)$$

and $\mathbf{N} = \mathbf{M} - \mathbf{A}$. The results of the solution of this linear system using the new SMW splittings are summarized in Table 2. In this case, the Jacobi method did not converge at all. Luckily, the Gauss-Seidel method converged to the exact solution. In this experiment the new iteration had pretty much the same performance as our GMRES implementation.

	method	Gauss-Seidel	GMRES	SMW(3)	$eSMW(\omega = 1.86) (4)$
$n = 10^3$	time in sec (iterations)	0.846(72)	0.024(47)	0.091(251)	0.061(137)
$n = 10^4$	time in sec (iterations)	10.40(72)	0.053(44)	0.136(251)	0.080(137)
$n = 3 \times 10^4$	time in sec (iterations)	56.64(72)	0.282(43)	0.425(251)	0.280(137)

Table 2. Number of iterations and CPU time required for the cubic spline finite elements matrix

As we mentioned before, the new methodology can be applied to more general matrices that are not necessarily symmetric or diagonally dominant. To demonstrate this we consider the matrix

$$\mathbf{A} = \begin{pmatrix} 5 & 3 & 2 & 2 \\ 1 & 4 & 3 & 2 \\ 2 & 1 & 4 & 3 \\ 4 & 2 & 1 & 5 \end{pmatrix} ,$$

which is nearly circulant with N the matrix with -1 on its four corners and zeros elsewhere. The SMW iteration (3) converges in 20 iterations. The extrapolated SMW iteration (4) with $\omega_{\rm opt} \approx 0.851$ converges in only 9 iterations, while the Jacobi method does not converge, the Gauss-Seidel method luckily requires 78 iterations and the GMRES required only 3 iterations. The parameter $\omega_{\rm opt}$ was computed using the formula (5).

Remark 3.1. Note that the cases we present in this paper can be handled by other numerical methods and are commonly encountered in finite element and finite difference implementations. However, there are cases where the new method outperforms other methods. For example, consider a dense $n \times n$ circulant matrix \mathbf{M} (for large n), and as \mathbf{N} any sparse matrix that satisfies the requirement stated in Theorem 2.1. We tested such a case with \mathbf{M} constructed by a random vector with entries following the uniform distribution over [1,2). In such a case, the method presented here is the only trivial approach to handle it even for $n=10^6$. This particular example can be found in [14] where the resulting matrix \mathbf{A} is ill-conditioned but still the new method converges almost instantaneously within a few iterations. On the other hand, all the other numerical methods we tested (including the GMRES implementation of the Python module scipy.sparse.linalg) failed to converge even when n=100. The particular experiment can be found in [14].

Remark 3.2. We underline that the performance of all the methods we used can be improved using various techniques (parallelization, preconditioning, etc.) and even with better implementations. For this reason, these experiments serve as indication of the performance of the new iteration. On the other hand, the simplicity of the new method and its overall good performance.

3.2. Extensions to saddle point problems. In this section we consider a linear system that can be derived when we use a mixed formulation of a finite element method for some nonlinear and dispersive wave equations with zero Dirichlet boundary conditions [15]. Consider the linear system

$$\begin{pmatrix} \mathbf{A}_1 & \mathbf{B}_1 \\ \mathbf{B}_2 & \mathbf{A}_2 \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{pmatrix} \ ,$$

where A_1 , A_2 are square $n \times n$ and $m \times m$, and B_1 and B_2 are $n \times m$ and $m \times n$ matrices respectively. The unknown vectors \mathbf{x} and \mathbf{y} are accordingly n and m-dimensional vectors. In what follows we assume for simplicity that A_i and B_i are invertible $n \times n$ matrices.

Taking into account the special structure of the matrices A_1 and A_2 , which are assumed to be near M_1 and M_2 , respectively, we propose the following iterative method:

Write $\mathbf{A}_1 = \mathbf{M}_1 - \mathbf{N}_1$ and $\mathbf{A}_2 = \mathbf{M}_2 - \mathbf{N}_2$, then we define the block Jacobi-SMW iteration

(6)
$$\begin{pmatrix} \mathbf{M}_1 & \\ & \mathbf{M}_2 \end{pmatrix} \begin{pmatrix} \mathbf{x}^{(k+1)} \\ \mathbf{y}^{(k+1)} \end{pmatrix} = \begin{pmatrix} \mathbf{N}_1 & -\mathbf{B}_1 \\ -\mathbf{B}_2 & \mathbf{N}_2 \end{pmatrix} \begin{pmatrix} \mathbf{x}^{(k)} \\ \mathbf{y}^{(k)} \end{pmatrix} + \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{pmatrix} \quad \text{for } k = 0, 1, \dots,$$

and $(\mathbf{x}^{(0)}, \mathbf{y}^{(0)})^T$ is a given initial guess of the solution. If $\mathbf{M}_1 = \mathcal{F}_1 \mathbf{D}_1 \mathcal{F}_1^{-1}$ and $\mathbf{M}_2 = \mathcal{F}_2 \mathbf{D}_2 \mathcal{F}_2^{-1}$ are circulant matrices with \mathcal{F}_i the corresponding Fourier matrices, then the inversion of the block-diagonal matrix

$$\begin{pmatrix} \mathbf{M}_1 & \\ & \mathbf{M}_2 \end{pmatrix} = \begin{pmatrix} \mathcal{F}_1 & \\ & \mathcal{F}_2 \end{pmatrix} \begin{pmatrix} \mathbf{D}_1 & \\ & \mathbf{D}_2 \end{pmatrix} \begin{pmatrix} \mathcal{F}_1^{-1} & \\ & \mathcal{F}_2^{-1} \end{pmatrix} ,$$

is trivial using FFT and in particular the diagonal of \mathbf{D}_i is actually the FFT of the vector comprised the first row of \mathbf{M}_i . Denoting the vectors $\mathbf{u}^{(k)} = (\mathbf{x}^{(k)}, \mathbf{y}^{(k)})^T$ and $\mathbf{b} = (\mathbf{b}_1, \mathbf{b}_2)^T$, and the matrices

$$\mathbf{D} = \begin{pmatrix} \mathbf{D}_1 & \\ & \mathbf{D}_2 \end{pmatrix}, \quad \mathbf{F} = \begin{pmatrix} \mathcal{F}_1 & \\ & \mathcal{F}_2 \end{pmatrix}, \quad \mathbf{F}^{-1} = \begin{pmatrix} \mathcal{F}_1^{-1} & \\ & \mathcal{F}_2^{-1} \end{pmatrix} \quad \text{and} \quad \mathbf{N} = \begin{pmatrix} \mathbf{N}_1 & -\mathbf{B}_1 \\ -\mathbf{B}_2 & \mathbf{N}_2 \end{pmatrix} \;,$$

we express the iterative method in the form

(7)
$$\mathbf{u}^{(k+1)} = \mathbf{F}\mathbf{D}^{-1}\mathbf{F}^{-1}(\mathbf{N}\mathbf{u}^{(k)} + \mathbf{b}).$$

where **D** is actually the FFT of the vector comprised the first row of M_1 followed by the first row of M_2 .

Another extension is the block-Gauss-Seidel-SMW variant of the new iterative method, which can be formulated as

(8)
$$\begin{pmatrix} \mathbf{M}_1 \\ \mathbf{B}_2 \\ \mathbf{M}_2 \end{pmatrix} \begin{pmatrix} \mathbf{x}^{(k+1)} \\ \mathbf{y}^{(k+1)} \end{pmatrix} = \begin{pmatrix} \mathbf{N}_1 \\ \mathbf{N}_2 \end{pmatrix} \begin{pmatrix} \mathbf{x}^{(k)} \\ \mathbf{y}^{(k)} \end{pmatrix} + \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{pmatrix} \text{ for } k = 0, 1, \dots,$$

and with given $(\mathbf{x}^{(0)}, \mathbf{y}^{(0)})$. This is expected to converge in less iterations than the Jacobi method since it utilises the updates $\mathbf{x}^{(k+1)}$ in the computation of the $\mathbf{y}^{(k+1)}$. In the special case where the matrices \mathbf{M}_1 and \mathbf{M}_2 are nearly circulant, then the implementation is straightforward again. If the matrices \mathbf{B}_i are also circulant, then the method can be accelerated again using the FFT of $\mathbf{x}^{(k)}$ in all the intermediate stages. In our implementation we assume that \mathbf{B}_1 and \mathbf{B}_2 are sparse and we use sparse matrix-vector multiplication as it is implemented in Python in CSR storage format, cf. e.g. [13].

As an example we consider the case where $\mathbf{A}_1 = \mathbf{A}_2 = \mathbf{A}$ is the mass matrix \mathcal{M} with entries $\mathbf{A}_{ij} = \int_0^1 \phi_i \phi_j \, dx$ and $\mathbf{B}_{ij} = \int_0^1 \phi_i \phi_j' \, dx$, where ϕ_i are the standard hat basis functions of the space of continuous and piecewise-linear functions S_h^1 . The matrix \mathbf{B} is tri-diagonal $\mathbf{B} = \operatorname{tridiag}(-1/2, 0, 1/2)$ but not circulant. Such matrices occur in mixed formulations of finite element methods like those in [15] and [7]. In this particular case the method converges very fast. The performance of the new method in comparison with the sparse Gauss-Seidel method and GMRES is presented in Table 3.

	method	Gauss-Seidel	GMRES	SMW(6)	SMW (8)
$n = 10^3$	time in sec (iterations)	0.527(25)	0.021(22)	0.040(39)	0.025(37)
$n = 10^4$	time in sec (iterations)	10.11(25)	0.035(21)	0.048(39)	0.042(37)
$n = 3 \times 10^4$	time in sec (iterations)	64.62(25)	0.266(21)	0.152(39)	0.153(37)

Table 3. Number of iterations and CPU time for linear elements in a mixed formulation

The methodology presented in this section can be extended to larger block matrices obtained in saddle point problems and also to two-dimensional problems with tensor products of splines in a very similar manner. A particular, classical example is presented in the next section.

3.3. An application to two-dimensional Poisson equation. We consider the classical five-point finite difference method for the numerical solution of the Poisson equation, [18],

(9)
$$\Delta u = f, \quad (x, y) \in \Omega = [0, 1] \times [0, 1],$$
$$u(x, y) = g, \quad (x, y) \in \partial \Omega.$$

For simplicity we consider the uniform grid with $\Delta x = \Delta y = 1/m$. Applying the five-point stencil finite difference scheme we obtain to a block-tridiagonal system of equations $\mathbf{A}\mathbf{x} = \mathbf{b}$ where \mathbf{A} is the $m^2 \times m^2$ matrix

$$\mathbf{A} = \begin{pmatrix} \mathbf{D} & \mathbf{I} & & & \\ \mathbf{I} & \mathbf{D} & \mathbf{I} & & & \\ & \ddots_{-} & \ddots_{-} & \ddots_{-} & & \\ & & \mathbf{I} & \mathbf{D} & \mathbf{I} \\ & & & \mathbf{I} & \mathbf{D} \end{pmatrix} \;,$$

 $\mathbf{D} = \mathrm{tridiag}(1, -4, 1)$ is $m \times m$ matrix, and \mathbf{I} is the $m \times m$ identity matrix. In order to devise the block-Gauss-Seidel-SMW iteration for this particular case we write $\mathbf{D} = \mathbf{M} - \mathbf{N}$ where \mathbf{M} is the circulant matrix $\mathbf{M} = \mathrm{circulant}(1, -4, 1)$ and \mathbf{N} is such that $\mathbf{N}_{i,j} = 0$ for all i, j except for $\mathbf{N}_{1,m} = \mathbf{N}_{m,1} = 1$. Moreover, we write the k-th iteration and the right hand side as a block-column vectors:

$$\mathbf{x}^{(k)} = (\mathbf{x}_1^{(k)}, \mathbf{x}_2^{(k)} \dots, \mathbf{x}_m^{(k)})^T \quad \text{and} \quad \mathbf{b} = (\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_m)^T \ .$$

Then the algorithm is as follows:

To test this algorithm we took m = 200 (i.e. $\mathbf{A} \in \mathbb{R}^{20000 \times 20000}$). For simplicity, we took again **b** to be the vector that results in the solution **x** with $x_i = 1$ for all *i*. The method with $TOL = 10^{-7}$ converged within 503 iterations and it required 2.98 seconds approximately. Here it is pointless to provide any comparison with the other classical iterative methods but we report that the python implementation of GMRES converged in 5211 iterations and it required more than 30 seconds and the actual difference between the exact solution and the numerical was of the order 10^{-5} even if the residual was of the order 10^{-9} . It is obvious by the requirements

Algorithm 1 Block-Gauss-Seidel-SMW iteration for the Poisson equation

```
Set the initial guess \mathbf{x}_i^{(0)} for i=1,2,\ldots,m

Set the iteration k=0, the tolerance TOL and the maximum number of iterations MAXIT

Initialize Error = TOL + 1

while Error > TOL and k < MAXIT do

Solve \mathbf{M}\mathbf{x}_1^{(k+1)} = \mathbf{N}\mathbf{x}_1^{(k)} - \mathbf{x}_2^{(k)} + \mathbf{b}_1

for i=2,3,\ldots,m-1 do

Solve \mathbf{M}\mathbf{x}_i^{(k+1)} = \mathbf{N}\mathbf{x}_i^{(k)} - \mathbf{x}_{i-1}^{(k+1)} - \mathbf{x}_{i+1}^{(k)} + \mathbf{b}_i

end for

Solve \mathbf{M}\mathbf{x}_m^{(k+1)} = \mathbf{N}\mathbf{x}_m^{(k)} - \mathbf{x}_{m-1}^{(k+1)} + \mathbf{b}_m

Set Error = \|\mathbf{x}^{(k)} - \mathbf{x}^{(k+1)}\| and k = k+1

end while

Return the approximation \mathbf{x}^{(k)}
```

of this experiment that the new method can be used in large-scale problems due to its simplicity, minimum storage requirements as well as its performance.

The implementation of this algorithm along with implementations of the previous variants of the SMW iteration and the setup of all the examples presented here can be found in [14]. All the experiments performed on an Apple computer with processor M1 of 2020 and 16GB RAM.

4. Conclusions

A new splitting based on the Sherman-Morrison-Woodbury formula was presented for the iterative solution of linear systems of certain form. After introducing the notion of nearly M matrices, we proved that for such matrices the new method converges. The new method can be used with finite element and finite difference formulations to solve banded, non-symmetric, block systems (saddle point problems) seamlessly. The applications should not be limited to those we presented here. Among the several advantages of the new method, one is its simple implementation in case of nearly circulant matrices. The speed of the new method can be increased using classical extrapolation techniques and can compete standard iterative methods. The performance of the new iteration is increasing with the dimension of the matrix and becomes faster even than the classical GMRES for large matrices.

ACKNOWLEDGMENTS

The author would like to express his gratitude to Prof. A. Hadjidimos for the fruitful discussions on the subject.

References

- [1] M. Benzi, G. Golub, and J. Liesen. Numerical solution of saddle point problems. Acta Numerica, 14:1—137, 2005.
- [2] S. Brenner. Fast solvers for mixed finite element methods. In P. Wriggres and C. Carstensen, editors, *Mixed finite element technologies*, pages 57–88. Springer, 2009.
- [3] M. Chen. On the solution of circulant linear systems. SIAM Journal on Numerical Analysis, 24(3):668-683, 1987.
- [4] J. Ding and G. Yao. The eigenvalue problem of a specially updated matrix. Applied Mathematics and Computation, 185(1):415

 420, 2007.
- [5] J. Ding and A. Zhou. Eigenvalues of rank-one updated matrices with some applications. Applied Mathematics Letters, 20:1223–1226, 2007.
- [6] G. Golub and C. Van Loan. Matrix computations. Johns Hopkins University Press, 2012.
- [7] L. Guo and H. Chen. H₁-Galerkin mixed finite element method for the regularized long wave equation. Computing, 77:205-221, 2006.
- [8] A. Hadjidimos. A survey of the iterative methods for the solution of linear systems by extrapolation, relaxation and other techniques. *Journal of Computational and Applied Mathematics*, 20:37–51, 1987.
- [9] A. Hadjidimos. Successive overrelaxation (SOR) and related methods. Journal of Computational and Applied Mathematics, 123:177–199, 2000.
- [10] A. Hadjidimos. The saddle point problem and the Manteuffel algorithm. BIT Numerical Mathematics, 56:1281–1302, 2016.
- [11] A. Hadjidimos and M. Tzoumas. On equivalence of three-parameter iterative methods for singular symmetric saddle-point problem. Numerical Algorithms, 86:1391–1419, 2021.
- [12] D. Kahaner, C. Moler, and S. Nash. Numerical methods and software. Prentice-Hall, Inc., 1989.
- [13] D. Mitsotakis. Computational mathematics: An introduction to numerical analysis and scientific computing with Python. Chapman and Hall/CRC Press, New York, 2023.

- [14] D. Mitsotakis. Github repository: An iterative method based on sherman-morrison-woodbury regular splitting for nearly circulant matrices. Technical report, 2023. https://github.com/dmitsot/SMW-iteration.
- [15] D. Mitsotakis, H. Ranocha, D. Ketcheson, and E. Süli. A conservative fully discrete numerical method for the regularized shallow water wave equations. SIAM Journal on Scientific Computing, 43:B508–B537, 2021.
- [16] Y. Saad. Iterative methods for sparse linear systems. SIAM, Philadelphia, 2003.
- [17] Y. Saad and M. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM Journal on Scientific and Statistical Computing, 7:856–869, 1986.
- [18] G. Smith. Numerical solution of partial differential equations: Finite difference methods. Oxford university press, 1985.
- [19] R. Varga. Matrix Iterative Analysis. Springer Berlin Heidelberg, 1999.

D. Mitsotakis: Victoria University of Wellington, School of Mathematics and Statistics, PO Box 600, Wellington 6140, New Zealand

Email address: dimitrios.mitsotakis@vuw.ac.nz