DELFT UNIVERSITY OF TECHNOLOGY

REPORT 13-11

Evaluation of Multilevel Sequentially Semiseparable Preconditioners on CFD Benchmark Problems Using IFISS

Yue Qiu, Martin B. van Gijzen, Jan-Willem van Wingerden
Michel Verhaegen, and Cornelis Vuik

ISSN 1389-6520

Reports of the Delft Institute of Applied Mathematics

Delft 2013

Copyright © 2013 by Delft Institute of Applied Mathematics, Delft, The Netherlands.

No part of the Journal may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission from Delft Institute of Applied Mathematics, Delft University of Technology, The Netherlands.

Evaluation of Multilevel Sequentially Semiseparable Preconditioners on CFD Benchmark Problems Using IFISS*

Yue Qiu †, Martin B. van Gijzen ‡, Jan-Willem van Wingerden *,
Michel Verhaegen *, and Cornelis Vuik†

Revised on: July 20th, 2014

Abstract

This paper studies a new preconditioning technique for sparse systems arising from discretized partial differential equations (PDEs) in computational fluid dynamics (CFD), which exploit 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 PDEs in linear computational complexity with respect to the problem size. In contrast to the standard block preconditioners, we exploit the global MSSS structure of the 2 by 2 block system from the discretized Stokes equation and linearized Navier-Stokes equation. This avoids the approximation of the Schur complement, which is a big advantage over standard block preconditioners. Numerical experiments on standard CFD benchmark problems in IFISS were carried out to evaluate the performance of the MSSS preconditioners. It was illustrated that the MSSS preconditioner yields mesh size independence convergence. Moreover, the convergence is almost insensitive to the viscosity parameter. Comparison with the algebraic multigrid (AMG) method and the geometric multigrid (GMG) method, the MSSS preconditioning technique is more robust than both the AMG method and the GMG method, and considerably faster than the AMG method.

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 [2] [3]. Normally, the system matrix A is large and sparse. Many efforts have been dedicated to finding efficient

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

†Delft Center for System and Control, Delft University of Technology, 2628 CD Delft, the Netherlands, Y.Qiu@tudelft.nl, YueCiou@Gmail.com.

‡Delft Institute of Applied Mathematics, Delft University of Technology, 2628 CD Delft, the Netherlands, M.B.vanGijzen@tudelft.nl.

[2]This work is an extension of the paper [1] that has been presented in the 11th International Conference of Numerical Analysis and Applied Mathematics, Rhodes, Greece, 2013.

solution methods for such systems. There are two approaches in general, direct solution methods and iterative solution methods.

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 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] [5] [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 [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], and 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]. Multilevel sequentially semiseparable (MSSS) matrices generalize the sequentially semiseparable matrices to the multi-dimensional case. Discretization of higher dimensional (2D or 3D) PDEs on structured grid yields matrices with an MSSS structure [12] [13]. MSSS preconditioners have been previously studied in [1] [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 (PCG) method by MSSS matrix computations. These work just studied the symmetric positive problems from discretized scalar PDEs and did not deal with block systems arising from discretized coupled PDEs which are quite common in CFD problems. Meanwhile, these papers did not give comparison of the performance for the MSSS preconditioners with the other methods. In [1], MSSS preconditioners were extended to solve non-symmetric convection-diffusion equations.

Several other related structured matrices have been proposed in literature. This includes hierarchical semiseparable (HSS) matrices [14] [15], hierarchical ($\mathcal{H}$-) matrices [16] [17] [18], and $\mathcal{H}^2$-matrices [19] [20]. HSS matrix computations are usually applied in the multifrontal solver [21]. Some recent efforts devoted to preconditioning of symmetric positive definite systems can be found in [22] [23]. 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 [24] [25], Bebendorf extended $\mathcal{H}$-matrices to solving elliptic PDEs. Preconditioning techniques based on $\mathcal{H}$-matrix computations for CFD problems were studied in [17] [18]. In [17], an $\mathcal{H}-LU$ preconditioner was proposed to solve the convection-diffusion equation, while in [18] the augmented Lagrangian (AL) preconditioner based on $\mathcal{H}$-matrix computations was introduced to solve the discrete Oseen problems. It was shown that HSS matrices and $\mathcal{H}$-matrices can be used to represent the discretized PDEs on unstructured grid [24] [26]. For MSSS matrices, this is less natural. Although MSSS matrices do not give a direct representation of discretized PDEs on unstructured grid, there exist some relations between HSS matrices, $\mathcal{H}$-matrices and MSSS matrices to illustrate the possibility. It was shown in [15] that the HSS matrices and MSSS matrices can be transferred

from one to the other, which makes it possible to use some advanced concept from HSS matrices to represent discretized PDEs with unstructured grid by MSSS matrices. The advantage of MSSS matrix computations is its 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.

To keep this paper self-contained, we focused on MSSS matrix computations and assume that the grid to discretize PDEs is uniform. In this paper, we consider MSSS preconditioning techniques for CFD problems. For the discretized convection-diffusion equation using the finite element method, we exploit the MSSS structure of the system matrix. 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 to a single MSSS matrix. With the permutation, the discrete Stokes equation and discrete linearized Navier-Stokes equation could be put in the framework of MSSS matrix computations without computing an good approximation of the Schur complement for the standard block preconditioners, which is the key for standard preconditioning techniques and normally it is extremely expensive and difficult. This enables us to solve the CFD problems with preconditioned Krylov subspace methods in linear computational complexity. We evaluate the performance of the MSSS preconditioning technique on CFD benchmark problems in IFISS and compare with the algebraic multigrid (AMG) method and the geometric multugrid (GMG) method. Numerical experiments illustrate that the MSSS preconditioning technique yields mesh size independence convergence and eliminates or eliminates the convergence dependency on the viscosity parameter, which is a big advantage over the AMG and GMG methods. In addition to robustness, it was shown that the MSSS preconditioning technique is much 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 performed 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.

2 Multilevel Sequentially Semiseparable Matrices

Semiseparable matrices, whose sub-matrices taken from the lower-triangular or upper-triangular part are of rank 1, which is introduced in [27], appear in several types of applications, such as integral equations [28], Gauss-Markov processes [29], boundary value problems [30], and rational interpolation [31]. To generalize the semiseparable matrices, quasiseparable matrices, of which all sub-matrices extracted from the strictly lower-triangular or the strictly upper-triangular part, are of rank 1 [27]. 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) [32]. The property of low-rank off-diagonal blocks for SSS matrices is also investigated by Eidelman et. al. in [33], 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 [34] [35], LU/LDU [12] [33] can also be computed in a structure preserving way, which yields certain factors for such matrices with SSS structure. Moreover, all the operations mentioned above on SSS matrices can be performed in linear computational complexity. Besides, the memory consumption of SSS matrices also scale linearly with the problem size [12] [36].

To keep this paper self-contained, we review some definitions and concepts for SSS matrices, see also [1] [37]. 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 sequentially semiseparable (SSS) matrices are defined by Definition 2.1.

Definition 2.1. *[32]. 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 superscript ' T ' denotes the transpose of the matrix. The above 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 1 and they are called generators of the SSS matrix A.

Table 1: Generator size for the SSS matrix A in Definition 2.1

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

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. To keep a clear structure of this paper, Table 2 lists references that discuss how a given operation can be performed using SSS matrix arithmetic.

Table 2: [37] Commonly used operations on SSS matrices

Ax	$A \pm B$	AB	A^{-1}	LU	Model Reduction	$Lx = b^\star$
[32, 33, 36]	[32, 33, 36]	[32, 33, 36]	[9, 35, 38]	[12, 27, 37]	[36, 37]	[36]

* L is a lower-triangular SSS matrix.

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. *[37] 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.*

Within this multilevel framework, generators to represent an MSSS matrix of a higher hierarchy, are themselves MSSS matrices of a lower hierarchy. The 1-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 2-D Laplace equation with homogeneous Dirichlet boundary conditions, discretized using Q_1 finite element method, the system matrix is*

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

The matrix K is an MSSS (2-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

$$\begin{aligned} A &= \mathcal{SSS}(1,\ 0,\ -\frac{2}{3},\ \frac{8}{3},\ 1,\ 0,\ -\frac{2}{3}), \\ B &= \mathcal{SSS}(1,\ 0,\ -\frac{2}{3},\ -\frac{2}{3},\ 1,\ 0,\ -\frac{2}{3}). \end{aligned}$$

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 1 are satisfied for the Definition 2.1.*

Operations listed in Table 2 for the SSS matrices can be extended to the MSSS matrices, which can be also performed in linear computational complexity. The LU factorization can also be performed in a structure preserving way. Use the MSSS matrices in Example 2.1 for example, to perform the block LU factorization of the stiffness matrix K, the Schur complements are computed via the following recurrences,

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

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 the blocks A and B. It has been shown that A and B are SSS matrices, therefore, S_i is also an SSS matrix. If S_i has low off-diagonal rank, then S_i can be approximated accurately enough by an SSS matrix with low semiseparable order that will be introduced in the next section, in linear computational complexity.

In [39], it was shown that the off-diagonal blocks of the Schur complement 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 appear in the block LU factorization by SSS matrices with low semiseparable order. This block factorization can be performed in linear computational complexity. Due to the approximation of the Schur complements for performing the block LU factorization, the factorization computed is an approximate factorization, which can be used as a preconditioner. The details for this preconditioning technique 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 the 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. *[40] Let*

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

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

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

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

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

Definition 3.3. *[37] 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*

$$rank\ A(k+1:N, 1:k) = l_k,\ k = 1, 2, \cdots, N-1.$$

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

$$rank\ A(1:k, k+1:N) = u_k,\ k = 1, 2, \cdots, N-1.$$

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

Definition 3.4. *[37] 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 above, we have the following algorithm to compute the LU factorization of a k-level SSS matrix.

Lemma 3.1. *[27][12] 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 $A = \mathcal{MSSS}(P_s,\ R_s,\ Q_s,\ D_s,\ U_s,\ W_s,\ V_s)$. Let $A = LU$ be its block LU factorization. Then,*

1. *The factor L is a k-level sequentially semiseparable matrix of k-level block $(r^L,\ 0)$ semiseparable and 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*

$$\begin{aligned} L &= \mathcal{MSSS}(P_s,\ R_s,\ \hat{Q}_s,\ D_s^L,\ 0,\ 0,\ 0), \\ U &= \mathcal{MSSS}(0,\ 0,\ 0,\ D_s^U,\ \hat{U}_s,\ W_s,\ V_s). \end{aligned}$$

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

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

Initialize: $M_1 \leftarrow 0 \in \mathbb{R}^{r^l \times r^u}$ be a $(k-1)$-level SSS matrix
Compute the LU factorization of the $(k-1)$-level SSS matrix
$D_1 = D_1^L D_1^U$, 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**
$M_i = \hat{Q}_{i-1}^T \hat{U}_{i-1} + R_i M_{i-1} W_i$,
Compute the LU factorization of the $(k-1)$-level SSS matrix
$(D_i - P_i M_i V_i) = D_i^L D_i^U$,
Let, $\hat{U}_i = (D_i^L)^{-1}(U_i - P_i M_{i-1} W_i)$, $\hat{Q}_i = (D_i^U)^{-T}(Q_i - V_i M_{i-1}^T R_i^T)$.
end for
Compute the LU factorization of the $(k-1)$-level SSS matrix
$\left(D_n - P_n M_{n-1} V_n^T\right) = D_n^L D_n^U$
Output: D_i^L, D_i^U, $\hat{Q}_i$, $\hat{U}_i$

As explained in [37], 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 the growth of the semiseparable order of the $(k-1)$-level SSS generators, which induces growth of the computational complexity. Model order reduction algorithm is therefore necessary to reduce the semiseparable order or keep the semiseparable order under a threshold during the LU factorization. For details of the growth of the semiseparable order and the model order reduction, please refer to [37].

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

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}, \tag{5}$$

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 by 2 block system (5) is not an MSSS system but has MSSS blocks. Using Lemma 3.2 in [37], we can permute a matrix with MSSS blocks to a single MSSS matrix. This enables us to efficiently compute the LU factorization of a permuted saddle-point system with Algorithm 1 by MSSS matrix computations.

Lemma 3.2. *[11] [37] Let A, B, C and D be SSS matrices with the generators representations*

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

Then the relations

$$\begin{bmatrix} f \\ g \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}, \text{ and } \overline{\begin{bmatrix} f \\ g \end{bmatrix}} = \mathcal{T} \overline{\begin{bmatrix} a \\ b \end{bmatrix}}$$

are equivalent with row and column permutations of the matrix blocks. The vectors $\overline{\begin{bmatrix} f \\ g \end{bmatrix}}$ and $\overline{\begin{bmatrix} a \\ b \end{bmatrix}}$ are permutations of $\begin{bmatrix} f \\ g \end{bmatrix}$ and $\begin{bmatrix} a \\ b \end{bmatrix}$, respectively. The matrix $\mathcal{T}$ is an SSS matrix and has the generators representation

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

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

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 of $6n \times 6n$ for example, where

$$A = \begin{bmatrix} D_1^a & U_1^a V_2^{aT} & U_1^a W_2^a V_3^{aT} \\ P_2^a Q_1^{aT} & D_2^a & U_3^a V_3^{aT} \\ P_3^a R_2^a Q_1^{aT} & P_3^a Q_2^{aT} & D_3^a \end{bmatrix}, \quad B = \begin{bmatrix} D_1^b & U_1^b V_2^{bT} & U_1^b W_2^b V_3^{bT} \\ P_2^b Q_1^{bT} & D_2^b & U_3^b V_3^{bT} \\ P_3^b R_2^b Q_1^{bT} & P_3^b Q_2^{bT} & D_3^b \end{bmatrix}$$

$$C = \begin{bmatrix} D_1^c & U_1^c V_2^{cT} & U_1^c W_2^c V_3^{cT} \\ P_2^c Q_1^{cT} & D_2^c & U_3^c V_3^{cT} \\ P_3^c R_2^c Q_1^{cT} & P_3^c Q_2^{cT} & D_3^c \end{bmatrix}, \quad D = \begin{bmatrix} D_1^d & U_1^d V_2^{dT} & U_1^d W_2^d V_3^{dT} \\ P_2^d Q_1^{dT} & D_2^d & U_3^d V_3^{dT} \\ P_3^d R_2^d Q_1^{dT} & P_3^d Q_2^{dT} & D_3^d \end{bmatrix}.$$

The vectors $f = \begin{bmatrix} f_1 & f_2 & f_3 \end{bmatrix}^T$, $g = \begin{bmatrix} g_1 & g_2 & g_3 \end{bmatrix}^T$, $a = \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix}^T$, and $b = \begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix}^T$. Then there exists a permutation matrix Π of the following form

$$\Pi = \begin{bmatrix} \begin{bmatrix} I_n \\ 0 \end{bmatrix} \otimes I_3 & \begin{bmatrix} 0 \\ I_n \end{bmatrix} \otimes I_3 \end{bmatrix},$$

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

After the permutation, the permuted vector becomes

$$\overline{\begin{bmatrix} f \\ g \end{bmatrix}} = \Pi \begin{bmatrix} f \\ g \end{bmatrix} = \begin{bmatrix} f_1 & g_1 & f_2 & g_2 & f_3 & g_3 \end{bmatrix}^T,$$
$$\overline{\begin{bmatrix} a \\ b \end{bmatrix}} = \Pi \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} a_1 & b_1 & a_2 & b_2 & a_3 & b_3 \end{bmatrix}^T.$$

and the permuted block matrix is

$$\mathcal{T} = \overline{\begin{bmatrix} A & B \\ C & D \end{bmatrix}} = \Pi \begin{bmatrix} A & B \\ C & D \end{bmatrix} \Pi^T = \begin{bmatrix} D_1^t & U_1^t V_2^{tT} & U_1^t W_2^t V_3^{tT} \\ P_2^t Q_1^{tT} & D_2^t & U_3^t V_3^{tT} \\ P_3^t R_2^t Q_1^{tT} & P_3^t Q_2^{tT} & D_3^t \end{bmatrix},$$

where the generators for SSS matrix $\mathcal{T}$ have the following relation hold,

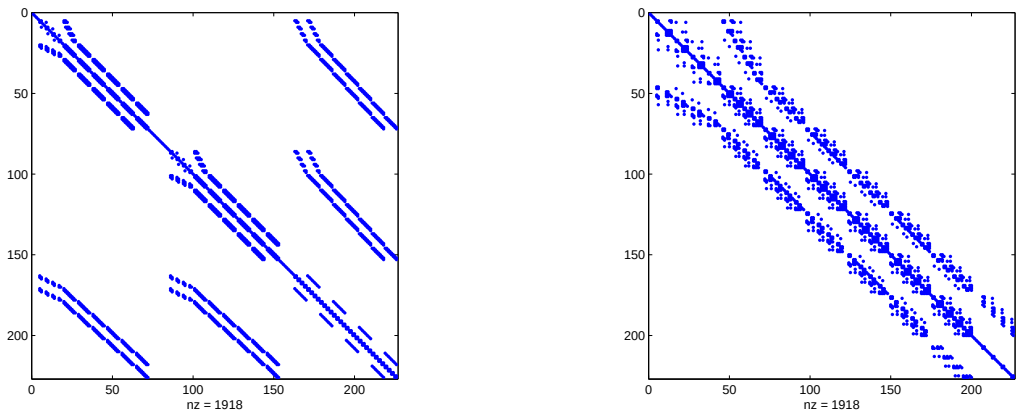
$$P_s^t = \begin{bmatrix} P_s^a & P_s^b & 0 & 0 \\ 0 & 0 & P_s^c & P_s^d \end{bmatrix}, \quad R_s^t = \begin{bmatrix} R_s^a & & & \\ & R_s^b & & \\ & & R_s^c & \\ & & & R_s^d \end{bmatrix}, \quad Q_s^t = \begin{bmatrix} Q_s^a & 0 & Q_s^c & 0 \\ 0 & Q_s^b & 0 & Q_s^d \end{bmatrix}, \quad U_s^t = \begin{bmatrix}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

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

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.

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 are permuted from the lowest level (1-level) to the highest level ($(k-1)$-level) by Lemma 3.2.*

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



(a) Before permutation

(b) After permutation

Figure 1: Structure of the saddle-point system of Stokes equation before and after permutation for $h = 2^{-2}$

4 Numerical Experiments

In this section, we test the performance of MSSS preconditioning techniques on CFD benchmark problems using IFISS. The convection-diffusion problem, Stokes problem and Navier-Stokes problem are considered. The algebraic multigrid (AMG) method and geometric multigrid (GMG) method in IFISS are also performed to compare their performance with that of the MSSS preconditioners. The MSSS matrix computations are performed using the so-called "Multilevel Sequentially Semiseparable Matrix Computations Toolbox" [41] under MATLAB. All the numerical experiments are implemented 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 in this paper, is reached. The MSSS matrix computation toolbox [41] and the test code for this paper can be found at `http://ta.twi.tudelft.nl/nw/users/yueqiu/software.html`.

In the tables that show results of the numerical experiments, to illustrate the linear computational complexity, we use the "preconditioning" column that reports the time to compute the approximate LU factorization for MSSS preconditioners or the time to setup the multigrid for the AMG and GMG method. The iterative solution method is chosen as the induction dimension reduction (IDR(s)) method [6] [42]. The time to solve the preconditioned system is reported in the "IDR(4)" column. The time spent in total is the sum of the time to compute the approximate the approximate LU factorization or the time to setup the multigrid for the AMG and GMG method and the time to solve the preconditioned system, which is reported in the "total" column in all the tables. All the columns concerning time in this paper are measured in seconds.

4.1 Convection-Diffusion Problem

Consider the example 3.1.4 in [43] for the convection-diffusion problem described in Example 4.1. The details of the discretization of the convection-diffusion equation is also introduced in [43]. To show the performance

of the MSSS preconditioning technique, we first test the diffusion dominance case that corresponds to the viscosity parameter $\epsilon = \frac{1}{200}$ and then test the convection dominance 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. *[43] Zero source term, recirculating wind, characteristic boundary layers.*

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

where $\Omega = \{(x,y)| -1 \le x \le 1, -1 \le y \le 1\}$, $\overrightarrow{\omega} = \big(2y(1-x^2),\ -2x(1-y^2)\big)$, $f = 0$. *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 with the MSSS preconditioner and the AMG and GMG method are listed in Table 3-5. The maximum semiseparable 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 for the mesh size $h = 2^{-7}$ is shown in Figure 2.

Table 3: MSSS 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

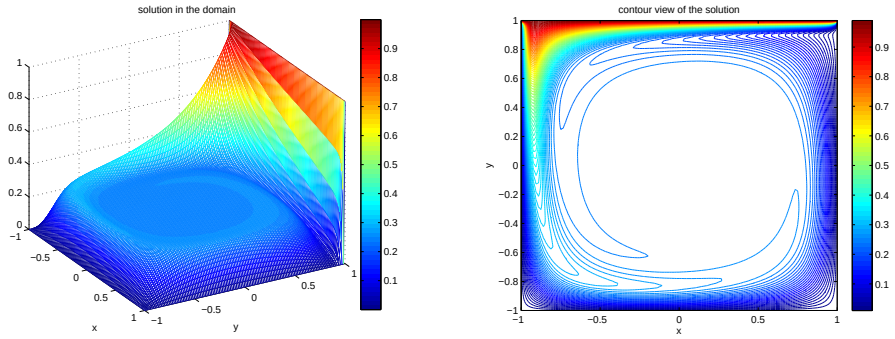
Table 4: AMG method for $\epsilon = \frac{1}{200}$

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

Table 5: GMG 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

As can be seen from Table 3, the MSSS preconditioner gives mesh size independence convergence for the convection-diffusion problem. Both the time to compute the approximate LU factorization with MSSS matrix computations and the time to solve the preconditioned system scale linearly with the problem size. Compared with the computational results shown in Table 4-5 for the AMG and GMG method, it is clear to see that the computational time of the AMG method setup is much bigger than the MSSS preconditioner, while the time to setup of the GMG method is much smaller than the MSSS preconditioner. Both the AMG and GMG methods give mesh size independence convergence. Table 4 illustrates that the computational complexity for the AMG method setup become bigger with the increase of the problem size and is bigger than linear. This is mostly due to the MATLAB is not competitive in speed and the MATLAB code in IFISS is not highly optimized.



(a) Solution of u.

(b) Contour view.

Figure 2: Solution of test problem 4.1 for $\epsilon = \frac{1}{200}$ and $h = 2^{-7}$

Next, we test the convection dominance case with the viscosity parameter $\epsilon = 10^{-4}$ for the MSSS preconditioner, the AMG and GMG method. The computational results are listed in Table 6-8. The solution for the mesh size $h = 2^{-7}$ with mesh size $h = 2^{-7}$ is shown in Figure 3.

Table 6: MSSS 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

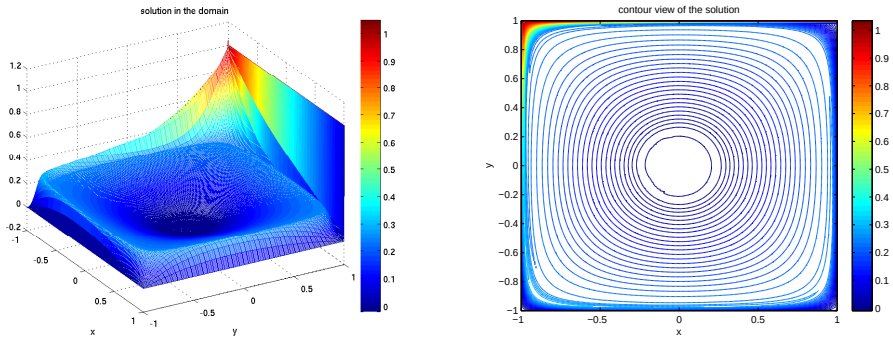
Table 7: AMG method with $\epsilon = 10^{-4}$

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	-

Table 8: GMG 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	-

This test case is the convection dominance problem, the system is ill-conditioned. It is more difficult to compute a good enough preconditioner to approximate the ill-conditioned system. Thus, bigger semiseparable order is needed to compute an accurate enough approximation compared with the case for $\epsilon = \frac{1}{200}$. This is illustrated by comparing the semiseparable order in Table 3 and Table 6. Due to the bigger semiseparable order, more computational time is needed. Even the time to compute the preconditioner and to solve the preconditioned system is bigger than the time for bigger ϵ. The computational time still scales linearly with the problem size. Due to the ill-condition of the problem to solve, the AMG and GMG method fail to solve the convention dominance system.



(a) Solution of u.

(b) Contour view.

Figure 3: Solution of test problem 4.1 for $\epsilon = 10^{-4}$ and $h = 2^{-7}$.

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

4.2 Stokes Problem

In this part, we evaluate the performance of the MSSS preconditioning technique for the lid-driven cavity problem of the Stokes equation described by Example 4.2, which is example 5.1.3 in [43]. Mixed finite element method discretization is also introduced in [43].

Example 4.2. *[43] Lid-driven cavity problem, enclosed flow boundary condition.*

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

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

$$\begin{bmatrix} K & B^T \\ B & -S_t \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}, \tag{6}$$

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 of the Stokes problem, n_u is the number of velocity grid points, and n_p is the number of pressure grid points.

The standard preconditioning technique for the saddle point system (6) is the block diagonal preconditioner $\mathcal{P}_1$ or the block lower-triangular preconditioner $\mathcal{P}_2$ that 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 generalized minimal residual (`GMRES`) [4] delivers to 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. `GMRES` delivers to the exact solution in at most two steps. We refer to [44] for an extensive study for such block preconditioners.

In general, the Schur complement S is difficult to compute due to the high computational complexity, or difficulty to compute a good approximation. It was shown in [45] that the Schur complements of the Stokes equation has the equivalent spectrum with the pressure mass matrix M_p, i.e., the relation

$$\gamma^2 \leq \frac{x^T BK^{-1}B^T x}{x^T M_p x} \leq \Gamma^2, \quad \forall x \in \mathbb{R}^{n_p} \backslash \{\mathbf{0}\} \tag{7}$$

hold, 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

$$\mathcal{P}_1 = \begin{bmatrix} K & \\ & M_p \end{bmatrix}, \quad \mathcal{P}_2 = \begin{bmatrix} K & \\ B & -M_p \end{bmatrix}. \tag{8}$$

This type of preconditioners are called the Silvester-Wathen preconditioner and are widely studies for the Stokes problems in [43] [45] [46].

In this paper, we first study the block diagonal preconditioners of the Silvester-Wathen preconditioner type. We choose the block diagonal preconditioner as

$$P = \begin{bmatrix} \hat{K} & \\ & \hat{M}_p \end{bmatrix} \tag{9}$$

where $\hat{K}$ is the approximation of K by the MSSS preconditioning technique, $\hat{M}_p$ is the lumped pressure mass matrix M_p. For comparison, the AMG and GMG method are also performed to approximate K for the block diagonal preconditioner. Due to the symmetric definiteness of the block diagonal preconditioner and the symmetry of the saddle point system, minimal residual (MINRES) method is chosen as the iterative solver [47]. The computational results are shown in Table 9-Table 11. The smoother of the AMG method is chosen as the point damped Jacobi.

Table 9: Silvester-Wathen preconditioner for the Stokes equation by MSSS 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

Table 10: Silvester-Wathen preconditioner for the Stokes equation by AMG 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

Table 11: Silvester-Wathen preconditioner for the Stokes equation by GMG method

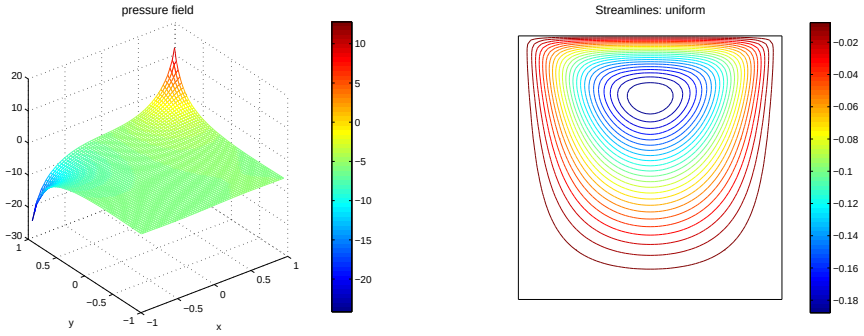
mesh size	problem size	No. iter.	preconditioning (sec.)	MINRES (sec.)	total (sec.)
2^{-4}	3.20e+03	34	0.09	0.09	0.18
2^{-5}	1.25e+04	34	0.14	0.28	0.42
2^{-6}	4.97e+04	32	0.61	0.58	1.19
2^{-7}	1.98e+05	32	2.01	2.00	4.01
2^{-8}	7.88e+05	30	3.26	7.38	10.64

As can be seen from Table 9-Table 11, the block diagonal preconditioner by the MSSS preconditioning techniques gives mesh size independence convergence. The time to compute the MSSS preconditioner and to solve the preconditioned system scales linearly with the problem size. Compared with the results of the block diagonal preconditioner by the AMG and GMG method, the number of iterations are almost the same for the MSSS block diagonal preconditioner. The MSSS preconditioning technique is faster than the AMG method.

It is not difficult to verify that for the discrete Stokes system (6), all the matrix blocks K, B and S_t are MSSS matrices. Thus we can apply Lemma 3.2 to permute the saddle-point system (6) with MSSS blocks to a single MSSS system. With the permuted MSSS system, the LU factorization for the MSSS matrices in Algorithm 3.1 can be performed to compute an efficient preconditioner. We call this LU factorization for the permuted system the global preconditioner. Due to the indefiniteness of the global preconditioner, the iterative solution method was chosen as IDR(s). The computational results for the discrete Stokes equation by the global preconditioner are listed in Table 12. The solution of the pressure field and the streamline are shown in Figure 4.

Table 12: Global preconditioner for the permuted Stokes equation

mesh size	problem size	iterations	preconditioning	IDR(4)	total
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



(a) Pressure distribution. (b) Streamline.

Figure 4: Solution of test example 4.2 for MSSS preconditioners

Compared with the results for the block diagonal preconditioners, the number of iterations and computational time is much reduced for global preconditioner. For middle and large size problems, the time for the global preconditioner is much less than the time for the AMG method.

Remark 4.2. *Since the global preconditioner does not need to compute an approximation of the Schur complement like the standard block preconditioners, the computational effort is much reduced. This is a big advantage compared with the standard block preconditioners.*

4.3 Navier-Stokes Problem

In this paper, we test the lid-driven cavity example 7.1.3 of the Navier-Stokes equation in [43] to evaluate the performance of the MSSS preconditioning technique.

Example 4.3. *[43] Lid-driven cavity problem, enclosed flow boundary condition.*

$$-\nu \bigtriangledown^2 \overrightarrow{u} + \overrightarrow{u} \cdot \bigtriangledown \overrightarrow{u} + \bigtriangledown p = \overrightarrow{f} \tag{10}$$

$$\bigtriangledown \cdot \overrightarrow{u} = 0 \tag{11}$$

with the boundary conditioners

$$\overrightarrow{u} = \overrightarrow{\omega} \text{ on } \Gamma_D \tag{12}$$

$$\nu \frac{\partial \overrightarrow{u}}{\partial n} - \overrightarrow{n} p = \overrightarrow{0} \text{ on } \Gamma_N \tag{13}$$

in a square domain $\{(x,y)| -1 \leq x \leq 1, -1 \leq y \leq 1\}$*, where the regularized cavity condition* $\{y = 1; -1 \leq x \leq 1 | u_x = 1 - x^4\}$ *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. Details about the finite element discretization and linearization are described in [43]. In this paper, we use the $Q_1 - P_0$ finite element method to discretize and the Newton

method to linearize. At each linearized step, we need to solve a 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 current velocity in the x and y directions, B_x and B_y are divergence matrices in the x and y directions, and S_t is a stabilization matrix of $Q_1 - P_0$ type. Due to the difficulty to compute a good enough approximation of the Schur complement, preconditioning of the Navier-Stokes equation is still a big challenge and hot topic. Some efforts to compute efficient approximation of the Schur complement for the block preconditioners can be found in [43] [48] [49].

Note that all the blocks in (14) have MSSS structure, it is easy to apply Lemma 3.2 to permute the block system (14) to a single MSSS system. This will give the global preconditioner as discussed in Section 4.2. To test the performance of the global preconditioner, we solve the system (14) for the second Newton step. The computational results for the global preconditioner are shown in Table 13. The pressure convection-diffusion (PCD) preconditioner [48] in IFISS is chosen for comparison. In the pressure convection-diffusion preconditioner, the Schur complement for the standard block preconditioners is approximate by

$$\hat{S}^{-1} = \hat{M}_p^{-1} A_p L_p^{-1}. \tag{15}$$

Here $\hat{M}_p^{-1}$ denotes an approximation solver of the pressure mass matrix system, A_p and L_p are approximation of the convection-diffusion operator and Laplace operator in the finite dimensional solution space of the pressure with some prescribed boundary conditions. For details of the pressure convection-diffusion preconditioner, please refer to [43] [48].

Since the GMG method is not implemented in IFISS, only the results of the PCD preconditioner computed by the AMG method are reported in Table 14. Due to much longer time for the AMG setup, the computational results for the case of $h = 2^{-8}$ is not listed in Table 14. The viscosity parameter ν is set to be 10^{-2} for both numerical experiments.

Table 13: MSSS preconditioner for the 2nd Newton Step

mesh size	problem size	iterations	preconditioning	IDR(4)	total
$2^{-4}(6)$	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

Table 14: Pressure convection-diffusion preconditioner by AMG method for the 2nd Newton Step

mesh size	problem size	iterations	preconditioning	IDR(4)	total
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

Computational results for the global preconditioner illustrate that the global preconditioner gives mesh size independent convergence. Moreover, the global preconditioner reduces the computational time significantly compared with the AMG method. The computational results verify the linear computational complexity of the MSSS preconditioning technique. For all the test problems of the Navier-Stokes problem in this paper, the MSSS preconditioning technique is much faster than the AMG method.

5 Conclusions

In this paper, we have studied a new class of preconditioners based on multilevel sequentially semiseparable (MSSS) matrix computations for computational fluid dynamics (CFD) problems. To evaluate its performance, the algebraic multigrid (AMG) method in IFISS is also performed on the CFD benchmark problems. Numerical experiments for the convection-diffusion equation, the Stokes equation and the Navier-Stokes equation show that the MSSS preconditioners is mesh size independent. Compared with the AMG method, the required number of iterations is significantly reduced. Moreover, the MSSS preconditioning technique is more robust and faster than the AMG method.

The mesh independent convergence of the MSSS preconditioner is an open problem and is the ongoing research of the authors. Some recent efforts to analyze the preconditioner can be found in [50]. In this reference, Napov explained that the accuracy of the incomplete Cholesky factorization depends only on the approximation accuracy of the off-diagonal blocks and gives the 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 the accuracy of the approximation of the off-diagonal blocks is directly related with the semiseparable order. The results in the paper only apply to positive definite systems of SSS type. Our results in this manuscript results indicate that it also holds for indefinite and multilevel SSS systems.

References

[1] Y. Qiu, M.B. van Gijzen, J.W. van Wingerden, and M. Verhaegen. A class of efficient preconditioners with multilevel sequentially semiseparable matrix structure. *AIP Conference Proceedings*, 1558(1):2253–2256, 2013.

[2] T.J. Chung. *Computational fluid dynamics*. Cambridge University Press, Cambridge, second edition, 2010.

[3] M. Benzi, M.A. Olshanskii, and Z. Wang. Modified augmented Lagrangian preconditioners for the incompressible Navier-Stokes equations. *International Journal for Numerical Methods in Fluids*, 66(4):486–508, 2011.

[4] G.H. Golub and C.F. Van Loan. *Matrix computations*. Johns Hopkins University Press, Baltimore, 1996.

[5] Y. Saad. *Iterative methods for sparse linear systems*. Society for Industrial and Applied Mathematics, Philadelphia, 2003.

[6] P. Sonneveld and M.B. van Gijzen. IDR(s): A family of simple and fast algorithms for solving large nonsymmetric systems of linear equations. *SIAM Journal on Scientific Computing*, 31(2):1035–1062, 2008.

[7] J. Zhang. Preconditioned Krylov subspace methods for solving nonsymmetric matrices from CFD applications. *Computer Methods in Applied Mechanics and Engineering*, 189(3):825–840, 2000.

[8] D. Silvester, H. Elman, and A. Ramage. Incompressible Flow and Iterative Solver Software (IFISS) version 3.2, May 2012. `http://www.manchester.ac.uk/ifiss/`.

[9] P. Dewilde and A.J. Van der Veen. *Time-varying systems and computations*. Kluwer Academic Publisher, Boston, 1998.

[10] J.K. Rice and M. Verhaegen. Distributed control in multiple dimensions: A structure preserving computational technique. *IEEE Transactions on Automatic Control*, 56(3):516–530, 2011.

[11] J.K. Rice. *Efficient Algorithms for Distributed Control: a Structured Matrix Approach*. PhD thesis, Delft University of Technology, 2010.

[12] J. Gondzio and P. Zhlobich. Multilevel quasiseparable matrices in PDE-constrained optimization. *arXiv preprint arXiv:1112.6018*, pages 1–20, 2011.

[13] P. Dewilde, H.Y. Jiao, and S. Chandrasekaran. 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*, volume 197 of *Operator Theory: Advances and Applications*, pages 99–132. Birkhäser Basel, 2010.

[14] Shiv Chandrasekaran, Patrick Dewilde, Ming Gu, W Lyons, and T Pals. A fast solver for HSS representations via sparse matrices. *SIAM Journal on Matrix Analysis and Applications*, 29(1):67–81, 2006.

[15] Zhifeng Sheng, Patrick Dewilde, and Shivkumar Chandrasekaran. Algorithms to solve hierarchically semi-separable systems. In Daniel Alpay and Victor Vinnikov, editors, *Operator Theory: Advances and Applications*, volume 176, pages 255–294. Birkhäuser Basel, 2007.

[16] Wolfgang Hackbusch. A sparse matrix arithmetic based on $\mathcal{H}$-matrices. part i: Introduction to $\mathcal{H}$-matrices. *Computing*, 62(2):89–108, 1999.

[17] S. Le Borne and L. Grasedyck. $\mathcal{H}$-matrix preconditioners in convection-dominated problems. *SIAM Journal on Matrix Analysis and Applications*, 27(4):1172–1183, 2006.

[18] Steffen Börm and Sabine Le Borne. $\mathcal{H}$-LU factorization in preconditioners for augmented Lagrangian and grad-div stabilized saddle point systems. *International Journal for Numerical Methods in Fluids*, 68(1):83–98, 2012.

[19] Wolfgang Hackbusch and Steffen Börm. Data-sparse approximation by adaptive $\mathcal{H}^2$-matrices. *Computing*, 69(1):1–35, 2002.

[20] Steffen Börm. $\mathcal{H}^2$-matrices–multilevel methods for the approximation of integral operators. *Computing and Visualization in Science*, 7(3-4):173–181, 2004.

[21] J. Xia, S. Chandrasekaran, M. Gu, and X. Li. Superfast multifrontal method for large structured linear systems of equations. *SIAM Journal on Matrix Analysis and Applications*, 31(3):1382–1411, 2010.

[22] Jianlin Xia. A robust inner-outer hierarchically semi-separable preconditioner. *Numerical Linear Algebra with Applications*, 19(6):992–1016, 2012.

[23] Jianlin Xia and Ming Gu. Robust approximate Cholesky factorization of rank-structured symmetric positive definite matrices. *SIAM Journal on Matrix Analysis and Applications*, 31(5):2899–2920, 2010.

[24] M. Bebendorf. Why finite element discretizations can be factored by triangular hierarchical matrices. *SIAM Journal on Numerical Analysis*, 45(4):1472–1494, 2007.

[25] Mario Bebendorf. Hierarchical matrices. In *Lecture Notes in Computational Science and Engineering*, volume 63. Springer Berlin Heidelberg, 2008.

[26] Jianlin Xia. Efficient structured multifrontal factorization for general large sparse matrices. *SIAM Journal on Scientific Computing*, 35(2):832–860, 2013.

[27] R. Vandebril, M. Van Barel, and N. Mastronardi. *Matrix computations and semiseparable matrices: linear systems*. Johns Hopkins University Press, Baltimore, 2007.

[28] R.A. Gonzales, J. Eisert, I. Koltracht, M. Neumann, and G. Rawitscher. Integral equation method for the continuous spectrum radial Schrodinger equation. *Journal of Computational Physics*, 134(1):134–149, 1997.

[29] A. Kavcic and J.M.F. Moura. Matrices with banded inverses: inversion algorithms and factorization of Gauss-Markov processes. *IEEE Transactions on Information Theory*, 46(4):1495–1509, 2000.

[30] L. Greengard and V. Rokhlin. On the numerical solution of two-point boundary value problems. *Communications on Pure and Applied Mathematics*, 44(4):419–452, 1991.

[31] M. Van Barel, D. Fasino, L. Gemignani, and N. Mastronardi. Orthogonal rational functions and diagonal-plus-semiseparable matrices. In *International Symposium on Optical Science and Technology*, pages 162–170, 2002.

[32] S. Chandrasekaran, P. Dewilde, M. Gu, T. Pals, X. Sun, A.J. van der Veen, and D. White. Some fast algorithms for sequentially semiseparable representations. *SIAM Journal on Matrix Analysis and Applications*, 27(2):341–364, 2005.

[33] Y. Eidelman and I. Gohberg. On a new class of structured matrices. *Integral Equations and Operator Theory*, 34(3):293–324, 1999.

[34] Y. Eidelman, I. Gohberg, and V. Olshevsky. The QR iteration method for hermitian quasiseparable matrices of an arbitrary order. *Linear Algebra and Its Applications*, 404(15):305–324, 2005.

[35] Y. Eidelman and I. Gohberg. A modification of the Dewilde-van der Veen method for inversion of finite structured matrices. *Linear Algebra and Its Applications*, 343-344(1):419–450, 2002.

[36] S. Chandrasekaran, P. Dewilde, M. Gu, T. Pals, and A.J. van der Veen. Fast stable solvers for sequentially semi-separable linear systems of equations. Technical report, Lawrence Livermore National Laboratory, 2003.

[37] Y. Qiu, M.B. van Gijzen, J.W. van Wingerden, M. Verhaegen, and C. Vuik. 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. `http://ta.twi.tudelft.nl/nw/users/vuik/pub13.html`.

[38] Y. Eidelman and I. Gohberg. Inversion formulas and linear complexity algorithm for diagonal plus semiseparable matrices. *Computers & Mathematics with Applications*, 33(4):69–79, 1997.

[39] S. Chandrasekaran, P. Dewilde, M. Gu, and N. Somasunderam. On the numerical rank of the off-diagonal blocks of Schur complements of discretized elliptic PDEs. *SIAM Journal on Matrix Analysis and Applications*, 31(5):2261–2290, 2010.

[40] Y. Eidelman and I. Gohberg. On generators of quasiseparable finite block matrices. *Calcolo*, 42(3):187–214, 2005.

[41] Yue Qiu. Multilevel Sequentially Semiseparable (MSSS) Matrix Computation Toolbox version 0.7, 2013. `http://ta.twi.tudelft.nl/nw/users/yueqiu/software.html`.

[42] Martin B. Van Gijzen and Peter Sonneveld. Algorithm 913: An elegant IDR(s) variant that efficiently exploits biorthogonality properties. *ACM Transactions on Mathematical Software*, 38(1):5:1–5:19, 2011.

[43] H.C. Elman, D.J. Silvester, and A.J. Wathen. *Finite Elements and Fast Iterative Solvers: with Applications in Incompressible Fluid Dynamics*. Oxford University Press, New York, 2005.

[44] M. Benzi, G.H. Golub, and J. Liesen. Numerical solution of saddle point problems. *Acta Numerica*, 14:1–137, 2005.

[45] A. Wathen and D. Silvester. Fast iterative solution of stabilised Stokes systems. Part I: Using simple diagonal preconditioners. *SIAM Journal on Numerical Analysis*, 30(3):630–649, 1993.

[46] D. Silvester and A. Wathen. Fast iterative solution of stabilised Stokes systems. Part II: Using general block preconditioners. *SIAM Journal on Numerical Analysis*, 31(5):1352–1367, 1994.

[47] C. Paige and M. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 12(4):617–629, 1975.

[48] M. Olshanskii and Y. Vassilevski. Pressure schur complement preconditioners for the discrete Oseen problem. *SIAM Journal on Scientific Computing*, 29(6):2686–2704, 2007.

[49] M. Benzi and M. Olshanskii. An augmented Lagrangian-based approach to the Oseen problem. *SIAM Journal on Scientific Computing*, 28(6):2095–2113, 2006.

[50] Artem Napov. Conditioning analysis of incomplete Cholesky factorizations with orthogonal dropping. *SIAM Journal on Matrix Analysis and Applications*, 34(3):1148–1173, 2013.

Appendix

A Convection-diffusion Problem

Consider the following three convection-diffusion test problems described in Chapter 3 of [43].

Example A.1. *Zero source term, constant vertical wind, exponential boundary layer.*

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

where $\Omega = [-1,\ 1]^2$, $\overrightarrow{\omega} = (0,\ 1)$, $f = 0$ *and with Direchlet boundary conditions as* $u(x,\ -1) = x$, $u(x,\ 1) = 0$, $u(-1,\ y) \approx -1$, $u(1,\ y) \approx 1$, *where the later two approximations hold except near* $y = 1$.

Example A.2. *Zero source term, variable vertical wind, characteristic boundary layers.*

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

where $\Omega = [-1,\ 1]^2$, $\overrightarrow{\omega} = (0,\ 1+(x+1)^2/4)$, $f = 0$. *Dirichlet boundary values apply on the inflow boundary and characteristic segments.* u *is set to unity on the inflow boundary, and decreases to zero quadratically on the right wall, and cubically on the left wall. Zero Neumann boundary condition apply on* $x = 1$.

Example A.3. *Zero source term, constant wind speed at a* 30 *degree angle to the left of vertical, downstream boundary layer and interior.*

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

where $\Omega = [-1,\ 1]^2$, $\overrightarrow{\omega} = (-\sin\frac{\pi}{6},\ \cos\frac{\pi}{6})$, $f = 0$. *Dirichlet boundary conditions are imposed everywhere on* $\partial\Omega$ *with values either zero or unity with a jump discontinuity at the point* $(0,\ -1)$. *The inflow boundary is* $[x,\ -1] \cup [1,\ y]$.

A.1 Computational Results of Test Example A.1

The computational results of Example A.1 are in Table 15-18. The smoother of AMG is set to be the pointed damped Jacobi.

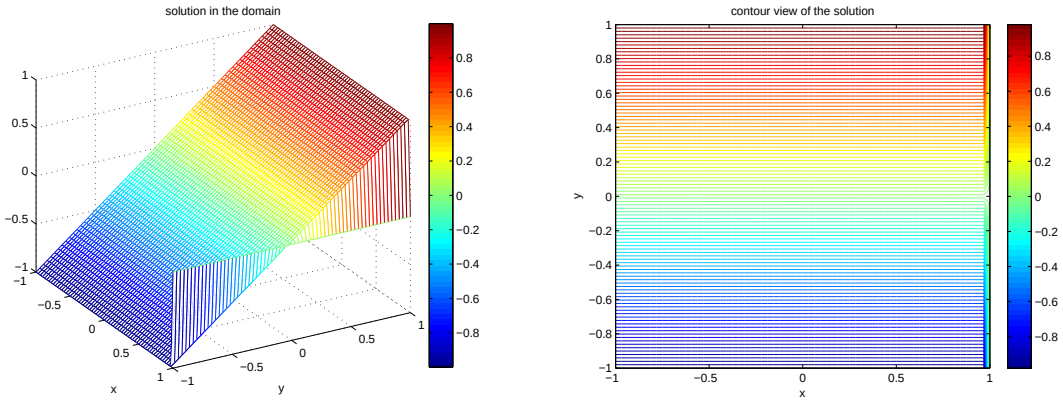
Table 15: MSSS Preconditioner with $\epsilon = \frac{1}{200}$

mesh size	problem size	iterations	preconditioning time	IDR(4) time	total time
$2^{-4}(4)$	1.09e+03	1	0.44	0.06	0.50
$2^{-5}(4)$	4.23e+04	1	1.07	0.17	1.24
$2^{-6}(4)$	1.66e+04	1	3.47	0.53	4.00
$2^{-7}(4)$	6.60e+04	1	13.11	1.92	15.03
$2^{-8}(4)$	2.63e+05	1	49.11	7.27	56.38

Table 16: AMG method with $\epsilon = \frac{1}{200}$

mesh size	problem size	iterations	preconditioning time	IDR(4) time	total time
2^{-4}	1.09e+03	9	0.51	0.24	0.76
2^{-5}	4.23e+04	5	1.87	0.06	1.93
2^{-6}	1.66e+04	8	12.53	0.14	12.67
2^{-7}	6.60e+04	7	112.61	0.35	112.96

The solution for the mesh size $h = 2^{-5}$ by MSSS preconditioners are shown in Figure 5.



(a) Solution of u.

(b) Contour view.

Figure 5: Solution of test Example A.1 for $\epsilon = \frac{1}{200}$ with the MSSS preconditioner

Now, we decrease the viscosity ϵ to 10^{-4}, the computational results for MSSS preconditioning and AMG method are shown in Table 17-18.

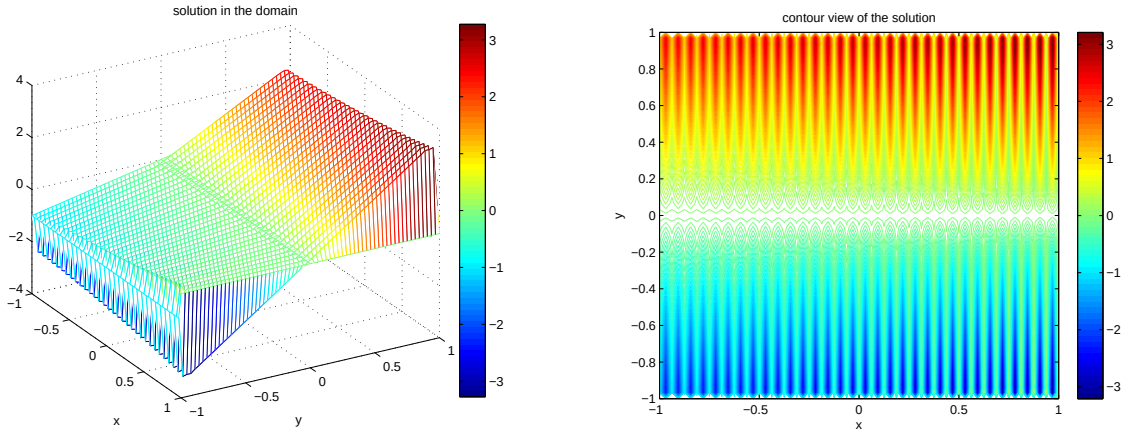
Table 17: MSSS Preconditioner with $\epsilon = 10^{-4}$

mesh size	problem size	iterations	preconditioning time	IDR(4) time	total time
$2^{-4}(4)$	1.09e+03	1	0.44	0.13	0.57
$2^{-5}(4)$	4.23e+04	1	1.00	0.17	1.17
$2^{-6}(4)$	1.66e+04	1	3.58	0.62	4.20
$2^{-7}(4)$	6.60e+04	1	13.71	2.04	15.75
$2^{-9}(4)$	2.63e+05	1	48.96	7.43	56.39

Table 18: AMG method with $\epsilon = 10^{-4}$

mesh size	problem size	iterations	preconditioning time	IDR(4) time	total time
2^{-4}	1.09e+03	19	0.34	0.08	0.42
2^{-5}	4.23e+04	100	2.30	no convergence	-
2^{-6}	1.66e+04	100	14.39	no convergence	-
2^{-7}	6.60e+04	100	110.24	no convergence	-

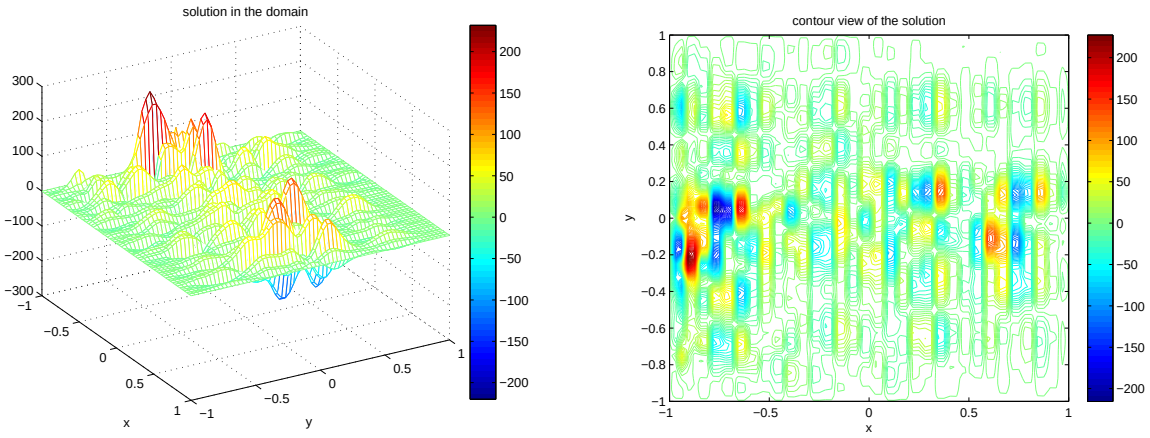
The solution for the mesh size $h = 2^{-5}$ by MSSS preconditioners are shown in Figure 6. The results by AMG method are shown in Figure 7. The relative residual in the IDR(4) iterations by AMG method is shown in Figure 8.



(a) Solution of u.

(b) Contour view.

Figure 6: Solution of test Example A.1 for $\epsilon = 10^{-4}$ with the MSSS preconditioner



(a) Solution of u.

(b) Contour view.

Figure 7: Solution of test Example A.1 for $\epsilon = 10^{-4}$ with AMG method

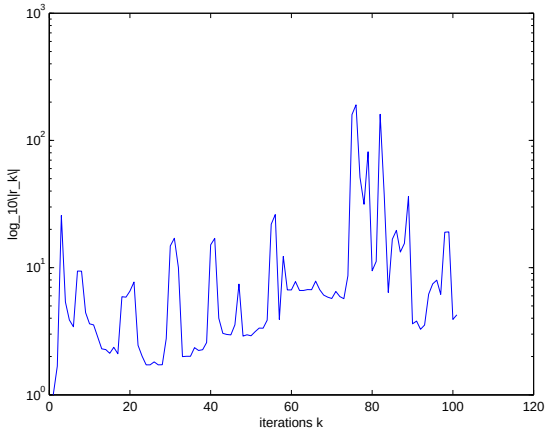


Figure 8: Relative residual by AMG method for $\epsilon = 10^{-4}$

For the smoother chosen as ILU. The computational results are the same with point damped Jacobi smoother for $\epsilon = 10^{-4}$.

A.2 Test Example A.2

For $\epsilon = \frac{1}{200}$, the computational results are shown in Figure 19-20. The smoother of AMG method was selected as ILU.

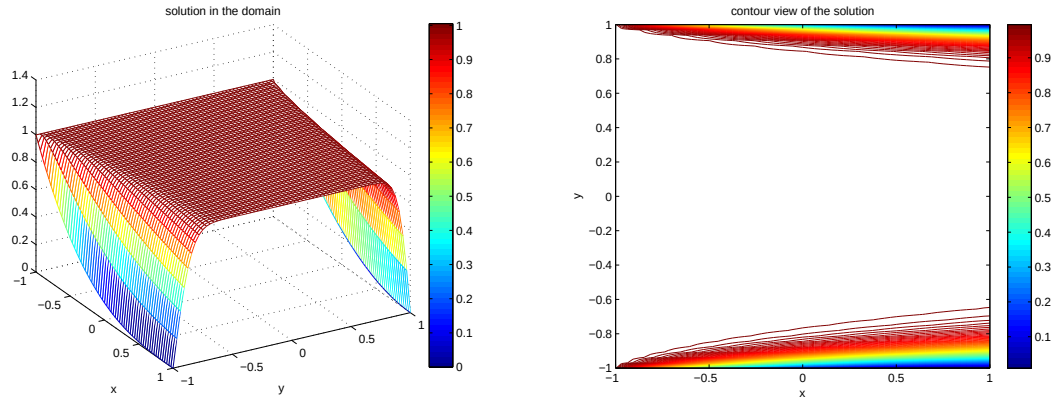
Table 19: MSSS Preconditioner with $\epsilon = \frac{1}{200}$

mesh size	problem size	iterations	preconditioning time	IDR(4) time	total time
$2^{-4}(4)$	1.09e+03	1	0.43	0.12	0.55
$2^{-5}(4)$	4.23e+04	1	1.22	0.23	1.45
$2^{-6}(4)$	1.66e+04	1	3.91	0.77	4.68
$2^{-7}(4)$	6.60e+04	1	14.56	2.07	16.63
$2^{-9}(4)$	2.63e+05	1	51.64	7.06	58.70

Table 20: AMG method with $\epsilon = \frac{1}{200}$

mesh size	problem size	iterations	preconditioning time	IDR(4) time	total time
2^{-4}	1.09e+03	8	0.44	0.03	0.47
2^{-5}	4.23e+04	3	2.32	0.05	2.37
2^{-6}	1.66e+04	2	12.13	0.06	12.19
2^{-7}	6.60e+04	3	114.09	0.17	114.26

The solution by MSSS preconditioners for $h = 2^{-5}$ is shown in figure 9.



(a) Solution of u.

(b) Contour view.

Figure 9: Solution of test example 4.2 for $\epsilon = \frac{1}{200}$

Decrease ϵ to 10^{-4}. The the computational results are shown in table 21-22.

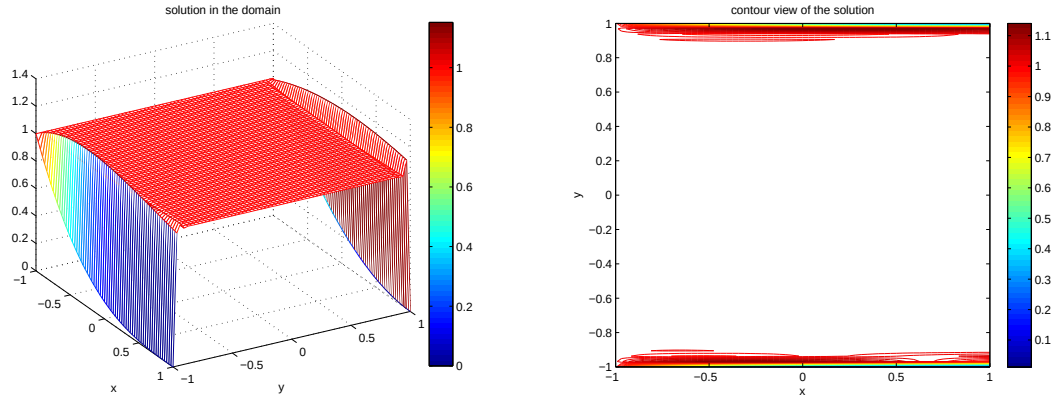
Table 21: MSSS Preconditioner with $\epsilon = 10^{-4}$

mesh size	problem size	iterations	preconditioning time	IDR(4) time	total time
$2^{-4}(4)$	1.09e+03	1	0.42	0.13	0.55
$2^{-5}(4)$	4.23e+04	1	1.27	0.33	1.60
$2^{-6}(4)$	1.66e+04	1	3.84	0.61	4.45
$2^{-7}(4)$	6.60e+04	1	14.10	2.03	16.13
$2^{-9}(4)$	2.63e+05	1	51.40	7.58	58.98

Table 22: AMG method with $\epsilon = 10^{-4}$

mesh size	problem size	iterations	preconditioning time	IDR(4) time	total time
2^{-4}	1.09e+03	100	0.39	no convergence	-
2^{-5}	4.23e+04	-	-	-	-
2^{-6}	1.66e+04	-	-	-	-
2^{-7}	6.60e+04	-	-	-	-

The solution of MSSS preconditioners for $h = 2^{-5}$ and $\epsilon = 10^{-4}$ is shown in figure 10.



(a) Solution of u.

(b) Contour view.

Figure 10: Solution of test example 4.2 for $\epsilon = 10^{-4}$

A.3 Test Example A.3

For $\epsilon = \frac{1}{200}$, the computational results are shown in table 23-24. The smoother of AMG method was selected as ILU.

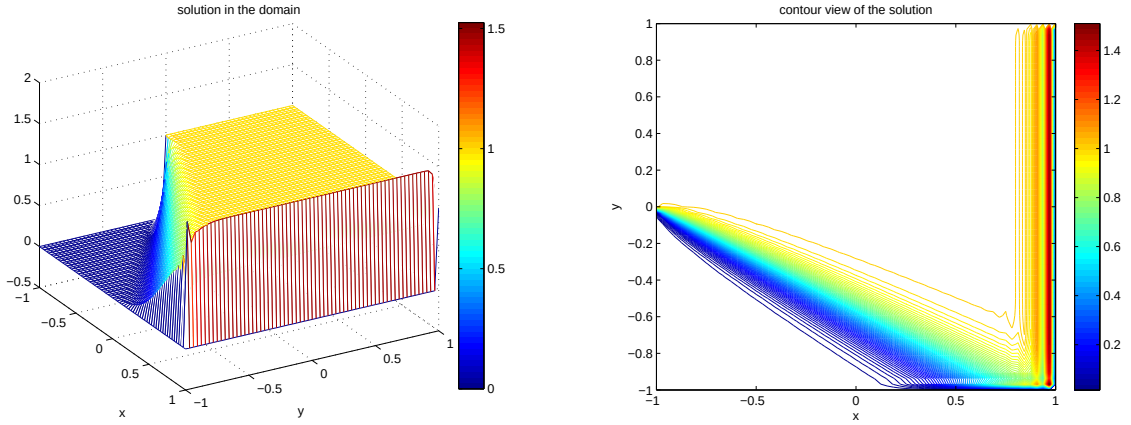
Table 23: MSSS Preconditioner with $\epsilon = \frac{1}{200}$

mesh size	problem size	iterations	preconditioning time	IDR(4) time	total time
$2^{-4}(4)$	1.09e+03	2	0.43	0.18	0.61
$2^{-5}(4)$	4.23e+04	1	1.21	0.23	1.44
$2^{-6}(4)$	1.66e+04	1	3.87	0.62	4.49
$2^{-7}(4)$	6.60e+04	1	13.56	2.06	15.62
$2^{-9}(4)$	2.63e+05	1	52.73	7.77	60.50

Table 24: AMG method with $\epsilon = \frac{1}{200}$

mesh size	problem size	iterations	preconditioning time	IDR(4) time	total time
2^{-4}	1.09e+03	8	0.38	0.06	0.44
2^{-5}	4.23e+04	5	1.64	0.06	1.70
2^{-6}	1.66e+04	3	12.06	0.08	12.14
2^{-7}	6.60e+04	5	119.22	0.33	119.55

The solution for MSSS preconditioners for $h = 2^{-5}$ with $\epsilon = \frac{1}{200}$ is shown in figure 11.



(a) Solution of u.

(b) Contour view.

Figure 11: Solution of test example A.3 for $\epsilon = \frac{1}{200}$

Decrease the ϵ to 10^{-4}, the computation results are shown in table 25-26.

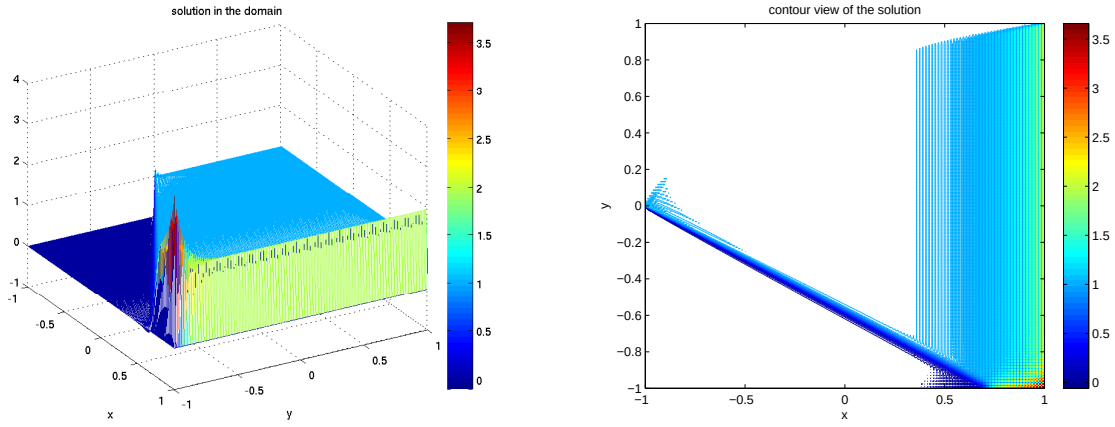
Table 25: MSSS Preconditioner with $\epsilon = 10^{-4}$

mesh size	problem size	iterations	preconditioning time	IDR(4) time	total time
$2^{-4}(13)$	1.09e+03	5	0.46	0.34	0.80
$2^{-5}(4)$	4.23e+04	6	1.41	1.04	2.45
$2^{-6}(8)$	1.66e+04	5	3.99	2.78	6.77
$2^{-7}(4)$	6.60e+04	5	14.02	9.82	23.84
$2^{-9}(4)$	2.63e+05	3	51.47	22.00	73.47

Table 26: AMG method with $\epsilon = 10^{-4}$

mesh size	problem size	iterations	preconditioning time	IDR(4) time	total time
2^{-4}	1.09e+03	100	0.44	no convergence	-
2^{-5}	4.23e+04	100	1.64	-	-
2^{-6}	1.66e+04	100	12.06	-	-
2^{-7}	6.60e+04	100	119.22	-	-

The solution for $\epsilon = 10^{-4}$ with MSSS preconditioner for $h = 2^{-7}$ is shown in figure 12.



(a) Solution of u.

(b) Contour view.

Figure 12: Solution of test example A.3 for $\epsilon = 10^{-4}$