Evaluation of multilevel sequentially semiseparable preconditioners on computational fluid dynamics benchmark problems using Incompressible Flow and Iterative Solver Software

Qiu, Y; van Gijzen, MB; van Wingerden, JW; Verhaegen, MHG; Vuik, C

DOI
10.1002/mma.3416

Publication date
2018

Document Version
Final published version

Published in
Mathematical Methods in the Applied Sciences

Special Issue Paper

Mathematical Methods in the Applied Sciences

Received 31 January 2014

Published online 5 March 2015 in Wiley Online Library

(wileyonlinelibrary.com) DOI: 10.1002/mma.3416
MOS subject classification: 65F08; 76Mxx; 65Y20

Evaluation of multilevel sequentially semiseparable preconditioners on computational fluid dynamics benchmark problems using Incompressible Flow and Iterative Solver Software

Yue Qiu[a], Martin B. van Gijzen[b*†], Jan-Willem van Wingerden[a], Michel Verhaegen[a] and Cornelis Vuik[b]

Communicated by T. Monovasilis

This paper studies a new preconditioning technique for sparse systems arising from discretized partial differential equations in computational fluid dynamics problems. This preconditioning technique exploits the multilevel sequentially semiseparable (MSSS) structure of the system matrix. MSSS matrix computations give a data-sparse way to approximate the *LU* factorization of a sparse matrix from discretized partial differential equations in linear computational complexity with respect to the problem size. In contrast to the standard block diagonal and block upper-triangular preconditioners, we exploit the global MSSS structure of the 2×2 block system from the discretized Stokes equation and linearized Navier-Stokes equation. This avoids approximating the Schur complement explicitly, which is a big advantage over standard block preconditioners. Through numerical experiments on standard computational fluid dynamics benchmark problems in Incompressible Flow and Iterative Solver Software, we show the performance of the MSSS preconditioners. They indicate that the global MSSS preconditioner not only yields mesh size independent convergence but also gives viscosity parameter and Reynolds number independent convergence. Compared with the algebraic multigrid (AMG) method and the geometric multigrid (GMG) method for block preconditioners, the MSSS preconditioning technique is more robust than both the AMG method and GMG method, and considerably faster than the AMG method. Copyright © 2015 John Wiley & Sons, Ltd.

Keywords: partial differential equations; multilevel sequentially semiseparable matrices; preconditioners; computational fluid dynamics; multigrid method

1. Introduction

The most time consuming part of a computational fluid dynamics (CFD) simulation is the solution of one or more linear systems of the following type

$$Ax = b\,, \tag{1}$$

where $A = [A_{ij}]$ is an $n \times n$ matrix and b is a given right-hand-side vector of compatible size [1, 2]. Normally, the system matrix A is large and sparse. Many efforts have been dedicated to finding efficient solution methods for such systems. There are two approaches in general: direct solution methods and iterative solution methods.

[a] *Delft Center for Systems and Control, Delft University of Technology, Mekelweg 2, Delft, 2628 CD, The Netherlands*
[b] *Delft Institute of Applied Mathematics, Delft University of Technology, Mekelweg 4, Delft, 2628 CD, The Netherlands*
* *Correspondence to: Martin B. van Gijzen, Delft Institute of Applied Mathematics, Delft University of Technology, Mekelweg 4, Delft, 2628 CD, The Netherlands.*
† *E-mail: M.B.vanGijzen@tudelft.nl*
This work is an extension of the paper [3] *that has been presented in the 11th International Conference of Numerical Analysis and Applied Mathematics, Rhodes, Greece, 2013.*

Direct solution methods factorize the coefficient matrix A into easily invertible matrices. The time and memory consumption of direct solution methods are predictable, and they are more robust than iterative solution methods. Unfortunately, direct solution methods can be prohibitively expensive both in terms of the memory consumption and computation time for many applications, especially for large CFD problems. For these problems, iterative solution methods usually perform much better than direct solution methods. The conjugate gradient (CG), minimal residual (MINRES), generalized minimal residual (GMRES), and induced dimension reduction (IDR(s)) methods are some of the most popular iterative solution methods [4–6]. Efficiency and robustness of iterative methods can be improved dramatically by combining the preconditioning techniques [7]. In this paper, we study a new class of preconditioners based on the multilevel sequentially semiseparable (MSSS) matrix structure of the system for CFD problems and evaluate the performance of MSSS preconditioners on standard CFD benchmark problems using the Incompressible Flow and Iterative Solver Software (IFISS) [8]. IFISS is a computational laboratory for experimenting with state-of-the-art preconditioned iterative solvers for the discrete linear equations that arise in incompressible flow modeling, which can be run under Matlab or Octave.

Sequentially semiseparable (SSS) matrices appear in many applications, such as circuits and systems [9], interconnected systems [10]. The SSS structure can be exploited so that many computations can be performed in linear computational complexity. SSS matrices make use of the property of the low rank off-diagonal blocks. Systems that arise from the discretization of 1D partial differential equations typically have an SSS structure [11]. MSSS matrices generalize the sequentially semiseparable matrices to the multi-dimensional case. Discretization of higher dimensional (2D or 3D) partial differential equations (PDEs) on structured grid yields matrices with an MSSS structure [12, 13]. MSSS preconditioners have been previously studied in [3, 12, 13]. In [13], Dewilde *et al*. solved a 3D Poisson equation using MSSS matrix computations, while Gondzio *et al*. studied this type of preconditioning technique for PDE-constrained optimization problems in [12]. Gondzio *et al*. solved the Schur complement system with preconditioned conjugate method by MSSS matrix computations. Both [12, 13] only consider the symmetric positive problems and did not deal with block systems arising from discretized coupled PDEs, which are quite common in CFD problems. Meanwhile, these papers do not give comparison of the performance for the MSSS preconditioners with the other methods. In [3], MSSS preconditioners were applied to solve non-symmetric convection–diffusion equations. MSSS matrix computations are also widely studied in the field of distributed control and identification of spatially interconnected systems. The results on systems and control are summarized in [11, 14].

Several other related structured matrices have been proposed in literature. This includes hierarchical semiseparable (HSS) matrices [15, 16], hierarchical ($\mathcal{H}$-) matrices [17–19], and $\mathcal{H}^2$-matrices [20, 21]. HSS matrix computations are usually applied in the multifrontal solver [22]. Some recent efforts devoted to preconditioning of symmetric positive definite systems by HSS matrix computations can be found in [23, 24]. As introduced in [11], MSSS matrices originate from interconnected systems, while $\mathcal{H}$-matrices and $\mathcal{H}^2$-matrices, which are more general structured matrices, originate from the approximation of the kernel of integral functions. In [25, 26], Bebendorf extended $\mathcal{H}$-matrix computations to solving elliptic PDEs problems. Preconditioning techniques based on $\mathcal{H}$-matrix computations for CFD problems were studied in [18, 19]. In [18], an $\mathcal{H}-LU$ preconditioner was proposed to solve the convection–diffusion equation, while in [19], the augmented Lagrangian preconditioner based on $\mathcal{H}$-matrix computations was introduced to solve the discrete Oseen problems. For unstructured grids, HSS/$\mathcal{H}$-matrices are well suited. It was shown in [25, 27] that HSS matrices and $\mathcal{H}$-matrices can be used to represent the discretized PDEs on unstructured grids. For MSSS matrices, this is less natural. Although MSSS matrices do not give a direct representation of discretized PDEs on unstructured grid, it was shown in [16] that the HSS matrices and 1-level MSSS matrices can be transferred from one to the other, which makes it possible for MSSS matrices to infer unstructured grids. The advantage of MSSS matrix computations is their simplicity and low cost, which is $\mathcal{O}(r^3N)$ with bounded small r, compared with $\mathcal{O}(N\log_2^\alpha N)$ with moderate α for $\mathcal{H}$-matrices. Using MSSS matrix computations to compute the preconditioner is motivated by the relation between interconnected systems and MSSS matrices, which is introduced in [11]. Once the grid for the discretization of PDEs is known, the MSSS matrix structure of the discretized system will automatically be known. This will naturally represent the sparse matrix as an MSSS matrix by considering the grid points as interconnected systems. The permutation of MSSS blocks to a single MSSS matrix is also direct and clear by checking the correspondence of interconnected systems with MSSS matrices, which is a big advantage of MSSS matrices over $\mathcal{H}$-matrices and HSS matrices. The permutation operation plays a key role for the preconditioning of the systems from discrete Stokes equation and linearized Navier-Stokes equation, which will be introduced in the later section.

In this paper, we consider MSSS preconditioning techniques for CFD problems on structured grids. For the discretized convection–diffusion equation, we exploit the MSSS structure of the global system matrix, whereas for the discretized Stokes and linearized Navier-Stokes problem, we exploit the MSSS structure of the blocks of the system and permute the system matrix with MSSS blocks into a single MSSS matrix. With this permutation, the discrete Stokes equation and discrete linearized Navier-Stokes equation can be put in the MSSS matrix framework and the computation of the Schur complement can be avoided. While computing an approximation of the Schur complement is the key for standard preconditioning techniques and normally is extremely expensive and difficult. This enables us to solve the CFD problems with Krylov subspace methods using MSSS preconditioners in linear computational complexity. We evaluate the performance of the MSSS preconditioning technique on CFD benchmark problems in IFISS and compare with the block preconditioning technique by the algebraic multigrid (AMG) method and the geometric multigrid (GMG) method. Numerical experiments illustrate that the MSSS preconditioning technique yields mesh size independent convergence and eliminates the convergence dependency on the viscosity parameter and the Reynolds number. This is a big advantage over the AMG and GMG methods. In addition to robustness, it is shown that the MSSS preconditioning technique is considerably faster than the AMG method.

The outline of this paper is as follows. In Section 2, we briefly introduce the MSSS matrices and the mostly used computations. Correspondence between MSSS matrices and discretized PDEs will also be stated in this section. The MSSS preconditioning technique for discretized scalar PDEs and coupled PDEs will be addressed in Section 3. Numerical experiments that evaluate the performance of the MSSS preconditioning technique for CFD benchmark problems are studied in Section 4. Performance comparison with the AMG and GMG method for such preconditioning technique is also contained in this section. Conclusions and remarks will be drawn in the

last section. A companion technical report [28] is also available that contains more numerical experiments on CFD benchmark problems to show the performance of the MSSS preconditioners.

2. Multilevel sequentially semiseparable matrices

Semiseparable matrices are matrices whose sub-matrices taken from the lower-triangular or upper-triangular part are of rank 1. They have been introduced in [29] and appear in several types of applications, such as integral equations [30], Gauss-Markov processes [31], boundary value problems [32], and rational interpolation [33]. Quasiseparable matrices generalize the semiseparable matrices, are matrices all sub-matrices extracted from the strictly lower-triangular or the strictly upper-triangular part, are of rank 1 [29]. If the sub-matrices taken from the strictly lower-triangular part and the strictly upper-triangular part are of low rank, not limited to 1, then this type of matrices is called sequentially semiseparable (SSS) [34]. The property of low-rank off-diagonal blocks for SSS matrices is also investigated by Eidelman *et al*. in [35], where they still call this type of matrices quasiseparable matrices. The SSS structure is closed under basic matrix–matrix operations such as addition, multiplication and inversion. Decompositions/factorizations such as the *QR* [36, 37], *LU/LDU* [12, 35] can also be computed in a structure preserving way such that the factors have SSS structure. Moreover, all the operations mentioned earlier on SSS matrices can be performed in linear computational complexity. Besides, the memory consumption of SSS matrices also scales linearly with the problem size [12, 38].

To keep this paper self-contained, we review some definitions and concepts for SSS matrices [3, 39]. The matrices in this paper will always be real, and their dimensions are compatible for the matrix–matrix operations and the matrix-vector operations when their sizes are not mentioned. The generators representation for SSS matrices are defined by Definition 2.1.

Definition 2.1 ([34])
Let A be an $N \times N$ matrix with the SSS structure. Let $m_1,\ m_2,\ \cdots\ m_n$ be positive integers with $N = m_1 + m_2 + \cdots + m_n$ such that A can be written in the following block-partitioned form:

$$A_{ij} = \begin{cases} U_i W_{i+1} \cdots W_{j-1} V_j^T, & i < j; \\ D_i, & i = j; \\ P_i R_{i-1} \cdots R_{j+1} Q_j^T, & i > j \end{cases} \tag{2}$$

where the superscript 'T' denotes the transpose of the matrix. The previous representation of A is called the generators representation. The sequences $\{U_i\}_{i=1}^{n-1}$, $\{W_i\}_{i=2}^{n-1}$, $\{V_i\}_{i=2}^{n}$, $\{D_i\}_{i=1}^{n}$, $\{P_i\}_{i=2}^{n}$, $\{R_i\}_{i=2}^{n-1}$, $\{Q_i\}_{i=1}^{n-1}$ are matrices whose sizes are listed in Table I, and they are called generators of the SSS matrix A.

With the generators parametrization in Definition 2.1, the SSS matrix A can be denoted by the following generators representation

$$A = \mathcal{SSS}(P_s, R_s, Q_s, D_s, U_s, W_s, V_s). \tag{3}$$

Take $n = 4$, for example, the SSS matrix A has the following representation,

$$\begin{bmatrix} D_1 & U_1V_2^T & U_1W_2V_3^T & U_1W_2W_3V_4^T \\ P_2Q_1^T & D_2 & U_2V_3^T & U_2W_3V_4^T \\ P_3R_2Q_1^T & P_3Q_2^T & D_3 & U_3V_4^T \\ P_4R_3R_2Q_1^T & P_4R_3Q_2^T & P_4Q_3^T & D_4 \end{bmatrix}. \tag{4}$$

The structure of SSS matrices can be exploited so that fast computations in linear computational complexity are enabled, where operations are performed on its generators. Table II lists references that discuss how a given operation can be performed using SSS matrix arithmetic.

Multilevel sequentially semiseparable matrices extend the sequentially semiseparable matrices to multi-dimensional case. Similar to Definition 2.1 for SSS matrices, the generators representation for MSSS matrices, specifically the k-level SSS matrices, is defined by Definition 2.2.

Definition 2.2 ([39])
The matrix A is said to be a k-level SSS matrix if all its generators are $(k-1)$-level SSS matrices. The 1-level SSS matrix is the SSS matrix that satisfies Definition 2.1.

Table I. Generator size for the sequentially semiseparable matrix A in Definition 2.1.

Generators	U_i	W_i	V_i	D_i	P_i	R_i	Q_i
Sizes	$m_i \times k_i$	$k_{i-1} \times k_i$	$m_i \times k_{i-1}$	$m_i \times m_i$	$m_i \times l_i$	$l_{i-1} \times l_i$	$m_i \times l_{i+1}$

Table II. [39] Commonly used operations on sequentially semiseparable matrices.

Operations	Ax	$A \pm B$	AB	A^{-1}	LU	Model reduction	$Lx = b^\star$
References	[34, 35, 38]	[34, 35, 38]	[34, 35, 38]	[9, 37, 40]	[12, 29, 39]	[38, 39]	[38]

$^\star L$ is a lower-triangular SSS matrix.

Within this multilevel framework, generators to represent an MSSS matrix of a higher hierarchy, are themselves MSSS matrices of a lower hierarchy. The one-level SSS matrix is the one of the lowest hierarchy. Basic operations of MSSS matrices are still closed under this structure. In Example 2.1, we use a simple example to show how the lower-level SSS matrices are related with high-level SSS matrices and the correspondence between MSSS matrices and discretized PDEs.

Example 2.1
For the 2D Poisson equation with homogeneous Dirichlet boundary conditions, discretized using the Q_1 finite element method, the stiffness matrix is given by

$$K = \begin{bmatrix} A & B & & & \\ B & A & B & & \\ & B & \ddots & \ddots & \\ & & \ddots & \ddots & B \\ & & & B & A \end{bmatrix}, \text{ where } A = \begin{bmatrix} \frac{8}{3} & -\frac{2}{3} & & & \\ -\frac{2}{3} & \frac{8}{3} & -\frac{2}{3} & & \\ & -\frac{2}{3} & \ddots & \ddots & \\ & & \ddots & \ddots & -\frac{2}{3} \\ & & & -\frac{2}{3} & \frac{8}{3} \end{bmatrix}, \text{ and } B = \begin{bmatrix} -\frac{2}{3} & -\frac{2}{3} & & & \\ -\frac{2}{3} & -\frac{2}{3} & -\frac{2}{3} & & \\ & -\frac{2}{3} & \ddots & \ddots & \\ & & \ddots & \ddots & -\frac{2}{3} \\ & & & -\frac{2}{3} & -\frac{2}{3} \end{bmatrix}.$$

The matrix K is an MSSS (two-level SSS) matrix and can be denoted as

$$K = \mathcal{MSSS}(I,\ 0,\ B,\ A,\ I,\ 0,\ B),$$

where I is an identity matrix and the matrices A and B are 1-level SSS matrices, which can be represented as

$$A = \mathcal{SSS}\left(1,\ 0,\ -\frac{2}{3},\ \frac{8}{3},\ 1,\ 0,\ -\frac{2}{3}\right),$$
$$B = \mathcal{SSS}\left(1,\ 0,\ -\frac{2}{3},\ -\frac{2}{3},\ 1,\ 0,\ -\frac{2}{3}\right).$$

Remark 2.1
It is not necessary for the main diagonal blocks, super-diagonal blocks, or the sub-diagonal blocks of SSS matrices or MSSS matrices to be constant just like Example 2.1. The MSSS matrices can also represent matrices from discretized PDEs with variable coefficients. The sizes of these generators can even be different from each other as long as conditions in Table I are satisfied for the Definition 2.1.

Operations listed in Table II for the SSS matrices can be extended to the MSSS matrices. These operations can be also performed in linear computational complexity. The *LU* factorization can also be performed in a structure preserving way. Given the MSSS matrix K in Example 2.1 for example, the Schur complements for the block *LU* factorization of the stiffness matrix K are computed via the following recurrences,

$$\begin{aligned} S_0 &= A, \\ S_{i+1} &= A - BS_i^{-1}B. \end{aligned} \tag{5}$$

It can be seen that after the first iteration, the Schur complements are not sparse anymore. This makes the standard block *LU* factorization more expensive if it does not use any kind of fill-in minimization reordering. However, if we investigate the MSSS structure of K, we can make use of the SSS structure of its blocks A and B. It has been shown that A and B are SSS matrices; therefore, S_i is also an SSS matrix. If the off-diagonal blocks of S_i have low numerical rank, then S_i can be approximated accurately enough by an SSS matrix with low semiseparable order in linear computational complexity. The semiseparable order will be introduced in the next section.

In [41], it was shown that the off-diagonal blocks of the Schur complements for discretized 2D PDEs with constant coefficients have low numerical rank. And this rank is bounded by a small constant that is independent of the problem size. This makes it efficient to approximate the Schur complements in (5) by SSS matrices with low semiseparable order. By using this approximation, this block factorization can be performed in linear computational complexity. Because of the approximation of the Schur complements for performing the block *LU* factorization, this factorization is an approximate factorization, which can be used as a preconditioner. The details for this block *LU* factorization will be introduced in the next section.

3. Multilevel sequentially semiseparable preconditioners

As previously mentioned, an inexact *LU* factorization can be computed in linear computational complexity by MSSS matrix computations. The semiseparable order defined in Definition 3.1 plays an important role in the MSSS matrix computations. In this paper, we use MATLAB style for matrices notations, i.e., for a matrix A, $A(i:j, s:t)$ selects rows of blocks from i to j and columns of blocks from s to t of A.

Definition 3.1 ([42])
Let

$$\text{rank}\, A(k+1:n, 1:k) = l_k,\ k = 1, 2, \cdots, n-1.$$

The numbers l_k $(k = 1, 2, \cdots, n-1)$ are called the lower order numbers of the matrix A. Let

$$\text{rank}\, A(1:k, k+1:n) = u_k,\ k = 1, 2, \cdots, n-1.$$

The numbers u_k $(k = 1, 2, \cdots, n-1)$ are called the upper order numbers of the matrix A. Set $r^l = \max_k l_k$ and $r^u = \max_k u_k$, where r^l and r^u are called the lower quasiseparable order and the upper quasiseparable order of A, respectively.

Definition 3.2 ([11])
The SSS matrix A with lower and upper semiseparable order r^l and r^u is called block (r^l, r^u) semiseparable.

Definitions 3.3 and 3.4 extend the definitions in Definitions 3.1 and 3.2 for SSS matrices to the MSSS matrices case.

Definition 3.3 ([39])
Let the matrix A be an $N \times N$ block k-level SSS matrix with its generators be $M \times M$ block $(k-1)$-level SSS matrices. Let

$$\operatorname{rank} A(s+1:N, 1:s) = l_s,\ s = 1, 2, \cdots, N-1.$$

The numbers $l_s (s = 1, 2, \cdots, N-1)$ are called the k-level lower order numbers of the matrix A. Let

$$\operatorname{rank} A(1:s, s+1:N) = u_s,\ s = 1, 2, \cdots, N-1.$$

The numbers $u_s (s = 1, 2, \cdots, N-1)$ are called the k-level upper order numbers of the matrix A. Set $r^l = \max_s l_s$ and $r^u = \max_s u^s$, where r^l and r^u are called the k-level lower semiseparable order and the k-level upper semiseparable order for the k-level SSS matrix A, respectively.

Definition 3.4 ([39])
The k-level SSS matrix A with k-level lower and upper semiseparable order r^l and r^u is called k-level block (r^l, r^u) semiseparable.

With the definitions defined earlier, we have the following algorithm to compute the *LU* factorization of a k-level SSS matrix.

Lemma 3.1 ([12, 29])
Let A be a strongly regular $N \times N$ block k-level sequentially semiseparable matrix of k-level block (r^l, r^u) semiseparable and denoted by its generators representation $A = \mathcal{MSSS}(P_s, R_s, Q_s, D_s, U_s, W_s, V_s)$. Here, we say that a matrix is strongly regular by which we mean that the leading principal minors are nonsingular. Let $A = LU$ be its block LU factorization, then,

1. The block lower-triangular factor L is a k-level sequentially semiseparable matrix of k-level block $(r^L, 0)$ semiseparable, and the block upper-triangular factor U is a k-level sequentially semiseparable matrix of k-level block $(0, r^U)$ semiseparable. Moreover, $r^L = r^l$ and $r^U = r^u$.
2. The factors L and U can be denoted by the generators representation

$$L = \mathcal{MSSS}\left(P_s, R_s, \hat{Q}_s, D_s^L, 0, 0, 0\right),$$
$$U = \mathcal{MSSS}\left(0, 0, 0, D_s^U, \hat{U}_s, W_s, V_s\right).$$

where $\hat{Q}_s$, D_s^L, D_s^U and $\hat{U}_s$ are $(k-1)$-level sequentially semiseparable matrices. They are computed by the following algorithm:

Algorithm 1 *LU* factorization of a k-level SSS matrix A

Input: $\{P_s\}_{s=2}^N, \{R_s\}_{s=2}^{N-1}, \{Q_s\}_{s=1}^{N-1}, \{D_s\}_{s=1}^N, \{U_s\}_{s=1}^{N-1}, \{W_s\}_{s=2}^{N-1}, \{Q_s\}_{s=2}^N$

$D_1 = D_1^L D_1^U$ (*LU* factorization of $(k-1)$-level SSS matrix)
Let $\hat{U}_1 = (D_1^L)^{-1} U_1$ and $\hat{Q}_1 = (D_1^L)^{-T} Q_1$
for $i = 2 : N-1$ **do**
  if $i == 2$ **then**
    $M_i = \hat{Q}_{i-1}^T \hat{U}_{i-1}$
  else
    $M_i = \hat{Q}_{i-1}^T \hat{U}_{i-1} + R_{i-1} M_{i-1} W_{i-1}$
  end if
  $\left(D_i - P_i M_i V_i^T\right) = D_i^L D_i^U$ (*LU* factorization of $(k-1)$-level SSS matrix)
  Let $\hat{U}_i = (D_i^L)^{-1}(U_i - P_i M_i W_i)$, $\hat{Q}_i = (D_i^U)^{-T}(Q_i - V_i M_i^T R_i^T)$.
end for
$M_N = \hat{Q}_{N-1}^T \hat{U}_{N-1} + R_{N-1} M_{N-1} W_{N-1}$
$\left(D_N - P_N M_N V_N^T\right) = D_N^L D_N^U$ (*LU* factorization of $(k-1)$-level SSS matrix)

Output: $\{D_s^L\}_{s=1}^N, \{D_s^U\}_{s=1}^N, \left\{\hat{Q}_s\right\}_{s=1}^{N-1}, \left\{\hat{U}_s\right\}_{s=1}^{N-1}$

For the proof of the lemma, we refer to [12, 29].

As explained in [39], to compute the *LU* factorization of a k-level SSS matrix using Algorithm 1, the matrix-matrix operations are performed on its $(k-1)$-level SSS generators. This leads to a growth of the semiseparable order of the $(k-1)$-level SSS generators, which induces an increase of the computational complexity. Model order reduction is therefore necessary to reduce the semiseparable order or keep the semiseparable order under a threshold in the *LU* factorization. For details of the growth of the semiseparable order and the model order reduction, please refer to [39].

With Algorithm 1, we can compute an approximate *LU* factorization in linear complexity with MSSS matrix computations for a wide class of discretized scalar PDEs. For CFD problems, usually, we need to solve a set of coupled PDEs that after discretization gives the 2×2 block system of the following form

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}, \tag{6}$$

where $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{m\times n}$. It is not difficult to verify that the matrices A and B are MSSS matrices for the discrete Stokes and discrete linearized Navier-Stokes equation. The 2×2 block system (6) itself is not an MSSS system but has MSSS blocks. Using Lemma 3.2, we can permute a matrix with MSSS blocks into a single MSSS matrix. This enables us to efficiently compute an *LU* factorization of a permuted saddle-point system with Algorithm 1 by MSSS matrix computations.

Lemma 3.2 ([39])
Let A, B, C, and D be SSS matrices with the following generators representations

$$\begin{aligned} A &= \mathcal{SSS}\left(P_s^a, R_s^a, Q_s^a, D_s^a, U_s^a, W_s^a, V_s^a\right), \\ B &= \mathcal{SSS}\left(P_s^b, R_s^b, Q_s^b, D_s^b, U_s^b, W_s^b, V_s^b\right), \\ C &= \mathcal{SSS}\left(P_s^c, R_s^c, Q_s^c, D_s^c, U_s^c, W_s^c, V_s^c\right), \\ D &= \mathcal{SSS}\left(P_s^d, R_s^d, Q_s^d, D_s^d, U_s^d, W_s^d, V_s^d\right). \end{aligned}$$

Then, there exists a permutation matrix Ψ with $\Psi\Psi^T = \Psi^T\Psi = I$ where I is an identity matrix with proper size such that

$$\mathcal{T} = \Psi \begin{bmatrix} A & B \\ C & D \end{bmatrix} \Psi^T$$

and the matrix $\mathcal{T}$ is an SSS matrix. Its generators representation are given by

$$\mathcal{T} = \mathcal{SSS}(P_s^t, R_s^t, Q_s^t, D_s^t, U_s^t, W_s^t, V_s^t),$$

where $P_s^t = \begin{bmatrix} P_s^a & P_s^b & 0 & 0 \\ 0 & 0 & P_s^c & P_s^d \end{bmatrix}$, $Q_s^t = \begin{bmatrix} Q_s^a & 0 & Q_s^c & 0 \\ 0 & Q_s^b & 0 & Q_s^d \end{bmatrix}$, $D_s^t = \begin{bmatrix} D_s^a & D_s^b \\ D_s^c & D_s^d \end{bmatrix}$, $U_s^t = \begin{bmatrix} U_s^a & U_s^b & 0 & 0 \\ 0 & 0 & U_s^c & U_s^d \end{bmatrix}$, $V_s^t = \begin{bmatrix} V_s^a & 0 & V_s^c & 0 \\ 0 & V_s^b & 0 & V_s^d \end{bmatrix}$, $W_s^t = \begin{bmatrix} W_s^a & & & \\ & W_s^b & & \\ & & W_s^c & \\ & & & W_s^d \end{bmatrix}$, $R_s^t = \begin{bmatrix} R_s^a & & & \\ & R_s^b & & \\ & & R_s^c & \\ & & & R_s^d \end{bmatrix}$.

For the proof of Lemma 3.2, we refer to [39].

In the following, we use an example to show in details how to do such permutation mentioned in Lemma 3.2. Take the 3 × 3 block SSS matrices for example, where

$$A = \begin{bmatrix} D_1^a & U_1^a V_2^{aT} & U_1^a W_2^a V_3^{aT} \\ P_2^a Q_1^{aT} & D_2^a & U_2^a V_3^{aT} \\ P_3^a R_2^a Q_1^{aT} & P_3^a Q_2^{aT} & D_3^a \end{bmatrix}, \quad B = \begin{bmatrix} D_1^b & U_1^b V_2^{bT} & U_1^b W_2^b V_3^{bT} \\ P_2^b Q_1^{bT} & D_2^b & U_2^b V_3^{bT} \\ P_3^b R_2^b Q_1^{bT} & P_3^b Q_2^{bT} & D_3^b \end{bmatrix}$$

$$C = \begin{bmatrix} D_1^c & U_1^c V_2^{cT} & U_1^c W_2^c V_3^{cT} \\ P_2^c Q_1^{cT} & D_2^c & U_2^c V_3^{cT} \\ P_3^c R_2^c Q_1^{cT} & P_3^c Q_2^{cT} & D_3^c \end{bmatrix}, \quad D = \begin{bmatrix} D_1^d & U_1^d V_2^{dT} & U_1^d W_2^d V_3^{dT} \\ P_2^d Q_1^{dT} & D_2^d & U_2^d V_3^{dT} \\ P_3^d R_2^d Q_1^{dT} & P_3^d Q_2^{dT} & D_3^d \end{bmatrix}.$$

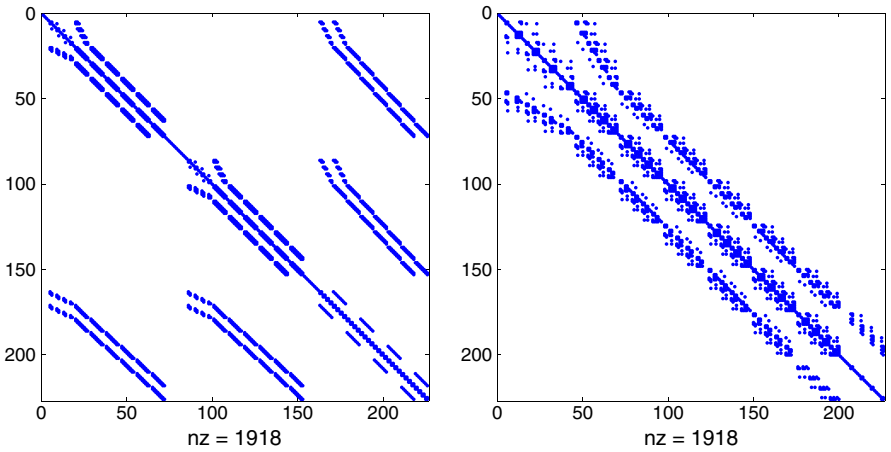
Then, there exists a permutation matrix Ψ of the following form

$$\Psi = \left[\begin{bmatrix} I_n \\ 0 \end{bmatrix} \otimes I_3, \begin{bmatrix} 0 \\ I_n \end{bmatrix} \otimes I_3 \right],$$

where $I_n \in \mathbb{R}^{n\times n}$ is the identity matrix and $\otimes$ denotes the Kronecker product.

After the permutation, the permuted block matrix becomes

$$\mathcal{T} = \Psi \begin{bmatrix} A & B \\ C & D \end{bmatrix} \Psi^T = \begin{bmatrix} D_1^t & U_1^t V_2^{tT} & U_1^t W_2^t V_3^{tT} \\ P_2^t Q_1^{tT} & D_2^t & U_2^t V_3^{tT} \\ P_3^t R_2^t Q_1^{tT} & P_3^t Q_2^{tT} & D_3^t \end{bmatrix},$$



(a) Before permutation (b) After permutation

Figure 1. Structure of the saddle-point system of Stokes equation before and after permutation for $h = 2^{-2}$ by the $Q_1 - P_0$ finite element discretization. [Colour figure can be viewed at wileyonlinelibrary.com]

where the generators for SSS matrix $\mathcal{T}$ satisfy the following relations,

$$P_s^t = \begin{bmatrix} P_s^a & P_s^b & 0 & 0 \\ 0 & 0 & P_s^c & P_s^d \end{bmatrix}, \quad R_s^t = \begin{bmatrix} R_s^a & & & \\ & R_s^b & & \\ & & R_s^c & \\ & & & R_s^d \end{bmatrix}, \quad Q_s^t = \begin{bmatrix} Q_s^a & 0 & Q_s^c & 0 \\ 0 & Q_s^b & 0 & Q_s^d \end{bmatrix}, \quad U_s^t = \begin{bmatrix} U_s^a & U_s^b & 0 & 0 \\ 0 & 0 & U_s^c & U_s^d \end{bmatrix},$$

$$W_s^t = \begin{bmatrix} W_s^a & & & \\ & W_s^b & & \\ & & W_s^c & \\ & & & W_s^d \end{bmatrix}, \quad V_s^t = \begin{bmatrix} V_s^a & 0 & V_s^c & 0 \\ 0 & V_s^b & 0 & V_s^d \end{bmatrix}, \quad D_s^t = \begin{bmatrix} D_s^a & D_s^b \\ D_s^c & D_s^d \end{bmatrix},$$

and $s = 1,\ 2$ for U_s^t and Q_s^t, $s = 2$ for W_s^t and R_s^t, $s = 2,\ 3$ for V_s^t and P_s^t, and $s = 1,\ 2,\ 3$ for D_s^t.

One can apply Lemma 3.2 to permute a matrix with SSS blocks into a single SSS matrix by using a permutation matrix Ψ. Moreover, this permutation matrix is not explicitly multiplied on both sides of the matrix to be permuted. Generators of the permuted matrix is just a re-grouping of the generators of its SSS blocks. Very few cost is consumed for such permutation.

Remark 3.1

Extending Lemma 3.2 to the k-level SSS matrix case is also possible. If A, B, C, and D are k-level SSS matrices, then their generators are $(k-1)$-level SSS matrices. For the permuted k-level SSS matrix $\mathcal{T}$, its $(k-1)$-level SSS matrix generators with $(k-1)$-level SSS matrix blocks are permuted into a single $(k-1)$-level SSS matrix by applying Lemma 3.2 recursively from the lowest level to the top level.

By applying Lemma 3.2, the saddle-point system structure of the discretized Stokes equation using the $Q_1 - P_0$ finite element method discretization on a square domain before and after permutation is shown in Figure 1.

4. Numerical experiments

In this section, we test the performance of MSSS preconditioning techniques on CFD benchmark problems using IFISS. The convection–diffusion, Stokes, and Navier-Stokes problems are considered. The AMG and GMG methods in IFISS are also used to compare their performance with that of the MSSS preconditioners. The MSSS matrix computations are implemented under MATLAB. All the numerical experiments are performed in MATLAB 2011b on a desktop of Intel Core i5 CPU of 3.10 GHz and 16 Gb memory with the Debian GNU/Linux 7.2 system. The iterative solution methods are terminated if the 2-norm of the residual is reduced by a factor of 10^{-6} or the maximum number of iterations, which is set to 100, is reached. The MSSS matrix computation toolbox,[§] and the test code are available at `http://ta.twi.tudelft.nl/nw/users/yueqiu/software.html`.

In the tables that give numerical results, the 'preconditioning' column reports the time to compute the approximate *LU* factorization for MSSS preconditioners or the time to setup the multigrid for the AMG or GMG method. IDR(s) [6,43] is chosen as the iterative solution method. The 'IDR(4)' column reports the time to solve the preconditioned system. The total time is the sum of the time to compute the preconditioner and the time to solve the preconditioned system, which is reported in the 'total' column. All the columns concerning time in this paper are measured in seconds.

[§] *MSSS Matrix Computation Toolbox, version 0.7, 2013.*

4.1. *Convection–diffusion problem*

We first consider the convection–diffusion problem described in Example 4.1, which is given as the example 3.1.4 in [44]. The details of the discretization of the convection–diffusion equation can also be found in [44]. To investigate the performance of the MSSS preconditioning technique, we first consider the diffusion-dominated case that corresponds to the viscosity parameter $\epsilon = \frac{1}{200}$. Next, we consider the convection-dominated case, which has a viscosity parameter $\epsilon = 10^{-4}$. These experiments are also performed using the AMG and GMG method for comparison.

Example 4.1 ([44])
Zero source term, recirculating wind, characteristic boundary layers.

$$\begin{aligned} -\epsilon\nabla^2 u + \vec{\omega}\cdot\nabla u &= f \text{ in } \Omega \\ u &= u_D \text{ on } \Gamma_D \\ \frac{\partial u}{\partial n} &= g_N \text{ on } \Gamma_N \end{aligned} \tag{7}$$

where $\Omega = \{(x,y)| -1 \le x \le 1, -1 \le y \le 1\}$, $\vec{\omega} = \big(2y(1-x^2),\ -2x(1-y^2)\big)$, $f = 0$. Dirichlet boundary are imposed everywhere and there are discontinuities at the two corners of the wall, $x = 1$, $y = \pm 1$.

We use the Q_1 finite element method to discretize the convection–diffusion equation. First, we consider a moderate value for the viscosity parameter $\epsilon = \frac{1}{200}$, the computational results by the MSSS preconditioner and the AMG and GMG method are listed in Tables III–V. The maximum semiseparable order for the MSSS preconditioner is in the brackets that follow after the mesh size. The smoother for the AMG and GMG method is chosen as the incomplete *LU* factorization (`ilu(1)`). The solution corresponds to the mesh size $h = 2^{-7}$ is shown in Figure 2.

Table III illustrates that the MSSS preconditioner gives mesh size independent convergence for the convection–diffusion problem with $\epsilon = \frac{1}{200}$. Both the time to compute the approximate *LU* factorization by MSSS matrix computations and the time to solve the preconditioned system scale linearly with the problem size. Compared with the computational results in Tables IV and V for the AMG and GMG method, we can see that the time of the AMG method setup is much bigger than that of the MSSS preconditioner, while the time for the GMG method is much smaller than for the MSSS preconditioner. Both the AMG and GMG methods give mesh size independent convergence. Table IV illustrates that the computational complexity for setting up the AMG method grows with the

Table III. Multilevel sequentially semiseparable preconditioner for $\epsilon = \frac{1}{200}$.

Mesh size	Problem size	No. iter.	Preconditioning (sec.)	IDR(4) (sec.)	Total (sec.)
$2^{-4}(4)$	1.09e+03	4	0.48	0.31	0.79
$2^{-5}(5)$	4.23e+03	4	1.22	0.74	1.96
$2^{-6}(5)$	1.66e+04	4	4.16	2.20	6.36
$2^{-7}(7)$	6.60e+04	4	16.11	8.09	24.20
$2^{-8}(7)$	2.63e+05	4	63.15	30.42	93.58

IDR, induced dimension reduction.

Table IV. Algebraic multigrid method for $\epsilon = \frac{1}{200}$.

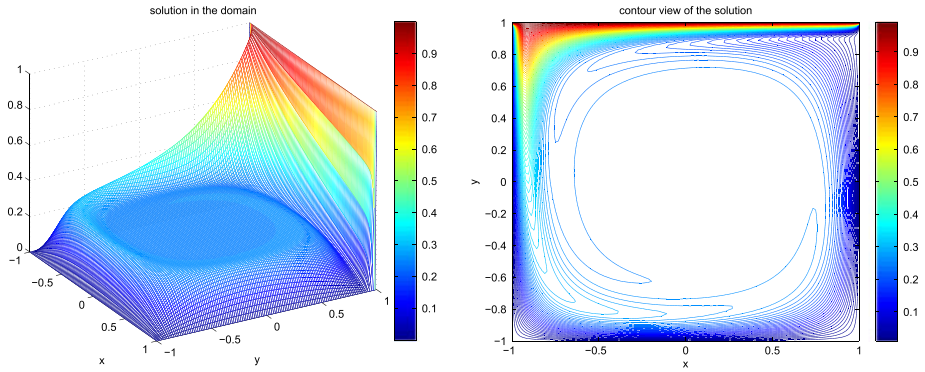
Mesh size	Problem size	No. iter.	Preconditioning (sec.)	IDR(4) (sec.)	Total (sec.)
2^{-4}	1.09e+03	8	0.49	0.06	0.55
2^{-5}	4.23e+03	4	2.38	0.05	2.43
2^{-6}	1.66e+04	4	14.30	0.17	14.47
2^{-7}	6.60e+04	4	127.71	0.28	127.99
2^{-8}	2.63e+05	4	2513.11	1.53	2514.64

IDR, induced dimension reduction.

Table V. Geometric multigrid method for $\epsilon = \frac{1}{200}$.

Mesh size	Problem size	No. iter.	Preconditioning (sec.)	IDR(4) (sec.)	Total (sec.)
2^{-4}	1.09e+03	5	0.02	0.02	0.04
2^{-5}	4.23e+03	4	0.05	0.03	0.08
2^{-6}	1.66e+04	3	0.12	0.04	0.16
2^{-7}	6.60e+04	3	0.46	0.08	0.54
2^{-8}	2.63e+05	3	2.72	0.31	3.03

IDR, induced dimension reduction.



(a) Solution of u. (b) Contour view.

Figure 2. Solution of test problem 4.1 for $\epsilon = \frac{1}{200}$ and $h = 2^{-7}$. [Colour figure can be viewed at wileyonlinelibrary.com]

Table VI. Multilevel sequentially semiseparable preconditioner with $\epsilon = 10^{-4}$.

Mesh size	Problem size	No. iter.	Preconditioning (sec.)	IDR(4) (sec.)	Total (sec.)
$2^{-4}(12)$	1.09e+03	14	0.46	0.84	1.30
$2^{-5}(24)$	4.23e+03	11	1.61	1.89	3.50
$2^{-6}(26)$	1.66e+04	12	6.68	6.80	13.48
$2^{-7}(26)$	6.60e+04	14	29.90	16.68	46.58
$2^{-8}(10)$	2.63e+05	5	66.63	38.22	104.85

IDR, induced dimension reduction.

Table VII. Algebraic multigrid method with $\epsilon = 10^{-4}$.

Mesh size	Problem size	No. iter.	Preconditioning (sec.)	IDR(4) (sec.)	Total (sec.)
2^{-4}	1.09e+03	100	0.49	No convergence	-
2^{-5}	4.23e+03	100	2.41	No convergence	-
2^{-6}	1.66e+04	100	14.53	No convergence	-
2^{-7}	6.60e+04	100	131.27	No convergence	-
2^{-8}	2.63e+05	100	2498.11	No convergence	-

IDR, induced dimension reduction.

Table VIII. Geometric multigrid method with $\epsilon = 10^{-4}$.

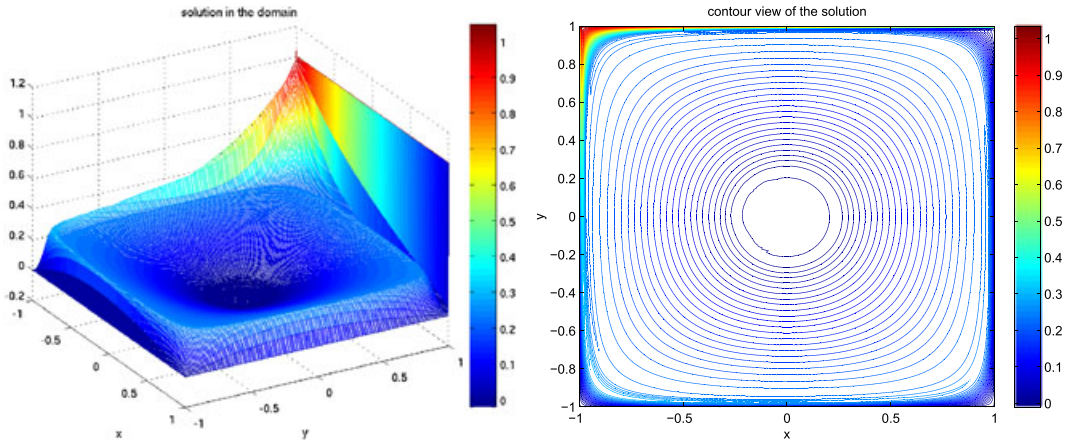
Mesh size	Problem size	No. iter.	Preconditioning (sec.)	IDR(4) (sec.)	Total (sec.)
2^{-4}	1.09e+03	100	0.02	No convergence	-
2^{-5}	4.23e+03	100	0.04	No convergence	-
2^{-6}	1.66e+04	100	0.12	No convergence	-
2^{-7}	6.60e+04	100	0.48	No convergence	-
2^{-8}	2.63e+05	100	2.81	No convergence	-

IDR, induced dimension reduction.

problem size and is bigger than linear. This is most probably due to the fact that the AMG method implemented in IFISS is not of linear computational complexity.

Next, we test the convection-dominated case with the viscosity parameter $\epsilon = 10^{-4}$ for the MSSS preconditioner, the AMG and GMG method. The computational results are reported in Tables VI–VIII. The solution for the mesh size $h = 2^{-7}$ is shown in Figure 3.

For the convection-dominated test case, the system is ill-conditioned. It is therefore more difficult to compute a good enough preconditioner. A larger semiseparable order is needed to compute an accurate enough approximation compared with the case of $\epsilon = \frac{1}{200}$. This is illustrated by comparing the semiseparable orders in Table III with the semiseparable orders in Table VI. Because of the bigger semiseparable order, more computational effort is needed. Even the time to compute the preconditioner and to solve the preconditioned system is bigger than the time for larger ϵ, the computational time still scales linearly with the problem size. Because of the ill-conditioning of the problem to solve, both the AMG and GMG methods fail.



(a) Solution of u. (b) Contour view.

Figure 3. Solution of test problem 4.1 for $\epsilon = 10^{-4}$ and $h = 2^{-7}$. [Colour figure can be viewed at wileyonlinelibrary.com]

Remark 4.1
Compared with the AMG and GMG methods, the MSSS preconditioner is more robust. Moreover, the MSSS preconditioning technique is considerably faster than the AMG method.

4.2. Stokes problem

Next, we evaluate the performance of the MSSS preconditioner for the lid-driven cavity problem of the Stokes equation described by Example 4.2, which is given as Example 5.1.3 in [44]. Mixed finite elements are used for discretization.

Example 4.2 ([44])
Lid-driven cavity problem, enclosed flow boundary condition.

$$\begin{aligned} -\nabla^2 u + \nabla p &= \vec{0} \\ \nabla \cdot u &= 0 \quad \text{in } \Omega \\ \vec{u} &= \vec{\omega} \text{ on } \Gamma_D \\ \frac{\partial \vec{u}}{\partial n} - \vec{n} p &= \vec{s} \text{ on } \Gamma_N \end{aligned} \tag{8}$$

in a square domain $\{(x,y)| -1 \le x \le 1, -1 \le y \le 1\}$, where the regularized cavity condition $\{y = 1; -1 \le x \le 1 | u_x = 1 - x^4\}$ is satisfied.

The discretized Stokes equation using $Q_1 - P_0$ finite element method has the following saddle-point system form

$$\begin{bmatrix} K & B^T \\ B & -S_t \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}, \tag{9}$$

where $K \in \mathbb{R}^{2n_u \times 2n_u}$ is the vector Laplace matrix, $B \in \mathbb{R}^{n_p \times 2n_u}$ is the divergence matrix, $S_t \in \mathbb{R}^{n_p \times n_p}$ is the stabilization term to satisfy the inf-sub condition for the Stokes problem, n_u is the number of velocity grid points, and n_p is the number of pressure grid points.

Standard preconditioning techniques for the saddle point system (9) are the block diagonal preconditioner $\mathcal{P}_1$ or the block lower-triangular preconditioner $\mathcal{P}_2$, they are described by

$$\mathcal{P}_1 = \begin{bmatrix} K & \\ & -S \end{bmatrix}, \quad \mathcal{P}_2 = \begin{bmatrix} K & \\ B & S \end{bmatrix},$$

where $S = -S_t - BK^{-1}B^T$ is the Schur complement. When the block diagonal preconditioner $\mathcal{P}_1$ is applied, the preconditioned system has three distinct eigenvalues, and the GMRES computes the exact solution in at most three steps. When the block lower-triangular preconditioner $\mathcal{P}_2$ is applied, the preconditioned system has two distinct eigenvalues. In this case, GMRES computes the exact solution in at most two steps. We refer to [45] for an extensive study for such block preconditioners.

In general, the Schur complement S is difficult to compute because of the high computational complexity. A standard way is to compute an approximation that has an equivalent spectrum with the Schur complement. However, the Schur complement approximation is problem dependent. It is still a big challenge to compute a good approximation of the Schur complement for some applications, such as optimal in-domain control of the Stokes equation [46]. For the discrete Stokes equation, it is shown in [47] that the Schur

complement has an equivalent spectrum with the pressure mass matrix M_p, i.e., the relation

$$\gamma^2 \leq \frac{x^T B K^{-1} B^T x}{x^T M_p x} \leq \Gamma^2, \quad \forall x \in \mathbb{R}^{n_p} \backslash \{\mathbf{0}\} \tag{10}$$

holds, where γ and Γ are constants that are independent of the mesh size h. Thus, the block preconditioners for the Stokes could be chosen as

$$\mathcal{P}_1 = \begin{bmatrix} K & \\ & M_p \end{bmatrix}, \quad \mathcal{P}_2 = \begin{bmatrix} K & \\ B & -M_p \end{bmatrix}. \tag{11}$$

This type of preconditioners are called the Silvester–Wathen preconditioner and is widely studied for the Stokes problems in [44,47,48].

Because the diagonal blocks of the block preconditioners are MSSS matrices, a natural way is to apply the Silvester–Wathen preconditioner to iteratively solve the discrete Stokes Eq. (9) by using MSSS matrix computations. In addition, the global system matrix of the discrete Stokes Eq. (9) has MSSS blocks, we can also apply the global MSSS preconditioner to iteratively solve the global system (9). Both MSSS block preconditioner and MSSS global preconditioner are studied in this part.

First, we test the block MSSS preconditioner case. Choose the block diagonal preconditioner as

$$\mathcal{P}_1 = \begin{bmatrix} \hat{K} & \\ & \hat{M}_p \end{bmatrix} \tag{12}$$

where $\hat{K}$ is the approximation of K and $\hat{M}_p$ is the lumped pressure mass matrix M_p. For comparison, the AMG and GMG methods, together with the MSSS matrix computations, are performed to approximate K. Because of the symmetric definiteness of the block diagonal preconditioner and the symmetry but indefiniteness of the saddle point system, minimal residual method [49] is chosen as the iterative solver. The results for block MSSS preconditioner are listed in Table IX and for the AMG and GMG methods are given in Tables X and XI. The smoother for the AMG and GMG method is chosen as the 'point damped Jacobi'.

Results in Tables IX and XI illustrate that the block preconditioners by MSSS matrix computations, together with the AMG and GMG methods, give mesh size independent convergence. The time to compute the block MSSS preconditioner and to solve the preconditioned system scale linearly with the problem size. This can be verified by Table IX. The setup time for the AMG method is still bigger than linear, while it is still not clear whether the AMG method implemented in IFISS has linear computational complexity or not.

Table IX. Silvester–Wathen preconditioner for the Stokes equation by multilevel sequentially semiseparable matrix computations.

Mesh size	Problem size	No. iter.	Preconditioning (sec.)	MINRES (sec.)	Total (sec.)
$2^{-4}(12)$	3.20e+03	33	0.36	3.82	4.18
$2^{-5}(12)$	1.25e+04	33	1.17	11.21	12.38
$2^{-6}(12)$	4.97e+04	33	3.97	37.15	41.12
$2^{-7}(12)$	1.98e+05	35	15.04	140.06	155.10
$2^{-8}(14)$	7.88e+05	33	62.55	558.64	621.19

MINRES, minimal residual.

Table X. Silvester–Wathen preconditioner for the Stokes equation by algebraic multigrid method.

Mesh size	Problem size	No. iter.	Preconditioning (sec.)	MINRES (sec.)	Total (sec.)
2^{-4}	3.20e+03	36	0.18	0.19	0.37
2^{-5}	1.25e+04	38	0.69	0.33	1.02
2^{-6}	4.97e+04	40	6.76	0.83	7.59
2^{-7}	1.98e+05	40	45.72	3.07	48.79
2^{-8}	7.88e+05	37	875.73	9.68	885.41

MINRES, minimal residual.

Table XI. Silvester–Wathen preconditioner for the Stokes equation by geometric multigrid method.

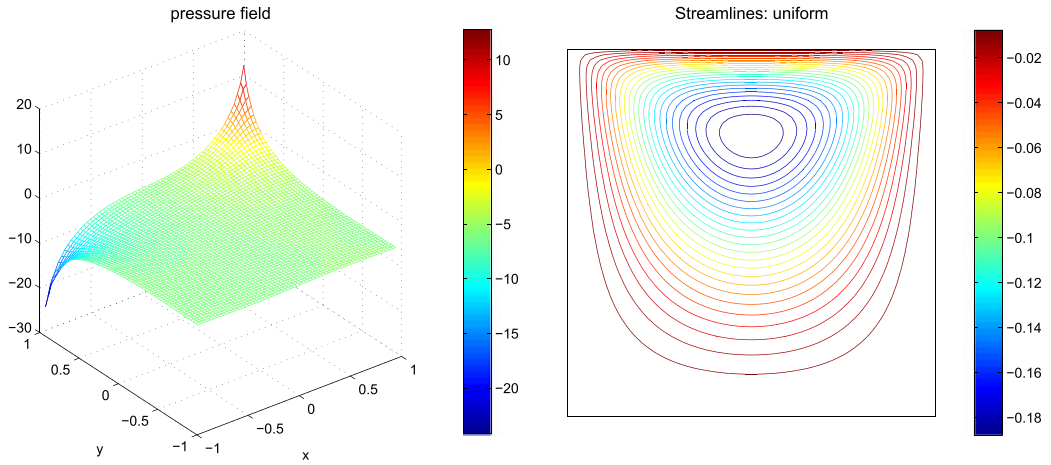
Mesh size	Problem size	No. iter.	Preconditioning (sec.)	MINRES (sec.)	Total (sec.)
2^{-4}	3.20e+03	34	0.09	0.09	0.18
2^{-5}	1.25e+04	34	0.14	0.28	0.42
2^{-6}	4.97e+04	32	0.61	0.58	1.19
2^{-7}	1.98e+05	32	2.01	2.00	4.01
2^{-8}	7.88e+05	30	3.26	7.38	10.64

MINRES, minimal residual.

Table XII. Global preconditioner for the permuted Stokes equation.

Mesh size	Problem size	No. iter.	Preconditioning(sec.)	IDR(4)sec.	Total(sec.)
2^{-4} (4)	3.20e+03	5	0.41	0.34	0.75
2^{-5} (6)	1.25e+04	5	1.29	0.94	2.23
2^{-6} (7)	4.97e+04	5	4.42	3.06	7.48
2^{-7} (9)	1.98e+05	4	16.47	9.01	25.48
2^{-8} (10)	7.88e+05	5	67.50	36.29	103.79

IDR, induced dimension reduction.



(a) Pressure distribution. (b) Streamline.

Figure 4. Solution of test example 4.2 for multilevel sequentially semiseparable preconditioners. [Colour figure can be viewed at wileyonlinelibrary.com]

For the block MSSS preconditioner, most time was spent in solving the preconditioned system. This is mainly due to the big overhead of the Matlab implementation in each iteration to solve the preconditioned system. Less time is needed if the number of iterations is reduced. Next, we focus on the global MSSS preconditioner for iteratively solving the Stokes system (9).

It is not difficult to verify that for the discrete Stokes system (9), all the matrix blocks K, B, and S_t are MSSS matrices. Thus, we can permute the saddle-point system (9) with MSSS blocks into a single MSSS system. Then, we can compute an approximate LU factorization for the global MSSS system. Because of the indefiniteness of the global preconditioner, IDR(s) is chosen as the iterative solver. The computational results for the global preconditioner are listed in Table XII. The solution of the pressure field and the streamlines are shown in Figure 4.

Computational results in Table XII show that the computational time scales linearly with the problem size for both computing the preconditioner and solving the preconditioned system. Meanwhile, the global MSSS preconditioner also gives mesh size independent convergence.

Compare the results for the block preconditioners shown in Tables IX–XI with the results for the global MSSS preconditioners in Table XII, we can see that the number of iterations for the global MSSS preconditioner is much more reduced. Thus, the time to solve the preconditioned system by the global MSSS preconditioner is also much less than the time for the block MSSS preconditioner.

Remark 4.2

The global MSSS preconditioner performs much better than the Silvester–Wathen preconditioner by the AMG method for the solution of middle-size and large-size discrete Stokes equation. Even the number of iterations for the Silvester–Wathen preconditioner by the GMG method is bigger than that for the global MSSS preconditioner, the total time is less. The Silvester–Wathen preconditioner by the GMG method seems appealing for the discrete Stokes equation.

4.3. Navier-Stokes problem

The last example we consider is the lid-driven cavity problem of the Navier-Stokes equation that is given in Example 4.3. It is introduced as Example 7.1.3 in [44].

Example 4.3 ([44])

Lid-driven cavity problem, enclosed flow boundary condition.

$$\begin{aligned} -\nu\nabla^2\vec{u} + \vec{u}\cdot\nabla\vec{u} + \nabla p &= \vec{f} \\ \nabla\cdot\vec{u} &= 0 \end{aligned} \tag{13}$$

with the boundary conditioners

$$\vec{u} = \vec{\omega} \text{ on } \Gamma_D$$
$$\nu \frac{\partial \vec{u}}{\partial n} - \vec{n} p = \vec{0} \text{ on } \Gamma_N$$

in a square domain $\{(x,y)| -1 \le x \le 1, -1 \le y \le 1\}$, where the regularized cavity condition $\{y = 1; -1 \le x \le 1 | u_x = 1 - x^4\}$ is satisfied.

Note that the Navier-Stokes equation is a nonlinear equation. To compute the solution numerically, the Navier-Stokes equation needs to be linearized and discretized. Details about the linearization and finite element discretization are described in [44]. In this paper, we use the Newton method to linearize and the $Q_1 - P_0$ finite element method to discretize. At each linearized step, we need to solve a linear system of the following form

$$\begin{bmatrix} \nu K_x + N + W_{xx} & W_{xy} & B_x^T \\ W_{yx} & \nu K_y + N + W_{yy} & B_y^T \\ B_x & B_y & -\frac{1}{\nu} S_t \end{bmatrix} \begin{bmatrix} \Delta u_x \\ \Delta u_y \\ \Delta p \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \\ g \end{bmatrix}, \tag{14}$$

where K_x, K_y are scalar Laplace matrices, N is the scalar convection matrix, W_{xx}, W_{xy}, W_{yx}, W_{yy} represent weak derivatives of the velocity u_x and u_y in the x and y directions, B_x and B_y are the divergence matrices in the x and y directions, and S_t is a stabilization matrix of the $Q_1 - P_0$ type. Due to the difficulty to compute a good enough approximation of the Schur complement for system (14), preconditioning of the Navier-Stokes equation is still a big challenge and a hot topic in research and engineering. Some efforts to compute efficient approximation of the Schur complement for the Navier-Stokes equation can be found in [44, 50, 51].

The generic form of system (14) can be written as

$$\begin{bmatrix} \mathbf{F} & \mathbf{B}^T \\ \mathbf{B} & -\frac{1}{\nu} S_t \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ p \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ g \end{bmatrix} \tag{15}$$

where **F**, **B** in (15) satisfy some partition rules of the matrix in (14). One of the standard block preconditioners for the linearized Navier-Stokes Eq. (15) is called the pressure convection–diffusion (PCD) preconditioner, which is discussed in [44, 50]. The PCD preconditioner can be written as

$$\mathcal{P} = \begin{bmatrix} \mathbf{F} & \mathbf{B}^T \\ \mathbf{0} & -S \end{bmatrix} \tag{16}$$

where S is the equivalent Schur complement and is given by

$$S = L_p A_p^{-1} M_p. \tag{17}$$

Here, A_p and L_p are the convection–diffusion operator and Laplace operator in the finite dimensional solution space of the pressure with some prescribed boundary conditions, M_p denotes the pressure mass matrix.

We also note that all the matrix blocks in (14) have MSSS structure; thus, we can permute the block system (14) into a single MSSS system and compute an approximate *LU* factorization of the global system. This gives us the global MSSS preconditioner as introduced in Section 4.2. To test the performance of the global MSSS preconditioner, we solve the system (14) at the second Newton step. For comparison, we also carry out numerical experiments to solve Example 4.3 at the second Newton step by the PCD preconditioner (16). For the PCD preconditioner (16), **F** and S are approximated by the multigrid method.

Because the GMG method is not implemented in IFISS, only the results of the PCD preconditioner computed by the AMG method are reported. Due to the quadratic convergence of the Newton method, it is not quite necessary to solve the system up to a very high accuracy at each linearized step. Thus, the stop criteria is set as the 2-norm of the residual is reduced by a factor of 10^{-4} at each linearized step.

First, we set the viscosity parameter ν to be 10^{-1}, the computational results for the global MSSS preconditioner and PCD preconditioner by the AMG method are given in Tables XIII and XIV.

Table XIII. Global multilevel sequentially semiseparable preconditioner for the second Newton step with $\nu = 10^{-1}$.

Mesh size	Problem size	No. iter.	Preconditioning (sec.)	IDR(4) (sec.)	Total (sec.)
$2^{-4}(6)$	3.20e+03	3	0.43	0.22	0.65
$2^{-5}(7)$	1.25e+04	3	1.33	0.59	1.92
$2^{-6}(7)$	4.97e+04	3	4.51	1.88	6.39
$2^{-7}(9)$	1.98e+05	3	19.47	6.75	26.22
$2^{-8}(11)$	7.88e+05	3	78.84	26.63	105.17

IDR, induced dimension reduction.

Table XIV. Pressure convection–diffusion preconditioner by the algebraic multigrid method for the second Newton step with $\nu = 10^{-1}$.

Mesh size	Problem size	No. iter.	Preconditioning (sec.)	IDR(4) (sec.)	Total (sec.)
2^{-4}	3.20e+03	21	0.78	0.12	0.90
2^{-5}	1.25e+04	24	4.30	0.25	4.55
2^{-6}	4.97e+04	23	39.98	0.67	40.65
2^{-7}	1.98e+05	24	631.75	2.72	634.47
2^{-8}	7.88e+05	24	4740.51	9.48	4749.99

IDR, induced dimension reduction.

Table XV. Global multilevel sequentially semiseparable preconditioner for the second Newton step with $\nu = 10^{-2}$.

Mesh size	Problem size	No. iter.	Preconditioning (sec.)	IDR(4) (sec.)	Total (sec.)
$2^{-4}(6)$	3.20e+03	3	0.39	0.14	0.53
$2^{-5}(6)$	1.25e+04	4	1.27	0.64	1.91
$2^{-6}(8)$	4.97e+04	3	4.41	1.83	6.24
$2^{-7}(10)$	1.98e+05	3	18.51	7.70	26.21
$2^{-8}(10)$	7.88e+05	3	75.31	31.58	106.89

IDR, induced dimension reduction.

Table XVI. Pressure convection–diffusion preconditioner by the algebraic multigrid method for the second Newton step with $\nu = 10^{-2}$.

Mesh size	Problem size	No. iter.	Preconditioning (sec.)	IDR(4) (sec.)	Total (sec.)
2^{-4}	3.20e+03	53	1.63	0.32	1.95
2^{-5}	1.25e+04	49	6.29	0.65	6.94
2^{-6}	4.97e+04	51	38.72	1.60	40.32
2^{-7}	1.98e+05	50	440.82	6.31	447.13
2^{-8}	7.88e+05	51	4561.32	26.14	4587.46

IDR, induced dimension reduction.

Numerical results in Tables XIII and XIV illustrate that both preconditioners give mesh size independent convergence. The number of iterations is much more reduced by the global MSSS preconditioner. Moreover, the computational time for the global MSSS preconditioner scales linearly with the problem size. However, this linear computational complexity property does not hold for the PCD preconditioner by the AMG method. This is illustrated by the computational time in Table XIV. We can also find that the global MSSS preconditioner is much faster than the PCD preconditioner by the AMG method.

To study the performance of both preconditioners for bigger Reynolds number, we decrease the viscosity parameter ν to 10^{-2}. The computational results are reported in Tables XV and XVI.

For bigger Reynolds number, both the global MSSS preconditioner and PCD preconditioner by the AMG method gives mesh size independent convergence. This is verified by the numerical results listed in Tables XV and XVI. In addition, the computational time for the global preconditioner is linear with the problem size while the PCD preconditioner by the AMG method does not have such linear computational complexity. Tables XV and XVI show that the global MSSS preconditioner is much faster than the PCD preconditioner by the AMG method.

Remark 4.3

According to the computational results for different Reynolds number in Tables XIII–XVI, we can find that the global MSSS preconditioner not only gives mesh size independent convergence but also gives Reynolds number independent convergence. However, the PCD preconditioner does not have the Reynolds number independent convergence property. Meanwhile, the global MSSS preconditioner behaves the linear computational complexity with the problem size while the PCD by the AMG method preconditioner does not have such linear computational complexity. Moreover, the global MSSS preconditioner is much faster and more robust than the PCD preconditioner by the AMG method.

5. Conclusions

In this paper, we have studied a new class of preconditioners for CFD problems. This type of preconditioners exploits the MSSS structure of the system matrix. By making use of the MSSS matrix computations, we can compute efficient preconditioners for CFD problems

in linear computational complexity. Compared with the standard block preconditioners for the discrete Stokes equation and linearized Navier-Stokes equation, we make use of the global MSSS structure of the system matrix. This avoids approximating the Schur complement explicitly, which is a big advantage over standard block preconditioners.

We apply the AMG and GMG methods to the CFD benchmark problems in IFISS to evaluate the performance of the MSSS preconditioners. Numerical experiments show that the global MSSS preconditioner gives not only mesh size independent but also viscosity parameter and Reynolds number independent convergence, while the standard preconditioners in IFISS do not yield viscosity parameter and Reynolds number independent convergence. For the convection–diffusion equation, the MSSS preconditioner is much faster and more robust than the AMG method. While the GMG method is faster for big viscosity parameter than the MSSS preconditioner. However, the GMG method fails to solve the convection-dominated convection–diffusion problem. For the Stokes equation, the GMG method is competitive among the AMG method and the MSSS preconditioning technique. For the Navier-Stokes equation, the global MSSS preconditioner is much faster and more robust than the AMG method.

The mesh size and Reynolds number independent convergence of the global MSSS preconditioner is still an open problem and is the ongoing research of the authors. Some recent efforts to analyze the preconditioner can be found in [52]. In this reference, Napov explains that the accuracy of the incomplete Cholesky factorization by SSS matrix computations depends only on the approximation accuracy of the off-diagonal blocks. The author also gives an analytical upper bound of the condition number of the preconditioned system. It is shown that the eigenvalues of the preconditioned systems are clustered around 1 and the radius of the clustering depends only on the accuracy of the approximation of the off-diagonal blocks, while this accuracy is directly related with the semiseparable order. These results only apply to positive definite systems of one-level SSS type. Our results in this paper indicate that this also holds for the indefinite and multilevel SSS systems.

Acknowledgements

The authors would like to thank the three anonymous reviewers that help to improve the quality of the paper.

This research is supported by the NWO Veni Grant #11930 'Reconfigurable Floating Wind Farms'.

References

1. Chung T. *Computational Fluid Dynamics* 2nd ed. Cambridge University Press: Cambridge, 2010.
2. Benzi M, Olshanskii M, Wang Z. Modified augmented Lagrangian preconditioners for the incompressible Navier-Stokes equations. *International Journal for Numerical Methods in Fluids* 2011; **66**(4):486–508.
3. Qiu Y, van Gijzen MB, van Wingerden JW, Verhaegen M. A class of efficient preconditioners with multilevel sequentially semiseparable matrix structure. *AIP Conference Proceedings* 2013; **1558**(1):2253–2256.
4. Golub G, Van Loan C. *Matrix Computations*. Johns Hopkins University Press: Baltimore, 1996.
5. Saad Y. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics: Philadelphia, 2003.
6. Sonneveld P, van Gijzen MB. IDR(s): a family of simple and fast algorithms for solving large nonsymmetric systems of linear equations. *SIAM Journal on Scientific Computing* 2008; **31**(2):1035–1062.
7. Zhang J. Preconditioned Krylov subspace methods for solving nonsymmetric matrices from CFD applications. *Computer Methods in Applied Mechanics and Engineering* 2000; **189**(3):825–840.
8. Silvester D, Elman H, Ramage A. *Incompressible Flow and Iterative Solver Software (IFISS) version 3.2*, 2012. (Available from: http://www.manchester.ac.uk/ifiss/) [Accessed on June 2012].
9. Dewilde P, Van der Veen AJ. *Time-varying Systems and Computations*. Kluwer Academic Publisher: Boston, 1998.
10. Rice J, Verhaegen M. Distributed control in multiple dimensions: a structure preserving computational technique. *IEEE Transactions on Automatic Control* 2011; **56**(3):516–530.
11. Rice J. Efficient algorithms for distributed control: a structured matrix approach, *Ph.D. Thesis*, 2010.
12. Gondzio J, Zhlobich P. Multilevel quasiseparable matrices in PDE-constrained optimization. *arXiv preprint arXiv:1112.6018* 2011:1–20.
13. Dewilde P, Jiao H, Chandrasekaran S. Model reduction in symbolically semi-separable systems with application to preconditioners for 3D sparse systems of equations. In *Characteristic Functions, Scattering Functions and Transfer Functions, Operator Theory: Advances and Applications*, Vol. 197. Birkhäser Basel: Negev, Israel, 2010; 99–132.
14. Qiu Y, van Gijzen MB, van Wingerden JW, Verhaegen M. On the application of a novel model order reduction algorithm for sequentially semi-separable matrices to the identification of one-dimensional distributed systems. *In Control Conference (ECC), 2014 European*, Strassburg, France, June 2014, pages 2750–2755.
15. Chandrasekaran S, Dewilde P, Gu M, Lyons W, Pals T. A fast solver for HSS representations via sparse matrices. *SIAM Journal on Matrix Analysis and Applications* 2006; **29**(1):67–81.
16. Sheng Z, Dewilde P, Chandrasekaran S. Algorithms to solve hierarchically semi-separable systems. In *Operator Theory: Advances and Applications*, Vol. 176, Alpay D, Vinnikov V (eds). Birkhäuser Basel: Negev, Israel, 2007; 255–294.
17. Hackbusch W. A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I: Introduction to $\mathcal{H}$-matrices. *Computing* 1999; **62**(2):89–108.
18. Le Borne S, Grasedyck L. $\mathcal{H}$-matrix preconditioners in convection-dominated problems. *SIAM Journal on Matrix Analysis and Applications* 2006; **27**(4):1172–1183.
19. Börm S, Le Borne S. $\mathcal{H}$-LU factorization in preconditioners for augmented Lagrangian and grad-div stabilized saddle point systems. *International Journal for Numerical Methods in Fluids* 2012; **68**(1):83–98.
20. Hackbusch W, Börm S. Data-sparse approximation by adaptive $\mathcal{H}^2$-matrices. *Computing* 2002; **69**(1):1–35.
21. Börm S. $\mathcal{H}^2$-matrices–multilevel methods for the approximation of integral operators. *Computing and Visualization in Science* 2004; **7**(3-4):173–181.
22. Xia J, Chandrasekaran S, Gu M, Li X. Superfast multifrontal method for large structured linear systems of equations. *SIAM Journal on Matrix Analysis and Applications* 2010; **31**(3):1382–1411.

23. Xia J. A robust inner-outer hierarchically semi-separable preconditioner. *Numerical Linear Algebra with Applications* 2012; **19**(6):992–1016.
24. Xia J, Gu M. Robust approximate Cholesky factorization of rank-structured symmetric positive definite matrices. *SIAM Journal on Matrix Analysis and Applications* 2010; **31**(5):2899–2920.
25. Bebendorf M. Why finite element discretizations can be factored by triangular hierarchical matrices. *SIAM Journal on Numerical Analysis* 2007; **45**(4):1472–1494.
26. Bebendorf M. *Hierarchical matrices*, A Means to Efficiently Solve Elliptic Boundary Value Problems, vol. 63. Springer: Berlin Heidelberg, 2008.
27. Xia J. Efficient structured multifrontal factorization for general large sparse matrices. *SIAM Journal on Scientific Computing* 2013; **35**(2):832–860.
28. Qiu Y, van Gijzen MB, van Wingerden JW, Verhaegen M, Vuik C, *Evaluation of multilevel sequentially semiseparable preconditioners on CFD benchmark problems using IFISS. Technical Report 13-11*, Delft Institution of Applied Mathematics, Delft University of Technology, 2013. (Available from: http://ta.twi.tudelft.nl/nw/users/yueqiu/publications.html) [Accessed on January 2014].
29. Vandebril R, Van Barel M, Mastronardi N. *Matrix Computations and Semiseparable Matrices: Linear Systems*. Johns Hopkins University Press: Baltimore, 2007.
30. Gonzales R, Eisert J, Koltracht I, Neumann M, Rawitscher G. Integral equation method for the continuous spectrum radial Schrodinger equation. *Journal of Computational Physics* 1997; **134**(1):134–149.
31. Kavcic A, Moura J. Matrices with banded inverses: inversion algorithms and factorization of Gauss-Markov processes. *IEEE Transactions on Information Theory* 2000; **46**(4):1495–1509.
32. Greengard L, Rokhlin V. On the numerical solution of two-point boundary value problems. *Communications on Pure and Applied Mathematics* 1991; **44**(4):419–452.
33. Van Barel M, Fasino D, Gemignani L, Mastronardi N. Orthogonal rational functions and diagonal-plus-semiseparable matrices. *International Symposium on Optical Science and Technology*, Seattle, USA, 2002, 162–170.
34. Chandrasekaran S, Dewilde P, Gu M, Pals T, Sun X, van der Veen A, White D. Some fast algorithms for sequentially semiseparable representations. *SIAM Journal on Matrix Analysis and Applications* 2005; **27**(2):341–364.
35. Eidelman Y, Gohberg I. On a new class of structured matrices. *Integral Equations and Operator Theory* 1999; **34**(3):293–324.
36. Eidelman Y, Gohberg I, Olshevsky V. The QR iteration method for hermitian quasiseparable matrices of an arbitrary order. *Linear Algebra and Its Applications* 2005; **404**(15):305–324.
37. Eidelman Y, Gohberg I. A modification of the Dewilde-van der Veen method for inversion of finite structured matrices. *Linear Algebra and Its Applications* 2002; **343-344**(1):419–450.
38. Chandrasekaran S, Dewilde P, Gu M, Pals T, van der Veen AJ, *Fast stable solvers for sequentially semi-separable linear systems of equations. Technical Report*, Lawrence Livermore National Laboratory, 2003.
39. Qiu Y, van Gijzen MB, van Wingerden JW, Verhaegen M, Vuik C, *Efficient preconditioners for PDE-constrained optimization problems with a multi-level sequentially semi-separable matrix structure. Technical Report 13-04*, Delft Institution of Applied Mathematics, Delft University of Technology, 2013. (Available from: http://ta.twi.tudelft.nl/nw/users/yueqiu/publications.html) [Accessed on December 2013].
40. Eidelman Y, Gohberg I. Inversion formulas and linear complexity algorithm for diagonal plus semiseparable matrices. *Computers & Mathematics with Applications* 1997; **33**(4):69–79.
41. Chandrasekaran S, Dewilde P, Gu M, Somasunderam N. On the numerical rank of the off-diagonal blocks of Schur complements of discretized elliptic PDEs. *SIAM Journal on Matrix Analysis and Applications* 2010; **31**(5):2261–2290.
42. Eidelman Y, Gohberg I. On generators of quasiseparable finite block matrices. *Calcolo* 2005; **42**(3):187–214.
43. van Gijzen MB, Sonneveld P. Algorithm 913: an elegant IDR(s) variant that efficiently exploits biorthogonality properties. *ACM Transactions on Mathematical Software* 2011; **38**(1):5:1–5:19.
44. Elman H, Silvester D, Wathen A. *Finite Elements and Fast Iterative Solvers: With Applications in Incompressible Fluid Dynamics*. Oxford University Press: New York, 2005.
45. Benzi M, Golub G, Liesen J. Numerical solution of saddle point problems. *Acta Numerica* 2005; **14**:1–137.
46. Rees T, Wathen A. Preconditioning Iterative Methods for the Optimal Control of the Stokes Equations. *SIAM Journal on Scientific Computing* 2011; **33**(5):2903–2926.
47. Wathen A, Silvester D. Fast iterative solution of stabilised Stokes systems. Part I: using simple diagonal preconditioners. *SIAM Journal on Numerical Analysis* 1993; **30**(3):630–649.
48. Silvester D, Wathen A. Fast iterative solution of stabilised Stokes systems. Part II: using general block preconditioners. *SIAM Journal on Numerical Analysis* 1994; **31**(5):1352–1367.
49. Paige C, Saunders M. Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis* 1975; **12**(4):617–629.
50. Olshanskii M, Vassilevski Y. Pressure Schur complement preconditioners for the discrete Oseen problem. *SIAM Journal on Scientific Computing* 2007; **29**(6):2686–2704.
51. Benzi M, Olshanskii M. An augmented Lagrangian-based approach to the Oseen problem. *SIAM Journal on Scientific Computing* 2006; **28**(6): 2095–2113.
52. Napov A. Conditioning analysis of incomplete Cholesky factorizations with orthogonal dropping. *SIAM Journal on Matrix Analysis and Applications* 2013; **34**(3):1148–1173.