Preconditioning Optimal Flow Control Problems Using Multilevel Sequentially Semiseparable Matrix Computations

邱越

图难于其易，为大于其细

–老子

He who wants to accomplish a big and difficult undertaking should start with easier things first and make sure that all details are attended to

– Lao-Tze (6th - 5th Century BC)

Preconditioning Optimal Flow Control Problems Using Multilevel Sequentially Semiseparable Matrix Computations

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus Prof. Ir. K. C. A. M. Luyben,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op,
maandag 7 december 2015 om 15:00 uur

door

Yue QIU

Master of Science in Control Theory and Control Engineering,
Northeastern University, China
geboren te Huainan, Anhui, China

This dissertation has been approved by the
promotors: Prof. Dr. Ir. M. Verhaegen, Prof. Dr. Ir. C. Vuik, and
copromotor: Dr. Ir. M. B. van Gijzen

Composition of the doctoral committee:

Rector Magnificus,	chairperson
Prof. Dr. Ir. M. Verhaegen,	Delft University of Technology, promotor
Prof. Dr. Ir. C. Vuik,	Delft University of Technology, promotor
Dr. Ir. M. B. van Gijzen,	Delft University of Technology, copromotor

Independent members:

Prof. Dr. A. Wathen,	Oxford University
Prof. Dr. P. Benner,	Max Planck Institute for Dynamics of Complex Technical Systems
Prof. Dr. Ir. B. Koren,	Eindhoven University of Technology
Prof. Dr. Ir. A.-J. van der Veen,	Delft University of Technology
Prof. Dr. Ir. A. Heemink,	Delft University of Technology, reserved

Other member:

Dr. Ir. J.-W. van Wingerden,	Delft University of Technology, supervisor

The work presented in this dissertation has been partly supported by Delft Institute of Applied Mathematics, Delft University of Technology.

ISBN: 978-94-6186-566-3

This dissertation was typeset using LaTeX 2_ε and edited in Vim under GNU/Linux 3.18 on x64. Printed by Proefschriftmaken.nl || Uitgeverij BOXPress, the Netherlands.

To my family

献给我的家人

海上生明月，天涯共此时

– 唐·张九龄

As the bright moon shines over the sea,
from far away you share this moment with me.

– Chang Chiu-ling (678 - 740)

Summary

Preconditioning Optimal Flow Control Problems Using Multilevel Sequentially Semiseparable Matrix Computations

Yue Qiu

Optimal flow control problems are important for applications in science and engineering. Solving such problems usually requires the solution of a large linear generalized saddle-point system. This linear system is sparse and highly indefinite. In order to solve such systems using Krylov subspace methods, efficient preconditioners are necessary to enhance their robustness and accelerate the convergence.

Standard block preconditioning techniques for the generalized saddle-point systems require an efficient approximation of the Schur complement. This is a big challenge since the Schur complement is large and dense, and therefore computationally expensive to approximate. For some problems, it is even impossible to approximate the Schur complement efficiently.

In this dissertation, we propose a new class of preconditioners for optimal flow control problems using multilevel sequentially semiseparable (MSSS) matrix computations. In contrast to standard block preconditioners, MSSS preconditioners do not approximate the Schur complement of the generalized saddle-point system but compute an approximate factorization of the global block system in linear computational complexity. This is a big advantage over block preconditioners. The key to this global factorization is that the Schur complements in this factorization usually have low off-diagonal rank. Therefore, these Schur complements can be approximated by matrices with a low rank off-diagonal structure. For this, MSSS matrix computations are very well suited.

Theoretical analysis shows that MSSS preconditioners yield a spectrum of the preconditioned system matrix that is contained in a circle centered at $(1, 0)$. This radius can be controlled arbitrarily by properly choosing a parameter in the MSSS preconditioner computations. This in turn implies that the convergence of MSSS preconditioned systems can be independent of the mesh size and regularization parameter for PDE-constrained optimization problems while for computational fluid dynamics problems, the convergence is independent of the mesh size and Reynolds number. Mesh size independent and wave number independent convergence can be

also obtained when applying the MSSS preconditioning technique to the Helmholtz problem. Numerical results verify the convergence property.

In this dissertation, we also studied the problem of optimal in-domain control of the Navier-Stokes equation. We use a simplified wind farm control example to formulate such a problem. Compared with standard PDE-constrained optimization problems where the controls are distributed throughout the whole domain, this in-domain control problem is even more difficult to solve since the control only acts on a few parts of the domain. This in turn gives a linear system of the generalized saddle-point type. Block preconditioners cannot give satisfactory performance for such problem because the Schur complement for such system is very difficult or even impossible to approximate efficiently. Applying MSSS preconditioners to this problem gives superior performance compared to block preconditioning techniques.

Samenvatting

Het Preconditioneren van Optimal Flow Control Problemen met Multilevel Sequentially Semiseparabel Matrix Berekeningen

Yue Qiu

Optimal flow control problemen zijn belangrijk in toepassingen in de wetenschap en in de industrie. De aanpak van dergelijke problemen vereist vaak het oplossen van grote algemene zadelpuntstelsels. Deze lineaire stelsels zijn ijl en sterk indefiniet. Om gebruik te kunnen maken van Krylov deelruimte methodes bij het oplossen van zulke stelsels zijn efficiente preconditioners vereist die de robuustheid vergroten en de convergentie versnellen.

Standaard blok-preconditioneringstechnieken voor algemene zadelpuntproblemen vereisen een efficiente benadering van het Schur complement. Deze benadering is een grote uitdaging, gezien het feit dat het Schur complement groot en vol is, en daardoor duur is om te benaderen. Voor sommige problemen is het zelfs onmogelijk om het Schur complement efficient te benaderen.

In dit proefschrift presenteren we een nieuwe klasse van preconditioneringen voor optimal flow control problemen welke gebruik maakt van multilevel sequential semiseparable (MSSS) matrix berekeningen. In tegenstelling tot standaard blok-preconditioneringen benaderen MSSS-preconditioneringen niet het Schur complement van het globale zadelpuntstelsel, maar in plaats daarvan berekenen zij een benaderende ontbinding van het globale blok-stelsel in lineaire rekencomplexiteit. Dit is een groot voordeel ten opzichte van blok-preconditioneringen. De sleutel tot deze globale ontbinding is het feit dat het Schur complement in deze ontbinding vaak een lage off-diagonal rang heeft. Daardoor kan dit Schur complement benaderd worden met matrices met een lage rang off-diagonal structuur. De MSSS matrix berekeningen zijn hiervoor uitermate geschikt.

Theoretische analyse toont aan dat MSSS-preconditionering een spectrum van het gepreconditioneerde stelsel geven dat in een cirkel rond het punt (1,0) ligt. De radius kan willekeurig gekozen worden door de selectie van een parameter in de berekening van de MSSS-preconditionering. Het gevolg is dat de convergentie van MSSS-gepreconditioneerde stelsels onafhankelijk is van de grid grootte en de regularizatie parameter in PDE-constrained optimalizatie problemen. In de vloeistof mechanica maakt dit de convergentie dan onafhankelijk van de maasgrootte en het getal van Reynolds. Convergentie onafhankelijk van maasgrootte en golfgetal kan

tot slot ook verkregen worden met MSSS preconditioning toegepast op Helmholtz problemen. Numerieke tests bevestigen de voorspelde convergentiesnelheid.

In dit proefschrift bestuderen we ook het probleem van optimal in-domain control van de Navier Stokes vergelijkingen. We gebruiken een versimpeld windparkmodel om het probleem te formuleren. Vergeleken met een standaard PDE-constrained optimalizatieprobleem, waar de controle is verdeeld over het volledige domein, is het in-domain probleem nog ingewikkelder omdat de controle op slechts enkele delen van het domein werkt. Dit resulteert in een algemeen lineair zadelpuntstelsel. Blok-preconditioneringen geven geen goed resultaat voor zulke problemen omdat het Schur complement van het stelsel niet efficient te benaderen is. Het toepassen van MSSS-preconditioneringen geeft een sterk verbeterd resultaat in vergelijking met blok-preconditioneringen.

Contents

1 Chapter

Background and State-of-the-Art

1.1 Motivation

Wind energy offers a possibility to reduce the green house gas emissions and the dependence on fossil energy. Offshore wind energy provides more power than land based because the wind is more steady, with less turbulence, and even larger wind turbines can be installed. Grouping the wind turbines in wind farm helps to reduce the sea-area usage and the impact to the environment. This grouping also cuts down the cost of maintenance and the connection cable for the power transmission [57]. The clustering of wind turbines in the wind farm makes upstream turbines interact with the downstream turbines. This aerodynamics interactions are caused by the wakes of turbines, cf. Figure 1.1.

Figure 1.1: Wakes effects for Horns Rev 1 offshore wind farm at Denmark. Source: Christian Steiness

Due to this wake interaction, a reduced velocity profile downstream is obtained. This in turn reduces the energy extraction by the wind turbines downstream because the extracted energy scales with the cubic power of the velocity at the location of turbines [76]. Wake effects make wind turbines downstream produce little or no energy and simply struggle to operate in intensive turbulence. To reduce the

wake effects, one has either to put turbines far away from each other or implement control strategies. The first approach gives a lay-out optimization problem [36]. Enlarging the distance between turbines will increase the cost for maintenance and investment. Therefore, a trade-off has to be made. Once the lay-out of turbines is fixed, the wake effect can be further reduced by coordinating control operations for all the turbines. This coordinate control aims at optimizing the total extracted power of a wind farm, which in turn gives a wind farm control problem. The wind farm control can be formulated as a PDE-constrained optimization problem, which will be introduced in the next section.

1.2 Background

The wind farm control introduced in the previous section can be formulated as a PDE-constrained optimization problem where the constraints are the partial differential equations (PDEs) that describe fluid dynamics problems. In this section, we will give a brief introduction of both PDE-constrained optimization problems and standard fluid dynamics problems.

1.2.1 PDE-Constrained Optimization

Most natural or physical processes are mathematically modeled by a system of linear/nonlinear partial differential equations (PDEs). Many science and engineering problems in control, design, and parameter optimization can be formulated as an optimization problem that is governed by PDEs [1, 59, 69]. The computational complexity of the PDE simulation often presents significant challenges. Due to recent advances in numerical linear algebra and the computational capability of modern computers, such simulations have become an essential tool for scientists and engineers in academia as well as industry, such as aerodynamics [69], optimal tomography [1], structural mechanics [7], optimal flow control [71], et al. to address this challenge.

One example of a PDE-constrained optimization problem is the temperature control in a bounded domain. Let $\Omega \in \mathbb{R}^d (d = 1, 2, 3)$ be a bounded set of mass, the heat distribution of this mass satisfies the following partial differential equation,

$$\mathcal{L}u = f, \tag{1.1}$$

subject to the following boundary condition

$$u = u_d \text{ on } \partial\Omega_D, \quad \text{and } \frac{\partial u}{\partial \overrightarrow{n}} = u_N \text{ on } \partial\Omega_N. \tag{1.2}$$

Here $\mathcal{L}$ is a linear differential operator and $\mathcal{L} = \dfrac{\partial}{\partial t} - \nabla^2$ for the heat equation, ∇ is the gradient operator, u is the heat distribution in the domain, t represents the time, and f is an outer source like microwave or electromagnetic induction of the mass. The boundary is denoted by $\partial\Omega$ and $\partial\Omega_D \cup \partial\Omega_N = \partial\Omega$, $\overrightarrow{n}$ is the unit normal vector that points outwards on the boundary. Suppose we have a desired

heat distribution profile $\hat{u}$ that prescribes the heat distribution in the domain that we expect. Take the heated rebar for example, it is heated by an outer source to make a proper temperature distribution for the shape control. This problem belongs to the typical tracking problem type, i.e., how to choose the input f such that u is as close to $\hat{u}$ as possible?

If we consider this problem under the functional space, say $L^2(\Omega)$, then we can formulate the following optimization problem,

$$\begin{aligned} \min_{u,f} \quad & \frac{1}{2} \|u - \hat{u}\|_{L^2(\Omega)} \\ s.t. \quad & \mathcal{L}u = f, \\ & u = u_d \text{ on } \partial\Omega_D, \\ & \frac{\partial u}{\partial \overrightarrow{n}} = u_N \text{ on } \partial\Omega_N. \end{aligned} \tag{1.3}$$

The optimization problem (1.3) is ill-posed [106]. To make this problem well-posed, we can add a Tychonoff regularization term [81, 106] that couples the cost function directly with the input f, which is given by

$$\begin{aligned} \min_{u,f} \quad & \frac{1}{2} \|u - \hat{u}\|_{L^2(\Omega)} + \beta \|f\|_{L^2(\Omega)} \\ s.t. \quad & \mathcal{L}u = f, \\ & u = u_d \text{ on } \partial\Omega_D, \\ & \frac{\partial u}{\partial \overrightarrow{n}} = u_N \text{ on } \partial\Omega_N. \end{aligned} \tag{1.4}$$

The regularization term $\beta \|f\|_{L^2(\Omega)}$ can be interpreted as a restriction on the input energy, while β corresponds to the weight of the restriction on the energy of the input.

The control input f and the state u that reaches the minimum of this optimization problem are called optimal input and optimal state, respectively. This type of problem is called distributed PDE control problem because that the control input f is distributed throughout the domain, cf. Chapter 2. Other types of optimal PDE control problems include the boundary control problem where the control only acts on the boundary or only the boundary condition can be adjusted [42, 70, 75], and in-domain control where the control input only acts on few parts of the domain, cf. [39, 84] and Chapter 5.

1.2.2 Computational Fluid Dynamics

The incompressible Navier-Stokes equations that describe the dynamics of Newtonian fluid [53] are given by

$$\begin{aligned} \frac{\partial \overrightarrow{u}}{\partial t} - \nu \nabla^2 \overrightarrow{u} + \overrightarrow{u} \cdot \nabla \overrightarrow{u} + \nabla p &= \overrightarrow{f}, \\ \nabla \cdot \overrightarrow{u} &= 0. \end{aligned} \tag{1.5}$$

Here ν is the kinematic viscosity of the fluid, $\overrightarrow{u}$ denotes the velocity of the fluid, p denotes the pressure, $\overrightarrow{f}$ is an outer source, and $\nabla\cdot$ represents the divergence operator.

We consider stationary flow in this dissertation, this corresponds to $\dfrac{\partial \overrightarrow{u}}{\partial t} = 0$. Then we have the steady-state Navier-Stokes equation,

$$\begin{aligned} -\nu\nabla^2\overrightarrow{u} + \overrightarrow{u}\cdot\nabla\overrightarrow{u} + \nabla p &= \overrightarrow{f}, \\ \nabla\cdot\overrightarrow{u} &= 0. \end{aligned} \tag{1.6}$$

The first equation in (1.6) is called the momentum equation which represents the conservation of the momentum for the fluid. The second equation in (1.6) is called the incompressibility condition and represents the conservation of the mass for the fluid. The incompressible Navier-Stokes equations complemented with proper boundary conditions form the most general model of incompressible viscous flow.

Note that the Navier-Stokes equation (1.6) is nonlinear. When the velocity of the flow is small or the flow is tightly confined, the nonlinear term can be neglected, which yields the Stokes equation

$$\begin{aligned} -\nu\nabla^2\overrightarrow{u} + \nabla p &= \overrightarrow{f}, \\ \nabla\cdot\overrightarrow{u} &= 0. \end{aligned} \tag{1.7}$$

Another equation that simplifies the momentum equation of the Navier-Stokes equation is the convection-diffusion equation. It is obtained by dropping the pressure gradient from the momentum equation of the Navier-Stokes equation, and taking a constant convection vector. This in turn gives

$$-\nu\nabla^2\overrightarrow{u} + \overrightarrow{w}\cdot\nabla\overrightarrow{u} = \overrightarrow{f}. \tag{1.8}$$

Here $\overrightarrow{w}$ is a constant vector. The vector equation (1.8) consists of few decoupled scalar PDEs. Therefore, the general scalar convection-diffusion equation is given by

$$-\nu\nabla^2 u + \overrightarrow{w}\cdot\nabla u = f. \tag{1.9}$$

Such equation describes the physical motion of many important processes, such as the motion of pollutant in a river or in air.

1.3 State-of-the-Art of Numerical Solution of Generalized Saddle-Point Systems

To numerically solve the PDE-constrained optimization problem, we need to discretize the cost function and the PDE constraints and then compute the optimality condition. To solve the fluid dynamics problem, we also need to discretize the PDEs

that model the fluid dynamics. After discretization, both the PDE-constrained optimization problem and computational fluid dynamics problem yield a linear system that is large, sparse and highly indefinite which is given by

$$\underbrace{\begin{bmatrix} A & B^T \\ B & -D \end{bmatrix}}_{\mathcal{A}} \underbrace{\begin{bmatrix} u \\ p \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} f \\ g \end{bmatrix}}_{b}. \tag{1.10}$$

For PDE-constrained optimization problems and the Stokes equation discretized by higher order mixed finite element method (Q_2–Q_1, Q_2–P_{-1} or Q_2–P_0), $A \in \mathbb{R}^{n\times n}$ is symmetric positive definite (SPD), $D = 0$ and $B \in \mathbb{R}^{m\times n}$ with $m \leq n$ has full column rank. This corresponds to a linear system of the saddle-point type. For the linearized Navier-Stokes equation after discretization, the eigenvalues of A have positive real parts, $B \in \mathbb{R}^{m\times n}$ with $m \leq n$ has full column rank, D is symmetric positive semi-definite that corresponds to a stabilization term for lower order (Q_1–P_0 or Q_1–Q_1) discretization, or $D = 0$ for higher order (Q_2–Q_1, Q_2–P_{-1} or Q_2–P_0) discretization. This yields a linear system of the generalized saddle-point type.

There are many research efforts devoted to numerically solving (1.10) efficiently. In general, there are two types of approaches, one is the direct solution methods, and the other type is the iterative solution methods. Among all these methods, Krylov subspace methods such as the conjugate gradient (`CG`) [60], minimal residual (`MINRES`) [92], generalized minimal residual (`GMRES`) and induced dimension reduction (`IDR(s)`) [119] methods, are the most favored.

Since the linear system (1.10) is large and highly indefinite, the robustness and efficiency of Krylov solvers need to be enhanced by applying so-called preconditioning techniques. Preconditioning techniques for the system (1.10) have been developed for many applications that arise from PDE-constrained optimization problems [59, 106], computational fluid dynamics problems [53, 118], parameter identification problems [26, 65], and have attracted many research efforts in the last decades [15, 48, 52, 53, 72, 86, 89, 96, 106, 118, 121, 130], cf. [15, 89] for a general survey.

The standard block preconditioners for (1.10) originate from the block factorization

$$\mathcal{A} = \begin{bmatrix} A & B^T \\ B & -D \end{bmatrix} = \begin{bmatrix} I & 0 \\ BA^{-1} & I \end{bmatrix} \begin{bmatrix} A & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} I & A^{-1}B^T \\ 0 & I \end{bmatrix}, \tag{1.11}$$

where $S = -D - BA^{-1}B^T$ is the Schur complement. The block-diagonal and block lower-triangular preconditioners can be chosen as

$$\mathcal{P}_d = \begin{bmatrix} A & \\ & -S \end{bmatrix}, \quad \mathcal{P}_t = \begin{bmatrix} A & \\ B & S \end{bmatrix}. \tag{1.12}$$

It is shown in [15] that

$$\lambda(\mathcal{P}_d^{-1}\mathcal{A}) = \left\{1,\ \frac{1 \pm \sqrt{5}}{2}\right\}, \quad \text{and} \quad \lambda(\mathcal{P}_t^{-1}\mathcal{A}) = \{1\},$$

and that the preconditioned matrix $\mathcal{P}_d^{-1}\mathcal{A}$ has minimum polynomial of degree 3, the preconditioned matrix $\mathcal{P}_t^{-1}\mathcal{A}$ has minimum polynomial of degree 2. Therefore, Krylov solvers such as `IDR(s)` and `GMRES` compute the solution at most three steps by using the block-diagonal preconditioner $\mathcal{P}_d$ and at most two steps by using the block lower-triangular preconditioner $\mathcal{P}_t$.

The Schur complement S in (1.11) is computationally expensive to compute explicitly because one needs to invert a large sparse matrix which in turn yields a full matrix and multiply a full matrix with a sparse matrix twice. It is therefore necessary to compute an efficient approximation of the Schur complement. The research effort devoted to block preconditioners focus on computing a spectrally equivalent Schur complement $\tilde{S}$ [17, 49, 50, 51, 90, 96, 98, 107, 116, 121, 130]. The block preconditioners highly depend on efficient approximation of the Schur complement, which is problem dependent. Moreover, the standard block preconditioners do not give Reynolds number independent convergence for CFD problems [53], or regularization parameter independent convergence for PDE-constrained optimization problems [106, 107]. Although some recent developments in block preconditioners give regularization parameter independent convergence for PDE-constrained optimization problems [93, 94, 96, 97], they fail to solve boundary control or in-domain control of PDEs. Preconditioning boundary control or in-domain control of PDEs is still a big challenge for block preconditioners because the Schur complement for such problems is even more difficult to approximate than that for the standard PDE-constrained optimization problems where the control are distributed throughout the whole domain [94, 107].

In this dissertation, we focus on developing efficient preconditioners for linear system (1.10) based on multilevel sequentially semiseparable (MSSS) matrix computations. Compared with standard block preconditioners, such MSSS preconditioners compute an approximate factorization of the global system matrix $\mathcal{A}$ in (1.10) up to a prescribed accuracy in linear computational complexity. The motivation for such a global factorization is that the Schur complements in the LU factorization of the matrix from discretization of PDEs often have low numerical off-diagonal rank [12, 35]. Therefore, the Schur complements can be efficiently approximated by rank structured matrices with low off-diagonal rank, such as hierarchical matrices ($\mathcal{H}$-matrices) [66], $\mathcal{H}^2$-matrices [67], hierarchically semiseparable (HSS) matrices [32], and multilevel sequentially semiseparable (MSSS) matrices [108].

HSS matrix computations are usually applied in the multifrontal solver [134]. Some recent efforts devoted to preconditioning of symmetric positive definite systems are described in [132, 135]. As introduced in [108], 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 [12, 13], Bebendorf extended $\mathcal{H}$-matrices to solving elliptic PDEs. Preconditioning techniques based on $\mathcal{H}$-matrix computations for CFD problems were studied in [24, 78]. In [78], an $\mathcal{H}$-LU preconditioner was proposed to solve the convection-diffusion equation, while in [24] the augmented Lagrangian (AL) preconditioner based on $\mathcal{H}$-matrix computations was introduced to solve the discrete Oseen problems.

The advantage of MSSS matrix computations is their simplicity and low cost,

which is $\mathcal{O}(r^3N)$ with bounded small $r \ll N$ and N is the number of blocks, compared with $\mathcal{O}(N \log_2^\alpha N)$ with moderate α for $\mathcal{H}$-matrices and HSS matrices, where N is the size of the matrix. Using MSSS matrix computations to compute the preconditioner is motivated by the relation between interconnected systems and MSSS matrices, which is introduced in [108]. Once the grid topology 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 over $\mathcal{H}$-matrices and HSS matrices. The permutation operation plays a vital role for computing an approximate factorization of the global system matrix $\mathcal{A}$ in linear computational complexity, as will be explained in Chapter 2. For unstructured grids, HSS/$\mathcal{H}$-matrices are well suited. It was shown in [12, 133] 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 [115] 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.

1.4 Contributions and Outline

Recent research efforts on preconditioning using structured matrix computations either focus on symmetric positive definite systems [12, 64, 87, 132, 135], or linear systems whose matrix eigenvalues have positive real parts [78]. Structured matrix computations on preconditioning indefinite linear systems are studied in [11, 54] for the discretized Helmholtz equation, and in [24] for discrete Oseen problem from computational fluid dynamics problems. However, these efforts either focus on linear systems from discretized scalar PDEs, or Schur complement approximation for the 2-by-2 block system from coupled PDEs for block preconditioners.

In this dissertation, we focus on developing efficient preconditioning techniques for optimal flow control problems using multilevel sequentially semiseparable (MSSS) matrix computations. Compared with standard block preconditioners, MSSS preconditioners compute an approximate global factorization of the system matrix up to a prescribed accuracy in linear computational complexity. This in turn avoids approximating the Schur complement of the block system. This is a big advantage over standard block preconditioners.

The contributions and outline of this dissertation are given as follows.

- In Chapter 2, we study the MSSS preconditioner for PDE-constrained optimization problems where the control is distributed throughout the whole domain. The contributions in this chapter include
  - We propose a global MSSS preconditioner for the saddle-point system that arises from PDE-constrained optimization problems. This global

MSSS preconditioner exploits the global MSSS structure of the saddle-point system matrix, which avoids approximation of the Schur complement of the saddle-point matrix;

- We also study numerically the performance of the block preconditioners that are computed by MSSS matrix computations. Our experiments show that the global MSSS preconditioner gives mesh size and regularization parameter independent convergence while the block MSSS preconditioner only gives mesh size independent convergence;
- Model order reduction is the key to get linear computational complexity for MSSS matrix computations. We propose a new model order reduction algorithm for 1-level MSSS matrix computations. Compared with the standard model order reduction algorithm [33], the new algorithm is computationally cheaper;
- Since the model order reduction for 2-level MSSS matrices is still an open problem, extending MSSS preconditioning technique to 3D problems is nontrivial. We develop an alternative approach to preconditioning optimal control of PDEs in 3D.

- In Chapter 3, we apply the MSSS preconditioners to computational fluid dynamics problems and evaluate their performance using Incompressible Flow & Iterative Solver Software (IFISS) [117]. The contributions are

  - We apply the MSSS preconditioner to the convection-dominated convection-diffusion problem and compare its performance with the algebraic multigrid (AMG) and geometric multigrid (GMG) methods. Numerical results show that the MSSS preconditioner is much more robust than both AMG and GMG methods;
  - We apply both block MSSS preconditioner and global MSSS preconditioner to the Stokes equation and compare their performance with AMG and GMG based block preconditioners;
  - We test the global MSSS preconditioner for the linearized Navier-Stokes equation and compare its performance with the pressure convection-diffusion (PCD) preconditioner [53] that uses the AMG method to approximate the $(1,1)$ block and both the AMG method and lumped pressure mass matrix to approximate the Schur complement. Reynolds number independent and mesh size independent convergence are obtained for the global MSSS preconditioner while only mesh size independent convergence is obtained for the PCD preconditioner.

- In Chapter 4, we make an analytical study of the convergence property of the MSSS preconditioner. The contributions in this chapter include

  - We extend the convergence analysis in [11, 87] for the symmetric positive definite case to general linear systems, where no symmetry or definiteness is assumed;
  - Our analysis also applies to block linear systems, while related work only considers the linear system from discretized scalar PDEs [11, 87];

  - We give an analytic bound for the error introduced by the model order reduction that is necessary for the MSSS preconditioning technique;
  - The analysis for MSSS preconditioning in [87] only concerns 1-level MSSS matrix computations, while our analysis also includes the 2-level MSSS case;
  - For the first time, we apply the MSSS preconditioning technique to the Helmholtz equation.

- We apply the MSSS preconditioner to the in-domain optimal control of PDEs in Chapter 5 and use a simplified wind farm control example to study the performance of MSSS preconditioners. The contributions in this chapter include,
  - We formulate an in-domain optimal control of PDEs problem and apply the reduced Newton's method based on implicit function theorem to solve this optimization problem. The reduced Newton's method yields a generalized saddle-point system to be solved;
  - By using the reduced Newton's method, we can compute the gradient and Hessian matrix with very little cost and reduce the computational complexity significantly;
  - The Schur complement of the generalized saddle-point system is quite difficult or even impossible to approximate, which prohibits a satisfactory performance of the block preconditioners. We apply the global MSSS preconditioner to solve the generalized saddle-point system;
  - We evaluate the performance of the global MSSS preconditioner using IFISS and compare its performance with standard block preconditioners.

- Conclusions are drawn in Chapter 6, where discussions and future research recommendations are also given.

2 Chapter

Efficient Preconditioners for PDE-Constrained Optimization Problems with MSSS Structure

PDE -constrained optimization problems yield a linear saddle-point system that has to be solved. We propose a preconditioner that makes use of the global MSSS structure and a preconditioner that exploits the block MSSS structure of the saddle-point system. For the computation of preconditioners based on MSSS matrix computations, model order reduction algorithms are essential to obtain a low computational complexity. We study two different model order reduction approaches, one is the new approximate balanced truncation with low-rank approximated gramians for SSS matrices and the other is the standard Hankel blocks approximation algorithm. We test our preconditioners on the problems of optimal control of the convection-diffusion equation in 2D and optimal control of 3D Poisson equation. For 2D problems, numerical experiments illustrate that both preconditioners have linear computational complexity and the global MSSS preconditioner reduces the number of iterations significantly and needs less computation time. Moreover, the approximate balanced truncation algorithm is computationally cheaper than the Hankel blocks approximation algorithm. Except for the mesh size independent convergence, the global MSSS preconditioner also gives regularization parameter independent convergence, while the block MSSS preconditioner just gives mesh size independent convergence. For 3D problems, both the block MSSS preconditioner and global MSSS preconditioner give nearly mesh size independent convergence. The global MSSS preconditioner reduces the number of iterations dramatically compared with the block MSSS preconditioner.

2.1 Introduction

PDE-constrained optimization problems have a wide application such as optimal flow control [22, 23], diffuse optical tomography [1], and linear (nonlinear) model predictive control [20]. The solution of these problems can be obtained by solving a large-scale linear system of the saddle-point type. Much effort has been dedicated to finding efficient iterative solution methods for such systems. Their performance highly depends on the choice of preconditioners. In this chapter, we study a class of preconditioners that exploits the multilevel sequentially semiseparable (MSSS) structure of the blocks of the saddle-point system.

Semiseparable matrices appear in several types of applications, e.g. integral equations [62], Gauss-Markov processes [73], boundary value problems [63], rational interpolation [124] and Kalman filtering [101]. Semiseparable matrices are matrices of which all the sub-matrices taken from the lower-triangular or the upper-triangular part are of rank at most 1 [128]. Sequentially semiseparable (SSS) matrices of which the off-diagonal blocks are of low-rank, not limited to 1, introduced by Dewilde et al. in [33] generalize the semiseparable matrices. Multilevel sequentially semiseparable matrices generalize the sequentially semiseparable matrices to the multi-dimensional cases. Systems that arise from the discretization of 1D partial differential equations typically have an SSS structure. Discretization of higher dimensional (2D or 3D) partial differential equations give rise to matrices that have an MSSS structure [40, 61]. Under the multilevel paradigm, generators that are used to represent a matrix of a higher hierarchy are themselves multilevel sequentially semiseparable of a lower hierarchy. The usual one-level sequentially semiseparable matrix is the one of the lowest hierarchy. Operations like the matrix inversion and the matrix-matrix multiplication are closed under this structure. The LU factorization can also be performed in a structure preserving way. This factorization results in a growth of the rank of the off-diagonal blocks of the Schur complements. As a result, the LU factorization is not of linear computational complexity because the low rank property of the off-diagonal blocks is the key to obtain linear computational complexity for MSSS matrix computations. Model order reduction can be used to reduce the rank of the off-diagonal blocks, which yields an inexact LU decomposition of an MSSS matrix that can be used as a preconditioner.

In [61], Gondzio et al. first introduced the MSSS matrix computations for preconditioning PDE-constrained optimization problems. They exploited the MSSS matrix structure of the blocks of the saddle-point system and performed an LU factorization using MSSS matrix computations to approximate the Schur complement of the saddle-point system. With this approximated Schur complement as a preconditioner, conjugate gradient iterations were performed to solve the saddle-point system block-by-block. As aforementioned, the model order reduction plays a vital role in obtaining a linear computational complexity of the LU factorization for MSSS matrices. In [61], Gondzio et al. used a standard model order reduction algorithm [33, 46] to reduce the computational complexity.

Our contribution in this chapter include: (1) We propose a new model order reduction algorithm for SSS matrix computations based on the correspondence between linear time-varying (LTV) systems and blocks of SSS matrices. This new model

order reduction algorithm is motivated by the work in [29, 31]. In [29], the approximate balanced truncation was addressed for the model order reduction of linear time invariant (LTI) systems, while in [31] the recursive low-rank approximation was performed to approximate the gramians of LTV systems. In this chapter, we use the low-rank approximation method in [31] and the approximate balanced truncation in [29] for the model order reduction for the SSS matrices. Compared with the model order reduction algorithms discussed in [33, 46], the approximate balanced truncation method for SSS matrices in this chapter is computationally cheaper. (2) With these model order reduction algorithms, we can compute an inexact LU factorization for the MSSS matrix blocks of the saddle-point system in linear computational complexity ($\mathcal{O}(N)$). This yields a block MSSS preconditioner for the saddle-point system. Exploiting the block structure of the saddle-point system is the standard preconditioning technique, which is described in [15]. However, only the single preconditioner for the last block of the saddle-point system is studied in [61]. (3) By permuting the blocks of the saddle-point system, we can also compute an inexact LU factorization of the global system with MSSS matrix computations in linear computational complexity. This gives a global MSSS preconditioner and this novel MSSS preconditioner gives mesh size and regularization parameter independent convergence. This is a big advantage over the block preconditioners. (4) Besides the problem of optimal control of the Poisson equation, we also study the problem of optimal control of the convection-diffusion equation. (5) Moreover, we extend these preconditioning techniques to the 3D saddle-point systems.

Note that the convergence of the block preconditioners depend on the regularization parameter β for the PDE-constrained optimization problems [106]. For small β, block preconditioners do not give satisfactory performance. Recent development in PDE-constrained optimization problems also gives β independent convergence [95, 96]. However, they fail to solve PDE-constrained optimization problems where control only act on few parts of the domain. The block preconditioners in [106] are more general. In this chapter, we apply block preconditioners developed in [106] computed by MSSS matrix computations and compare their performance with global MSSS preconditioner. Numerical results show that the global MSSS preconditioner gives not only mesh size independent convergence but also β independent convergence. Block MSSS preconditioners only yield mesh size independent convergence.

The outline of this chapter is as follows: Section 2.2 formulates a distributed optimal control problem constrained by PDEs. This problem yields a linear saddle-point system. We introduce the MSSS matrix computations in Section 2.3. In Section 2.4, we introduce the MSSS preconditioning technique. The new model order reduction algorithm for SSS matrices is also described. With MSSS matrix computations, we propose two types of preconditioners for saddle-point problem: the global MSSS preconditioner, and the block-diagonal MSSS preconditioner. In Section 2.5, we use the distributed optimal control of the convection-diffusion equation to illustrate the performance of these two preconditioners and the new model order reduction algorithm. Section 2.6 presents how to extend such preconditioning techniques to 3D saddle-point problems, and Section 2.7 draws the conclusions.

2.2 Problem Formulation

2.2.1 PDE-Constrained Optimization Problem

Consider the following PDE-constrained optimization problem described by

$$\begin{aligned} \min_{u,\ f} \ & \frac{1}{2}\|u-\hat{u}\|^2 \quad + \quad \beta\|f\|^2 \\ s.t. \ & \mathcal{L}u \quad = \quad f \quad \text{in } \Omega \\ & u \quad = \quad u_D \text{ on } \partial\Omega, \end{aligned} \tag{2.1}$$

where $\mathcal{L}$ is an operator, u is the system state, f is the system input and $\hat{u}$ is the desired state of the system, Ω is the domain and $\partial\Omega$ is the corresponding boundary, β is the weight of the system input in the cost function or the regularization parameter and $\beta > 0$. In this chapter, we consider $\mathcal{L} = -\nabla^2$ for optimal control of the Poisson equation and $\mathcal{L} = -\nu\nabla^2 + \overrightarrow{w}\cdot\nabla$ for optimal control of the convection-diffusion equation. Here $\overrightarrow{w}$ is a vector in Ω, ∇ is the gradient operator, and ν is a positive scalar. If we want to solve such a problem numerically, it is clear that we need to discretize these quantities involved at some point. There are two kinds of approaches, one is to derive the optimality conditions first and then discretize from there (*optimize-then-discretize*), the other is to discretize the cost function and the PDE first and then optimize that (*discretize-then-optimize*). For the problem of optimal control of the Poisson equation, both approaches lead to the same solution while different answers are reached for the problem of optimal control of the convection-diffusion equation [106]. Since our focus is on preconditioning, the *discretize-then-optimize* approach is chosen in this chapter.

By introducing the weak formulation and discretizing (2.1) using the Galerkin method, the discrete analogue of the minimization problem (2.1) is therefore given by

$$\begin{aligned} \min_{u,\ f} \ & \frac{1}{2}u^T M u - u^T b \quad + \quad c + \beta f^T M f \\ s.t. \ & Ku = Mf \quad + \quad d, \end{aligned} \tag{2.2}$$

where $K = [K_{i,j}] \in \mathbb{R}^{N\times N}$ with $K_{ij} = \int_\Omega \nabla\phi_i \nabla\phi_j d\Omega$ is the stiffness matrix, $M = [M_{i,j}] \in \mathbb{R}^{N\times N}$, $M_{ij} = \int_\Omega \phi_i\phi_j d\Omega$ is the mass matrix and is symmetric positive definite, $b = [b_i] \in \mathbb{R}^N$, $b_i = \int_\Omega \hat{u}_i\phi_i d\Omega$, $c \in \mathbb{R}$, $c = \int_\Omega \hat{u}^2 d\Omega$, $d = [d_i] \in \mathbb{R}^N$, $d_i = -\sum_{j=N+1}^{N+\partial N} u_j \int_\Omega \nabla\phi_j \cdot \nabla\phi_i d\Omega$. The ϕ_i ($i = 1,\ 2,\ \ldots\ N$) and ϕ_j ($j = 1,\ 2,\ \ldots\ N,\ N+1,\ \ldots\ N+\partial N$) form a basis of V_0^h and V_g^h, respectively. Here V_0^h and V_g^h are finite dimensional test space and solution space.

Consider the cost function in (2.2) and associate with the equality constraint, we

introduce the Lagrangian function

$$\mathcal{J}(u,f,\lambda) = \frac{1}{2}u^T M u - u^T b + c + \beta f^T M f + \lambda^T (Ku - Mf - d),$$

where λ is the Lagrange multiplier. Then it is well-known that the optimal solution is given by finding u, f and λ such that

$$\begin{aligned} \nabla_u \mathcal{J}(u,f,\lambda) &= Mu - b + K^T\lambda = 0, \\ \nabla_f \mathcal{J}(u,f,\lambda) &= 2\beta M f - M\lambda = 0, \\ \nabla_\lambda \mathcal{J}(u,f,\lambda) &= Ku - Mf - d = 0. \end{aligned}$$

This yields the linear system

$$\underbrace{\begin{bmatrix} 2\beta M & 0 & -M \\ 0 & M & K^T \\ -M & K & 0 \end{bmatrix}}_{\mathcal{A}} \underbrace{\begin{bmatrix} f \\ u \\ \lambda \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} 0 \\ b \\ d \end{bmatrix}}_{g}. \tag{2.3}$$

The system (2.3) is of the saddle-point system type [15], i.e., the system matrix $\mathcal{A}$ is symmetric and indefinite. It has the following structure

$$\mathcal{A} = \begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix}, \tag{2.4}$$

where $A \in \mathbb{R}^{n\times n}$ is symmetric positive definite, $B \in \mathbb{R}^{m\times n}$ has full column rank.

The system matrix of the saddle-point system (2.3) is large, sparse and highly indefinite. Preconditioned Krylov subspace methods, such as `MINRES` [92] and `IDR(s)` [119], are quite efficient for solving such systems.

2.2.2 Preconditioning Saddle-Point Systems

The performance of iterative solution methods highly depends on the choice of the preconditioners. For numerical methods to solve saddle-point system (2.3) and the construction of preconditioners, we refer to [15, 89] for an extensive survey. In this chapter, we study two types of preconditioners. The first exploits the MSSS structure of the blocks of the saddle-point system, whereas the second type exploits the global MSSS structure of the saddle-point system.

Block Preconditioners

Recall from (2.4), if A is nonsingular, then $\mathcal{A}$ admits the following LDL^T factorization given by

$$\begin{bmatrix} 2\beta M & 0 & -M \\ 0 & M & K^T \\ -M & K & 0 \end{bmatrix} = \begin{bmatrix} I & & \\ 0 & I & \\ -\frac{1}{2\beta}I & KM^{-1} & I \end{bmatrix} \begin{bmatrix} 2\beta M & & \\ & M & \\ & & S \end{bmatrix} \begin{bmatrix} I & 0 & -\frac{1}{2\beta}I \\ & I & M^{-1}K^T \\ & & I \end{bmatrix},$$

where $S = -\left(\frac{1}{2\beta}M + KM^{-1}K^T\right)$ is the Schur complement.

The most difficult part for this factorization is to compute the Schur complement S because computing the inverse of a large sparse matrix is expensive both in time and memory. Meanwhile, solving the system $Sx = b$ is also expensive since S is a large and full matrix. Note that all the blocks of (2.3) have a structure that is called multilevel sequentially semiseparable (MSSS), which will be introduced in the later section. Then the Schur complement S also has the MSSS structure but with a bigger semiseparable order. If we exploit the MSSS structure of (2.3), we can compute S in linear computational complexity.

In this chapter, we first study the block-diagonal preconditioner $\mathcal{P}_1$ for the saddle-point system (2.3), where

$$\mathcal{P}_1 = \begin{bmatrix} 2\beta\hat{M} & & \\ & \hat{M} & \\ & & -\hat{S} \end{bmatrix}. \tag{2.5}$$

Here $\hat{M}$ is an approximation of the mass matrix M and $\hat{S}$ is an approximation of the Schur complement S.

To approximate $S = -\left(\frac{1}{2\beta}M + KM^{-1}K^T\right)$, $\hat{S} = -KM^{-1}K^T$ can be used for big to middle range of β while $\hat{S} = -\frac{1}{2\beta}M$ could be chosen for small β [107]. The block lower-triangular preconditioner $\mathcal{P}_2$, which has the following form

$$\mathcal{P}_2 = \begin{bmatrix} 2\beta\hat{M} & & \\ 0 & \hat{M} & \\ -M & K & \hat{S} \end{bmatrix}, \tag{2.6}$$

is studied in the appendix of this dissertation.

Global Preconditioners

Since all the blocks of the saddle-point system (2.3) have the MSSS structure, there exists a permutation matrix Ψ that permutes the saddle-point matrix with MSSS blocks into a single MSSS matrix. This gives

$$\tilde{\mathcal{A}}\tilde{x} = \tilde{g}, \tag{2.7}$$

where $\tilde{\mathcal{A}} = \Psi\mathcal{A}\Psi^T$, $\tilde{x} = \Psi x$, and $\tilde{g} = \Psi g$ are permutations of $\mathcal{A}$, $\begin{bmatrix} f^T & u^T & \lambda^T \end{bmatrix}^T$, and $\begin{bmatrix} 0^T & b^T & d^T \end{bmatrix}^T$ in (2.3), respectively. This permutation will be introduced in the next section. After this permutation, the system matrix $\tilde{\mathcal{A}}$ is an MSSS matrix. We can compute an inexact LU factorization of $\tilde{\mathcal{A}}$ in linear computational complexity using MSSS matrix computations. This gives,

$$\tilde{\mathcal{A}} \approx \tilde{L}\tilde{U}, \tag{2.8}$$

which can be used as a preconditioner. We call this factorization (2.8) the global MSSS preconditioner. Since no information of β is dropped directly during the permutation and factorization, if we can compute a relatively accurate factorization, the global MSSS preconditioner will give β-independent convergence. This property for the standard block preconditioners $\mathcal{P}_1$ in (2.5) or $\mathcal{P}_2$ in (2.6) do not hold. This is a big advantage of the global MSSS preconditioner over the standard block preconditioners. Numerical examples in Section 2.5 demonstrate this advantage. Theoretical analysis in Chapter 4 states the underlying principle.

2.3 Multilevel Sequentially Semiseparable Matrices

Matrices in this chapter 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 of the sequentially semiseparable matrices are defined by Definition 2.1.

Definition 2.1 ([34]) *Let A be an $n \times n$ block matrix with SSS structure such that A can be written in the following block-partitioned form*

$$A_{ij} = \begin{cases} U_i W_{i+1} \cdots W_{j-1} V_j^T, & \text{if } i < j; \\ D_i, & \text{if } i = j; \\ P_i R_{i-1} \cdots R_{j+1} Q_j^T, & \text{if } i > j. \end{cases} \tag{2.9}$$

where the superscript ' T ' denotes the transpose of a matrix, and the sizes of $\{U_i\}_{i=1}^{n-1}$, $\{W_i\}_{i=2}^{n-1}$, $\{V_i\}_{i=2}^{n}$, $\{D_i\}_{i=1}^{n}$, $\{P_i\}_{i=2}^{n}$, $\{R_i\}_{i=2}^{n-1}$, $\{Q_i\}_{i=1}^{n-1}$ are listed in Table 2.1.

Table 2.1: Generators size for the SSS matrix A in Definition 2.1

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

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 and they are called generators of the SSS matrix A. With the generators representation defined in Definition 2.1, A can be denoted by

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

Note that we use the symbol $\mathcal{SSS}$ to represent a matrix that has an SSS structure. The parameters inside the brackets represent the sets of generators that parameterize an SSS matrix.

Remark 2.1 The generators of an SSS matrix are not unique, there exists a series of nonsingular transformations between two different sets of generators for the same SSS matrix A.

Remark 2.2 For an SSS matrix, only its generators are stored. If l_i and k_i are bounded by a small constant. Then the memory consumption for storing such matrix is linear with respect to the matrix size. This property is also introduced in [34].

Take $n = 4$ for example, the SSS matrix A is given by (2.10),

$$A = \begin{bmatrix} D_1 & U_1V_2^T & U_1W_2V_3^T & U_1W_2W_3V_4^T \\ P_2Q_1^T & D_2 & U_2V_3^T & U_2W_3V_4^T \\ P_3R_2Q_1^T & P_3Q_2^T & D_3 & U_3V_4^T \\ P_4R_3R_2Q_1^T & P_4R_3Q_2^T & P_4Q_3^T & D_4 \end{bmatrix}. \tag{2.10}$$

With the generators representation of SSS matrices, basic operations such as addition, multiplication and inversion are closed under the SSS matrix structure and can be performed in linear computational complexity. Moreover, decompositions/factorizations such as the QR factorization [45, 47], the LU decomposition [61, 128], and the ULV decomposition [125] can also be computed in linear computational complexity and in a structure preserving way.

Similar to Definition 2.1 for SSS matrices, the generators representation for MSSS matrices, specifically the k-level SSS matrices, is defined in Definition 2.2.

Definition 2.2 *The matrix A is said to be a k-level SSS matrix if it has a form like* (2.9) *and all its generators are $(k-1)$-level SSS matrices. The* 1*-level SSS matrix is the SSS matrix that satisfies Definition 2.1.*

Most operations for the SSS matrices can be extended to the MSSS matrices, which yields linear computational complexity for MSSS matrices. MSSS matrices have many applications, one of them is discretized partial differential equations (PDEs) [100].

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 2D Laplace equation with homogeneous Dirichlet boundary conditions on a square domain, discretized using the Q_1 finite element method on a uniform mesh, the system matrix is given by*

$$K = \begin{bmatrix} A & B & & \\ B & A & \ddots & \\ & \ddots & \ddots & B \\ & & B & A \end{bmatrix},$$

where

$$A = \begin{bmatrix} {}^{8}/_{3} & -{}^{2}/_{3} & & & \\ -{}^{2}/_{3} & {}^{8}/_{3} & -{}^{2}/_{3} & & \\ & -{}^{2}/_{3} & \ddots & \ddots & \\ & & \ddots & \ddots & -{}^{2}/_{3} \\ & & & -{}^{2}/_{3} & {}^{8}/_{3} \end{bmatrix}, \; B = \begin{bmatrix} -{}^{2}/_{3} & -{}^{2}/_{3} & & & \\ -{}^{2}/_{3} & -{}^{2}/_{3} & -{}^{2}/_{3} & & \\ & -{}^{2}/_{3} & \ddots & \ddots & \\ & & \ddots & \ddots & -{}^{2}/_{3} \\ & & & -{}^{2}/_{3} & -{}^{2}/_{3} \end{bmatrix}.$$

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

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

Note that we use the symbol $\mathcal{MSSS}$ here to represent a matrix that has an MSSS structure. The parameters inside the brackets represent the sets of generators that parameterize an MSSS matrix.

Remark 2.3 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 2.1 are satisfied for the Definition 2.1.

Note that for a saddle-point system from the PDE-constrained optimization problem, all its blocks are MSSS matrices. This enables us to compute an LU factorization of all its blocks with MSSS matrix computations in linear computational complexity. However, the saddle-point matrix is not itself an MSSS matrix but just has MSSS blocks, hence we cannot compute an approximate LU factorization of the saddle-point system matrix by using MSSS matrix computations.

Lemma 2.4 explains how to permute a matrix with SSS blocks into a single SSS matrix. This property can be extended to matrices with MSSS blocks. This enables us to compute an LU factorization of the global saddle point matrix by using MSSS matrix computations in linear computational complexity.

Lemma 2.4 ([108]) *Let A, B, C and D be SSS matrices with the following generators representations*

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

$$D \quad = \quad \mathcal{SSS}(P_s^d, R_s^d, Q_s^d, D_s^d, U_s^d, W_s^d, V_s^d).$$

Then there exists a permutation matrix Ψ *with* $\Psi\Psi^T = \Psi^T\Psi = I$ *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

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

$$U_s^t = \begin{bmatrix} U_s^a & U_s^b & 0 & 0 \\ 0 & 0 & U_s^c & U_s^d \end{bmatrix}, \qquad V_s^t = \begin{bmatrix} V_s^a & 0 & V_s^c & 0 \\ 0 & V_s^b & 0 & V_s^d \end{bmatrix},$$

$$W_s^t = \begin{bmatrix} W_s^a & & & \\ & W_s^b & & \\ & & W_s^c & \\ & & & W_s^d \end{bmatrix}, \qquad R_s^t = \begin{bmatrix} R_s^a & & & \\ & R_s^b & & \\ & & R_s^c & \\ & & & R_s^d \end{bmatrix}.$$

Proof: For the case that all the diagonal blocks of A have the same size and all the diagonal blocks of D also have the same size, i.e., $m_i^a = m^a$ and $m_i^d = m^d$, the permutation matrix Ψ has the following representation

$$\Psi = \left[\begin{bmatrix} I_{m^a} \\ 0 \end{bmatrix} \otimes I_n \quad \begin{bmatrix} 0 \\ I_{m^d} \end{bmatrix} \otimes I_n \right], \tag{2.11}$$

where $\otimes$ denotes the Kronecker product and I_n is an $n \times n$ identify matrix.
With the permutation matrix Ψ given by (2.11), the permuted matrix is given by

$$\mathcal{T} = \Psi \begin{bmatrix} A & B \\ C & D \end{bmatrix} \Psi^T. \tag{2.12}$$

And we can explicitly write the matrix $\mathcal{T}$ as

(2.13)

$$\mathcal{T} = \begin{bmatrix} \begin{pmatrix} D_1^a & D_1^b \\ D_1^c & D_1^d \end{pmatrix} & \begin{pmatrix} U_1^a V_2^{aT} & U_1^b V_2^{bT} \\ U_1^c V_2^{cT} & U_1^d V_2^{dT} \end{pmatrix} & \begin{pmatrix} U_1^a W_2^a V_3^{aT} & U_1^b W_2^b V_3^{bT} \\ U_1^c W_2^c V_3^{cT} & U_1^d W_2^d V_3^{dT} \end{pmatrix} & \cdots \\ \begin{pmatrix} P_2^a Q_1^{aT} & P_2^b Q_1^{bT} \\ P_2^c Q_1^{cT} & P_2^d Q_1^{dT} \end{pmatrix} & \begin{pmatrix} D_2^a & D_2^b \\ D_2^c & D_2^d \end{pmatrix} & \cdots & \ddots \\ \begin{pmatrix} P_3^a R_2^a Q_1^{aT} & P_3^b R_2^b Q_1^{bT} \\ P_3^c R_2^c Q_1^{cT} & P_3^d R_2^d Q_3^{dT} \end{pmatrix} & \cdots & \ddots & \ddots \\ \vdots & \ddots & \ddots & \ddots \end{bmatrix}.$$

It is clear to see that $\mathcal{T}$ is an SSS matrix and not difficult to verify that its generators are given in Lemma 2.4.

For the case that sizes of diagonal blocks A and D are varying, let $\{m_i^a\}_{i=1}^n$ and $\{m_i^d\}_{i=1}^n$ represent the diagonal blocks size of A and D, respectively. The permutation matrix Ψ is given by

$$\Psi = \left[\text{blkdiag}\left(\begin{bmatrix} I_{m_i^a} \\ 0 \end{bmatrix}\right) \quad \text{blkdiag}\left(\begin{bmatrix} 0 \\ I_{m_i^d} \end{bmatrix}\right)\right]. \tag{2.14}$$

Here in (2.14),

- $\text{blkdiag}\left(\begin{bmatrix} I_{m_i^a} \\ 0 \end{bmatrix}\right)$ is a block diagonal matrix with $\left\{\begin{bmatrix} I_{m_i^a} \\ 0 \end{bmatrix}\right\}_{i=1}^n$ as its diagonal blocks,
- $\text{blkdiag}\left(\begin{bmatrix} 0 \\ I_{m_i^d} \end{bmatrix}\right)$ is a block diagonal matrix with $\left\{\begin{bmatrix} 0 \\ I_{m_i^d} \end{bmatrix}\right\}_{i=1}^n$ as its diagonal blocks.

With the permutation matrix Ψ in (2.14), we can also get the permuted matrix $\mathcal{T}$ that has the form being represented by (2.12). Thus, it is an SSS matrix and its generators are given in Lemma 2.4. □

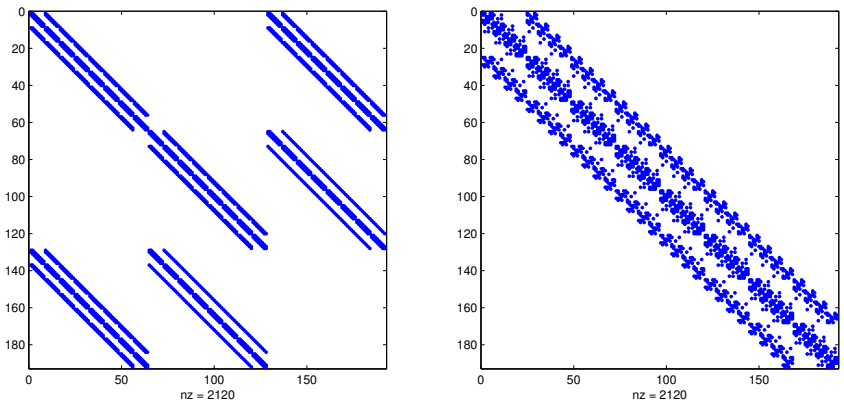
Remark 2.5 To one matrix with SSS blocks, one can apply Lemma 2.4 to permute it into a single SSS matrix by using a permutation matrix Ψ. However, this permutation matrix is not explicitly multiplied on both sides of the matrix to be permuted. The generators of the permuted matrix are combinations of the generators of its SSS blocks. This is illustrated by the generators representation of the permuted matrix in Lemma 2.4. Such permutations are cheaper to compute due to the fact that there is no matrix-matrix multiplication.

Remark 2.6 Lemma 2.4 is for a 2×2 block matrix, but it generalizes the case of matrices with different numbers of blocks.

Remark 2.7 Extending Lemma 2.4 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 2.4 recursively.

For the saddle-point system (2.3) derived from the optimal control of the convection-diffusion equation in $[0,\ 1] \times [0,\ 1]$, discretizing using the Q_1 finite element method on a uniform mesh yields a saddle-point system that has MSSS (2-level SSS) matrix blocks. The spy plot of the saddle-point matrix before and after permutation for mesh size $h = 2^{-3}$ are shown in Figure 2.1.

Here in Figure 2.1, "nz" denotes the number of nonzeros in a sparse matrix.



(a) Before permutation. (b) After permutation.

Figure 2.1: Structure of system matrix of (2.3) before and after permutation for $h = 2^{-3}$.

2.4 Multilevel Sequentially Semiseparable Preconditioners

The most important part of the PDE-constrained optimization problem is to solve a linear system of the saddle-point type. In the following part, we first introduce the *LU* factorization of MSSS matrices and then give a new model order reduction algorithm for SSS matrices, which is necessary for computing an approximate *LU* factorization in linear computational complexity. For comparison, the conventional model order reduction algorithm [33] is also discussed.

2.4.1 *LU* Factorization of Multilevel Sequentially Semiseparable Matrices

The semiseparable order defined in Definition 2.3 plays an important rule in the MSSS matrix computations. Note that Dewilde et al. and Eidelman et al. studied this kind of structured matrices independently, sequentially semiseparable matrices named in [34] are called quasiseparable matrices in [44]. In this chapter, we use the MATLAB style of notation for matrices, 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 2.3 ([46]) *Let A be an $n \times n$ block matrix and*

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

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

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

The numbers $u_s (s = 1, 2, \cdots, n-1)$ *are called the upper order numbers of the matrix* A. *Set* $r^l = \max l_s$ *and* $r^u = \max u_s$, *where* r^l *and* r^u *are called the lower quasiseparable order and the upper quasiseparable order of* A, *respectively.*

Definition 2.4 ([108]) *The SSS matrix* A *with lower and upper semiseparable order* r^l *and* r^u *is called block* $(r^l,\ r^u)$ *semiseparable.*

The semiseparable order for 1-level SSS matrices defined in Definition 2.3 can be directly extended to the multilevel cases, which leads to Definition 2.5.

Definition 2.5 *Let the matrix* A *be an* $N \times N$ *block* k*-level SSS matrix with its generators be* $(k-1)$*-level SSS matrices. Let*

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

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

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

The numbers $u_s (s = 1, 2, \cdots, N-1)$ *are called the* k*-level upper order numbers of the* k*-level SSS matrix* A. *Set* $r^l = \max l_s$ *and* $r^u = \max u_s$, *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 2.6 *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.*

By using these definitions, we can apply the following lemma to compute an LU factorization of a k-level SSS matrix.

Lemma 2.8 ([61, 128]) *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 if 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*

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

where $\hat{Q}_s$, D_s^L, D_s^U and $\hat{U}_s$ are $(k-1)$-level sequentially semiseparable matrices. They are computed by Algorithm 2.1.

Algorithm 2.1 *LU* factorization of a k-level SSS matrix A

1: **procedure** $LU(A)$ ▷ LU factorization of MSSS matrix A
2: **Input:** $(k-1)$-level SSS generators of A
3: $D_1 = D_1^L D_1^U$ (LU factorization of $(k-1)$-level SSS matrix)
4: Let $\hat{U}_1 = (D_1^L)^{-1} U_1$ and $\hat{Q}_1 = (D_1^L)^{-T} Q_1$
5: **for** $i = 2 : N-1$ **do**
6: **if** $i == 2$ **then**
7: $M_i = \hat{Q}_{i-1}^T \hat{U}_{i-1}$
8: **else**
9: $M_i = \hat{Q}_{i-1}^T \hat{U}_{i-1} + R_{i-1} M_{i-1} W_{i-1}$
10: **end if**
11: $\left(D_i - P_i M_i V_i^T\right) = D_i^L D_i^U$ (LU factorization of $(k-1)$-level SSS matrix)
12: Let $\hat{U}_i = (D_i^L)^{-1}(U_i - P_i M_i W_i)$, $\hat{Q}_i = (D_i^U)^{-T}(Q_i - V_i M_i^T R_i^T)$.
13: **end for**
14: $M_N = \hat{Q}_{N-1}^T \hat{U}_{N-1} + R_{N-1} M_{N-1} W_{N-1}$
15: $\left(D_N - P_N M_N V_N^T\right) = D_N^L D_N^U$ (LU factorization of $(k-1)$-level SSS matrix)
16: **Output:** $\left\{D_s^L\right\}_{s=1}^N$, $\left\{D_s^U\right\}_{s=1}^N$, $\left\{\hat{Q}_s\right\}_{s=1}^{N-1}$, $\left\{\hat{U}_s\right\}_{s=1}^{N-1}$
17: **end procedure**

Proof: For the proof of the lemma, we refer to [61, 128]. □

Remark 2.9 In Algorithm 2.1, the *LU* factorization of a 0-level SSS matrix is just the *LU* factorization of an ordinary matrix without SSS structure.

In Algorithm 2.1, for computing the *LU* factorization of a k-level SSS matrix, the matrix-matrix operations are performed on its $(k-1)$-level SSS generators, such as computing the recurrence of M_i in line 9 of Algorithm 2.1. Such operations lead to the growth of the $(k-1)$-level semiseparable order, which increases the computational complexity. This can be verified from the matrix-matrix operations introduced in [34, 44]. Take the 1-level SSS matrix A for example, the flops needed for computing A^2 is $\mathcal{O}(n^3 N)$, where n is the semiseparable order and N is the number of blocks of A [34]. To be specific, the following lemma is introduced.

Lemma 2.10 ([44]) *Let A_1, A_2 be SSS matrices of lower semiseparable order m_1 and n_1, respectively. Then the product $A_1 A_2$ is of lower semiseparable order at most $m_1 + n_1$. Let A_1, A_2 be SSS matrices of upper semiseparable order m_2 and n_2, respectively. Then the product $A_1 A_2$ is upper semiseparable of order at most $m_2 + n_2$.*

Remark 2.11 For a k-level SSS matrix, since the semiseparable order varies at different levels, results of Lemma 2.10 also hold for the k-level semiseparable order. But we do not know the exact upper bound of the $(k-1)$-level semiseparable order. We just know the $(k-1)$-level semiseparable order also increases.

Lemma 2.10 states that the semiseparable order grows by multiplying two SSS matrices, this also holds for adding two SSS matrices. There are similar results for multilevel SSS matrices multiplication and addition. Model order reduction is necessary to reduce the semiseparable order and keep the computational complexity of Algorithm 2.1 low. The aim of model order reduction for a k-level SSS matrix A with its generators representation $A = \mathcal{MSSS}(P_s,\ R_s,\ Q_s,\ D_s,\ U_s,\ W_s,\ V_s)$ is to find $(k-1)$-level SSS matrices $\hat{P}_s$, $\hat{R}_s$, $\hat{Q}_s$, $\hat{U}_s$, $\hat{W}_s$, $\hat{V}_s$ of smaller order compared with P_s, R_s, Q_s, U_s, W_s, V_s, respectively such that $\hat{A} = \mathcal{MSSS}(\hat{P}_s,\ \hat{R}_s,\ \hat{Q}_s,\ D_s,\ \hat{U}_s,\ \hat{W}_s,\ \hat{V}_s)$ is of k-level semiseparable order smaller than or equal to the minimal k-level semiseparable order of A. Meanwhile, $\hat{A}$ is an approximation of A up to a small tolerance ε, i.e., $\|\hat{A} - A\| < \varepsilon$.

Remark 2.12 Since the LU factorization of a k-level SSS matrix needs the model order reduction for $(k-1)$-level SSS matrices, the LU factorization in Lemma 2.8 is an exact factorization for SSS matrices because no model order reduction is needed for ordinary matrices (0-level SSS matrices). It is an inexact factorization for the k-level ($k \geq 2$) SSS matrices.

For discretized one-dimensional PDEs on a regular grid, the system matrix has a certain SSS structure. The LU factorization introduced in Lemma 2.8 could be performed as a direct solver. For discretized higher dimensional PDEs on regular grids, this LU factorization can be used as an efficient preconditioner.

2.4.2 Approximate Balanced Truncation for SSS Matrices

As introduced in the last subsection, the model order reduction plays a key role in the LU factorization of an MSSS matrix. In this subsection, we design a new model order reduction algorithm for SSS matrices. This new method exploits the correspondence between SSS matrices and linear time-varying (LTV) systems.

The SSS matrices have a realization of linear time-varying systems, which is studied by Dewilde et al. in [41]. Consider a mixed-causal system that is described by the following state-space model

$$\begin{aligned} \begin{bmatrix} x_{i+1}^c \\ x_{i-1}^a \end{bmatrix} &= \begin{bmatrix} R_i & \\ & W_i \end{bmatrix} \begin{bmatrix} x_i^c \\ x_i^a \end{bmatrix} + \begin{bmatrix} Q_i \\ V_i \end{bmatrix} u_i \\ y_i &= \begin{bmatrix} P_i & U_i \end{bmatrix} \begin{bmatrix} x_i^c \\ x_i^a \end{bmatrix} + D_i u_i, \end{aligned} \tag{2.15}$$

where x^c is the causal system state, x^a represents the anti-causal system state, u_i is the system input, and y_i is the system output. With zero initial system states and stack all the input and output as $\bar{u} = \begin{pmatrix} u_1^T, & u_2^T, & \dots & u_N^T \end{pmatrix}^T$, $\bar{y} =$

$\begin{pmatrix} y_1^T, & y_2^T, & \dots & y_N^T \end{pmatrix}^T$, the matrix $\mathcal{H}$ that describes the input-output behavior of this mixed-causal system, i.e., $\overline{y} = \mathcal{H}\overline{u}$, induces an SSS matrix structure. Take $N = 4$ for example, the matrix $\mathcal{H}$ is,

$$\mathcal{H} = \begin{bmatrix} D_i & U_1V_2 & U_1W_2V_3 & U_1W_2W_3V_4 \\ P_2Q_1 & D_2 & U_2V_3 & U_2W_3V_4 \\ P_3R_2Q_1 & P_3Q_2 & D_3 & U_3V_4 \\ P_4R_3R_2Q_1 & P_4R_3Q_2 & P_4Q_3 & D_4 \end{bmatrix}. \tag{2.16}$$

Using the LTV systems realization for SSS matrices, we have the following lemma that gives a direct link between LTV systems order and the semiseparable order.

Lemma 2.13 ([109]) *The lower and upper semiseparable order for an SSS matrix with minimal LTV system realization are* $\max\{l_i\}_{i=2}^{N}$ *and* $\max\{k_i\}_{i=1}^{M-1}$*, respectively. Here* $\{l_i\}_{i=2}^{M}$ *and* $\{k_i\}_{i=1}^{M-1}$ *are defined in Table 2.1.*

We describe the lemma in [109] more exactly by restricting the realization of an SSS matrix to be minimal in Lemma 2.13. It is not difficult to set an example of an SSS matrix with small semiseparable order, but its LTV systems realization is of bigger order. Lemma 2.13 states that the order of the causal LTV system is equal to the lower semiseparable order of an SSS matrix, while the order of the anti-causal LTV system is equal to the upper semiseparable order. Therefore, to reduce the semiseparable order of an SSS matrix is identical to reducing the order of its realization by mixed-causal LTV systems.

Model order reduction for LTV systems is studied in [30, 111]. In [111], a linear matrix inequality (LMI) approach was introduced to solve the Lyapunov inequalities to compute the controllability and observability gramians. In [30], the low-rank Smith method was presented to approximate the square-root of the controllability and observability gramians of LTV systems.

Since the causal LTV system and the anti-causal LTV system have similar structures that correspond to the strictly lower-triangular part and the strictly upper-triangular part of the matrix $\mathcal{H}$, respectively. We only consider the causal LTV system described by the following state-space model,

$$\begin{cases} x_{k+1} = R_k x_k + Q_k u_k \\ y_k = P_k x_k, \end{cases} \tag{2.17}$$

over the time interval $[k_o, k_f]$ with zero initial states. The controllability gramian $\mathcal{G}_c(k)$ and observability gramian $\mathcal{G}_o(k)$ are computed by the following Stein recurrence formulas:

$$\begin{aligned} \mathcal{G}_c(k+1) &= R_k \mathcal{G}_c(k) R_k^T + Q_k Q_k^T, \\ \mathcal{G}_o(k) &= R_k^T \mathcal{G}_o(k+1) R_k + P_k^T P_k, \end{aligned} \tag{2.18}$$

with initial conditions $\mathcal{G}_c(k_o) = 0$ and $\mathcal{G}_o(k_f + 1) = 0$.

Note that the controllability gramian $\mathcal{G}_c(k)$ and observability gramian $\mathcal{G}_o(k)$ are positive definite if the system is completely controllable and observable or semi-definite if the system is partially controllable and observable. Therefore, their eigenvalues are non-negative and often have a large jump at an early stage [6, 14]. This suggests to approximate these two Gramians at each step by a low-rank approximation. In this chapter, we just consider the case that the LTV systems are uniformly completely controllable and observable over the time interval, which means that the gramians $\mathcal{G}_c$ and $\mathcal{G}_o$ are positive definite. This is reasonable because the SSS matrices considered in this dissertation correspond to uniformly completely controllable and observable LTV systems.

Since the controllability gramian $\mathcal{G}_c(k)$ and observability gramian $\mathcal{G}_o(k)$ have similar structure, we will only use the controllability gramian $\mathcal{G}_c(k)$ to introduce the basic idea. The key to the low-rank approximation is to replace the factorization of the controllability gramian $\mathcal{G}_c(k)$

$$\mathcal{G}_c(k) = L_c(k)L_c^T(k), \tag{2.19}$$

where $L_c(k) \in \mathbb{R}^{M\times M}$ in each step k by its low-rank factorization,

$$\tilde{\mathcal{G}}_c(k) = \tilde{L}_c(k)\tilde{L}_c^T(k), \tag{2.20}$$

with $\tilde{L}_c(k) \in \mathbb{R}^{M\times m_k}$ where m_k is the ε-rank of $\mathcal{G}_c(k)$, and $m_k < M$. Here, the ε-rank of a matrix is defined by the number of singular values that are bigger than or equal to ε. Typically, m_k is set to be constant, i.e., $m_k = m$ at each step. Meanwhile, m_k can be also varying at different time steps. If $\mathcal{G}_c(k)$ is of low numerical rank, it is reasonable to use the rank m_k approximation (2.20) to approximate $\mathcal{G}_c(k)$.

Remark 2.14 The approximation of $\mathcal{G}_c(k)$ can be also performed by setting a threshold ε for the truncated singular values, which yields adaptive rank m_k. But special attention should be paid for the approximation of $\mathcal{G}_o(k)$ since m_k should be also used for the low-rank approximation of $\mathcal{G}_o(k)$. This is the guarantee for the existence of the contragredient transformation that is introduced by Lemma 2.16 such that the projection based balanced truncation for LTV systems can be performed.

Remark 2.15 Since the same rank should be used for the separate low-rank approximation of $\mathcal{G}_c(k)$ and $\mathcal{G}_o(k)$, if $\mathcal{G}_c(k)$ and $\mathcal{G}_o(k)$ are not well balanced, then there will be a relative big approximation error for the approximation of the Hankel map that will be introduced at (2.21) - (2.22). Here, "not well balanced" represents that the singular values for one matrix have a sharp decrease while not for the other.

In [29], the recursive low-rank gramians method was used to approximate the gramians of the linear time-invariant (LTI) systems. Such methods can also be applied to approximate the gramians of the LTV systems. This is studied by the same author in an earlier reference [31]. In this section, we study the connections between LTV systems and SSS matrices. Meanwhile, we extend the model order reduction algorithm for LTV systems to the model order reduction for SSS matrices. The

low-rank approximation method in [29, 31] was used to approximate the gramians of the LTV systems that the SSS matrix corresponds to and the approximate balanced truncation method was applied for the model order reduction. Even though the low-rank approximation method in this chapter and the one in [31] are quite similar, the novelty is that this algorithm has never been applied to reduce the rank of the off-diagonal blocks of structured matrices. Moreover, the low- rank approximation in [31] is more computationally expensive.

The low-rank approximation of the controllability and observability gramians for LTV systems is introduced in Algorithm 2.2.

Algorithm 2.2 Low-Rank Approximation of the Gramians for LTV Systems

procedure LOW_RANK($\mathcal{G}_c$, $\mathcal{G}_o$)
  Input: LTV system $\{P_k\}_{k=2}^{N}$, $\{R_k\}_{k=2}^{N-1}$, $\{Q_k\}_{k=1}^{N-1}$, reduced order $\{m_k\}_{k=2}^{N}$
  for $k = 2 : N$ **do**
    if $k == 2$ **then**
      $\left[\begin{array}{c} Q_{k-1} \end{array}\right] = U_c \Sigma_c V_c^T$ ▷ singular value decomposition
    else
      $\left[\begin{array}{c|c} Q_{k-1} & R_{k-1}\tilde{L}_c(k-1) \end{array}\right] = U_c \Sigma_c V_c^T$ (singular value decomposition)
    end if
    Partition $U_c = \left[\begin{array}{c|c} U_{c1} & U_{c2} \end{array}\right]$, $\Sigma_c = \left[\begin{array}{c|c} \Sigma_{c1} & \\ \hline & \Sigma_{c2} \end{array}\right]$, $U_{c1} \in \mathbb{R}^{M \times m_k}$, $\Sigma_{c1} \in \mathbb{R}^{m_k \times m_k}$.
    Let $\tilde{L}_c(k) = U_{c1}\Sigma_{c1} \in \mathbb{R}^{M \times m_k}$
  end for
  for $k = N : 2$ **do**
    if k==N **then**
      $\left[\begin{array}{c} P_k^T \end{array}\right] = U_o \Sigma_o V_o^T$ (singular value decomposition)
    else
      $\left[\begin{array}{c|c} P_k^T & R_k^T\tilde{L}_o(k+1) \end{array}\right] = U_o \Sigma_o V_o^T$ (singular value decomposition)
    end if
    Partition $U_o = \left[\begin{array}{c|c} U_{o1} & U_{o2} \end{array}\right]$, $\Sigma_o = \left[\begin{array}{c|c} \Sigma_{o1} & \\ \hline & \Sigma_{o2} \end{array}\right]$, $U_{o1} \in \mathbb{R}^{M \times m_k}$, $\Sigma_{o1} \in \mathbb{R}^{m_k \times m_k}$.
    Let $\tilde{L}_o(k) = U_{o1}\Sigma_{o1} \in \mathbb{R}^{M \times m_k}$
  end for
end procedure
Output: Approximated factors $\left\{\tilde{L}_c(k)\right\}_{k=2}^{N}$, $\left\{\tilde{L}_o(k)\right\}_{k=2}^{N}$

The balanced truncation approximates the LTV systems in the following way. The Hankel map, which maps the input from past to the output in the future, has the following definition for the LTV systems,

$$\mathcal{H}_k = \mathcal{O}_k \mathcal{C}_k, \tag{2.21}$$

where $\mathcal{O}_k$ and $\mathcal{C}_k$ are the state to outputs map, and input to state map at time instant k, respectively. Meanwhile, the following relation holds

$$\begin{aligned} \mathcal{G}_c(k) &= \mathcal{C}_k \mathcal{C}_k^T, \\ \mathcal{G}_o(k) &= \mathcal{O}_k^T \mathcal{O}_k, \end{aligned} \tag{2.22}$$

where in (2.22) $\mathcal{G}_c(k)$ and $\mathcal{G}_o(k)$ are the controllability gramian and observability gramian defined in (2.18), respectively.

The Hankel singular values $\sigma_{\mathcal{H}}$ are the singular values of the Hankel map, and it was computed via the following equations in the finite dimensional spaces.

$$\sigma_{\mathcal{H}_k}^2 = \lambda(\mathcal{H}_k^T \mathcal{H}_k) = \lambda(\mathcal{C}_k^T \mathcal{O}_k^T \mathcal{O}_k \mathcal{C}_k) = \lambda(\mathcal{C}_k \mathcal{C}_k^T \mathcal{O}_k^T \mathcal{O}_k) = \lambda(\mathcal{G}_c(k)\mathcal{G}_o(k)).$$

It was shown in [129] that for any two positive definite matrices, there always exits a so-called contragredient transformation such that

$$\Lambda_k = T_k^{-1} \mathcal{G}_c(k) T_k^{-T} = T_k^T \mathcal{G}_o(k) T_k, \tag{2.23}$$

where Λ_k is a diagonal matrix and

$$\Lambda_k = \operatorname{diag}(\lambda_{k_1},\ \lambda_{k_2},\ \cdots,\ \lambda_{k_M}). \tag{2.24}$$

With this contragredient transformation, we have

$$T_k^{-1} \mathcal{G}_c(k) \mathcal{G}_o(k) T_k = \Lambda_k^2. \tag{2.25}$$

This states that $\{\lambda_{k_i}\}_{i=1}^M$ are the Hankel singular values at time step k. Such contragredient transformation brings the systems into a "balanced" form, which means that the controllability gramian and observability gramian of the system are equal to a diagonal matrix. For the LTV system (2.17), after such a transformation, the balanced LTV system is,

$$\begin{cases} \bar{x}_{k+1} = T_{k+1}^{-1} R_k T_k \bar{x}_k + T_{k+1}^{-1} Q_k u_k \\ y_k = P_k T_k \bar{x}_k. \end{cases} \tag{2.26}$$

Since for system (2.26), the controllability and observability gramians are balanced, truncation can be performed to truncate the Hankel singular values that are below a set threshold. This could be done by using the left and right multipliers L_l and L_r that are defined by

$$L_l = \begin{bmatrix} I_{m_k} & 0 \end{bmatrix},\ L_r = \begin{bmatrix} I_{m_k} \\ 0 \end{bmatrix}, \tag{2.27}$$

where I_{m_k} is an $m_k \times m_k$ identity matrix and m_k is the reduced system dimension size at time step k. Then the reduced LTV system is

$$\begin{cases} \tilde{x}_{k+1} = \Pi_l(k+1)R_k\Pi_r(k)\tilde{x}_k + \Pi_l(k+1)Q_k u_k \\ y_k = P_k\Pi_r(k)\tilde{x}_k, \end{cases} \tag{2.28}$$

where $\tilde{x}_k = L_l\bar{x}_k$, $\Pi_l(k+1) = L_l T_{k+1}^{-1}$ and $\Pi_r(k) = T_k L_r$.

The reduced LTV system (2.28) is computed via a projection method with the projector defined by $\Pi(k) = \Pi_r(k)\Pi_l(k)$. This is because

$$\Pi_l(k)\Pi_r(k) = L_l T_k^{-1} T_k L_r = I_{m_k},$$

and

$$\Pi(k)^2 = \Pi_r(k)\Pi_l(k)\Pi_r(k)\Pi_l(k) = \Pi(k).$$

For the approximated gramians $\tilde{\mathcal{G}}_c(k)$ and $\tilde{\mathcal{G}}_o(k)$, which are positive semi-definite, we have the following lemma, which states that there also exists a contragredient transformation such that

$$\Lambda_k' = \bar{T}_k^{-1}\tilde{\mathcal{G}}_c(k)\bar{T}_k^{-T} = \bar{T}_k^T\tilde{\mathcal{G}}_o(k)\bar{T}_k, \tag{2.29}$$

where Λ_k' is a diagonal matrix and

$$\Lambda_k' = \mathrm{diag}(\lambda_{k_1}',\ \lambda_{k_2}',\ \cdots,\ \lambda_{k_m}',\ 0,\ \cdots,\ 0). \tag{2.30}$$

Lemma 2.16 ([129], Theorem 3) *Let symmetric positive semidefinite $Q,\ P \in \mathbb{R}^{M\times M}$ satisfy*

$$\mathit{rank}\ Q = \mathit{rank}\ P = \mathit{rank}\ QP = m,$$

where $m \le M$. Then there exists a nonsingular $W \in \mathbb{R}^{M\times M}$ (contragredient transformation) and positive definite diagonal $\Sigma \in \mathbb{R}^{m\times m}$ such that

$$Q = W\begin{bmatrix}\Sigma & 0\\ 0 & 0\end{bmatrix}W^T, \quad P = W^{-T}\begin{bmatrix}\Sigma & 0\\ 0 & 0\end{bmatrix}W^{-1}.$$

Proof: For the proof of the lemma, we refer to [129]. □

We have already explained that the diagonal entries of the matrix Λ_k' in (2.30) are the Hankel singular values of the approximate Hankel map in (2.21). If the controllability gramian $\mathcal{G}_c(k)$ and the observability gramian $\mathcal{G}_o(k)$ are well approximated by $\tilde{\mathcal{G}}_c(k)$ and $\tilde{\mathcal{G}}_o(k)$ separately, then $\tilde{\mathcal{G}}_c(k)\tilde{\mathcal{G}}_o(k)$ also approximates $\mathcal{G}_c(k)\mathcal{G}_o(k)$ well. This means that the approximate Hankel singular values $\{\lambda_{k_i}'\}_{i=1}^{m_k}$ are close to the original Hankel singular values $\{\lambda_{k_i}\}_{i=1}^{m_k}$.

In Algorithm 2.3, we show how to use the approximated gramians $\tilde{\mathcal{G}}_c(k)$ and $\tilde{\mathcal{G}}_o(k)$ to compute the reduced system. By using the approximated gramians, this method is called the approximate balanced truncation.

Algorithm 2.3 Approximate Balanced Truncation for LTV systems

procedure ABT(P, R, Q)
  Input: LTV system $\{P_k\}_{k=2}^{N}$, $\{R_k\}_{k=2}^{N-1}$, $\{Q_k\}_{k=1}^{N-1}$, reduced order $\{m_k\}_{k=2}^{N}$
  Apply Algorithm 2.2 to compute the approximated gramian factors $\left\{\tilde{L}_c(k)\right\}_{k=2}^{N}$ and $\left\{\tilde{L}_o(k)\right\}_{k=2}^{N}$
  for k=2 : N **do**
    Compute the singular value decomposition $\tilde{L}_c^T(k)\tilde{L}_o(k) = U_k\Sigma_k V_k^T$.
    Let $\Pi_l(k) = \Sigma_k^{-\frac{1}{2}} V_k^T \tilde{L}_o^T(k)$, and $\Pi_r(k) = \tilde{L}_c(k)U_k\Sigma_k^{-\frac{1}{2}}$
  end for
  for k=1 : N **do**
    if $k == 1$ **then**
      $\tilde{Q}_k = \Pi_l(k+1)Q_k$
    else if $k == N$ **then**
      $\tilde{P}_k = P_k\Pi_r(k)$
    else
      $\tilde{Q}_k = \Pi_l(k+1)Q_k$, $\tilde{R}_k = \Pi_l(k+1)R_k\Pi_r(k)$, $\tilde{P}_k = P_k\Pi_r(k)$
    end if
  end for
  Output: Reduced LTV system $\left\{\tilde{P}_k\right\}_{k=2}^{N}$, $\left\{\tilde{R}_k\right\}_{k=2}^{N-1}$, $\left\{\tilde{Q}_k\right\}_{k=1}^{N-1}$
end procedure

Lemma 2.17 *Algorithm 2.3 is a projection based approximate balanced truncation method for linear time-varying system* (2.17).

Proof: Here, we assume that the time-varying order before performing Algorithm 2.3 is uniform and equal to M. According to Lemma 2.16, there exists a contragredient transformation $\bar{T}_k \in \mathbb{R}^{M\times M}$ such that

$$\Lambda_k' = \bar{T}_k^{-1}\tilde{\mathcal{G}}_c(k)\bar{T}_k^{-T} = \bar{T}_k^T\tilde{\mathcal{G}}_o(k)\bar{T}_k,$$

where Λ_k' is a diagonal matrix and

$$\Lambda_k' = \begin{bmatrix}\Sigma_k & 0\\ 0 & 0\end{bmatrix}.$$

Here

$$\Sigma_k = \mathrm{diag}(\lambda_{k_1}',\ \lambda_{k_2}',\ \cdots,\ \lambda_{k_{m_k}}'),$$

and $\{\lambda'_{k_i}\}_{i=1}^{m_k}$ are the singular values of $\tilde{L}_c^T(K)\tilde{L}_o(k)$, i.e.,

$$\tilde{L}_c^T(K)\tilde{L}_o(k) = U_k \Sigma_k V_k^T.$$

According to the proof of Lemma 2.16, the contragredient transformation $\bar{T}_k$ are computed via

$$\bar{T}_k = \begin{bmatrix}\tilde{L}_c(k) & N_o(k)\end{bmatrix} \begin{bmatrix} U_k & 0 \\ 0 & \bar{V}_k \end{bmatrix} \begin{bmatrix} \Sigma_k & 0 \\ 0 & \bar{\Sigma}_k \end{bmatrix}^{-\frac{1}{2}}.$$

And the inverse of such transformation is computed by

$$\bar{T}_k^{-1} = \begin{bmatrix} \Sigma_k & 0 \\ 0 & \bar{\Sigma}_k \end{bmatrix}^{-\frac{1}{2}} \begin{bmatrix} V_k^T & 0 \\ 0 & \bar{U}_k^T \end{bmatrix} \begin{bmatrix} \tilde{L}_o(k)^T \\ N_c(k)^T \end{bmatrix},$$

where $N_o(k) \in \mathbb{R}^{M\times(M-m_k)}$ spans the null space of $\tilde{G}_o(k)$, $N_c(k) \in \mathbb{R}^{M\times(M-m_k)}$ spans the null space of $\tilde{G}_c(k)$, and we have the following singular value decomposition

$$\tilde{N}_c^T(k)\tilde{N}_o(k) = \bar{U}_k \bar{\Sigma}_k \bar{V}_k^T.$$

With this contragredient transformation $\bar{T}_k$, the left and right multipliers are computed via

$$\Pi_l(k) = \begin{bmatrix} I_{m_k} & 0 \end{bmatrix} T_k^{-1} = \begin{bmatrix} I_{m_k} & 0 \end{bmatrix} \begin{bmatrix} \Sigma_k & 0 \\ 0 & \bar{\Sigma}_k \end{bmatrix}^{-\frac{1}{2}} \begin{bmatrix} V_k^T & 0 \\ 0 & \bar{U}_k^T \end{bmatrix} \begin{bmatrix} \tilde{L}_o(k)^T \\ N_c(k)^T \end{bmatrix} = \Sigma_k^{-\frac{1}{2}} V_k^T \tilde{L}_o^T(k),$$

$$\Pi_r(k) = T_k \begin{bmatrix} I_{m_k} \\ 0 \end{bmatrix} = \begin{bmatrix}\tilde{L}_c(k) & N_o(k)\end{bmatrix} \begin{bmatrix} U_k & 0 \\ 0 & \bar{V}_k \end{bmatrix} \begin{bmatrix} \Sigma_k & 0 \\ 0 & \bar{\Sigma}_k \end{bmatrix}^{-\frac{1}{2}} \begin{bmatrix} I_{m_k} \\ 0 \end{bmatrix} = \tilde{L}_c(k) U_k \Sigma_k^{-\frac{1}{2}}.$$

The projection is defined via

$$\Pi_k = \Pi_r(k)\Pi_l(k),$$

since $\Pi_l(k)\Pi_r(k) = I_{m_k}$ and $\Pi_k^2 = \Pi_k$. □

Note that the low-rank approximation together with the approximate balanced truncation for linear time-invariant (LTI) system is studied in [28, 29]. Only balanced truncation for linear time-varying (LTV) system is studied in [31].

For an SSS matrix $A = \mathcal{SSS}(P_s,\ R_s,\ Q_s,\ D_s,\ U_s,\ W_s,\ V_s)$ with lower semiseparable order M, we have already explained its LTV system realization. Thus, Algorithm 2.2 and Algorithm 2.3 can be performed to reduce the order of the causal LTV system (2.17), which corresponds to reduce the lower semiseparable order. This yields the approximated SSS matrix $\tilde{A} = \mathcal{SSS}(\tilde{P}_s,\ \tilde{R}_s,\ \tilde{Q}_s,\ D_s,\ U_s,\ W_s,\ V_s)$. For the strictly upper-triangular part of A, we first transpose it to the strictly lower-triangular form then perform Algorithm 2.2 and Algorithm 2.3. After this reduction, we transpose the reduced strictly lower-triangular part to the strictly upper-triangular form.

2.4.3 Hankel Blocks Approximation

To compare with the approximate balanced truncation for SSS matrices, we describe the standard model order reduction algorithm in this part. It is called Hankel blocks approximation in [34, 33]. The Hankel blocks of an SSS matrix are defined by Definition 2.7.

Definition 2.7 ([33]) *Hankel blocks denote the off-diagonal blocks that extend from the diagonal to the northeast corner (for the upper case) or to the southwest corner (for the lower case).*

Take a 4×4 block SSS matrix A for example, the Hankel blocks for the strictly lower-triangular part are shown in Figure 2.2 by $\mathcal{H}_2$, $\mathcal{H}_3$ and $\mathcal{H}_4$.

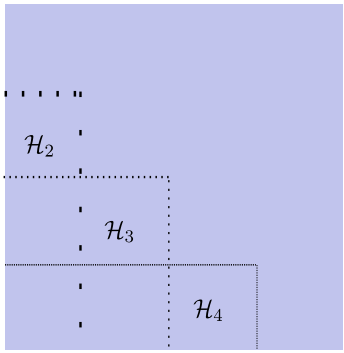


Figure 2.2: Hankel blocks of a 4×4 block SSS matrix

It is easy to very that for the Hankel blocks $\mathcal{H}_i$, $(i = 2, \ \ldots, \ N)$, the following relation hold

$$\mathcal{H}_i = \mathcal{O}_i \mathcal{C}_i, \ (i = 2, \ , \ldots, \ N), \tag{2.31}$$

where $\mathcal{O}_i$ and $\mathcal{C}_i$ are the current state to the current and future output map and the past input to the current state map for system (2.17), respectively. Moreover, the following relation hold for $\mathcal{O}_i$ and $\mathcal{C}_i$.

$$\mathcal{O}_{i-1} = \begin{bmatrix} P_{i-1} \\ \mathcal{O}_i R_{i-1} \end{bmatrix}, \ (i = 2, \ , \ldots, \ N-1), \ \mathcal{O}_N = P_N \tag{2.32}$$

$$\mathcal{C}_{i+1} = \begin{bmatrix} R_i \mathcal{C}_i & Q_i \end{bmatrix}, \ (i = 2, \ , \ldots, \ N-1), \ \mathcal{C}_2 = Q_1 \tag{2.33}$$

The two maps $\mathcal{C}_i$ and $\mathcal{O}_i$ also satisfy

$$\mathcal{G}_c(i) = \mathcal{C}_i \mathcal{C}_i^T, \ \mathcal{G}_o(i) = \mathcal{O}_i^T \mathcal{O}_i, \tag{2.34}$$

where $\mathcal{G}_c(i)$ and $\mathcal{G}_o(i)$ are the controllability gramian and observability gramian that satisfy (2.18).

The rank of the Hankel map $\mathcal{H}_i$ at time step i, i.e., the rank of the i-th Hankel block is the order of the states x_i of system (2.17) [125]. The standard way to reduce the rank of the Hankel blocks is given in [33]. This standard approach is based on the realization theory of a given Hankel map for LTV systems that is introduced in [41, 125], i.e., according to the given Hankel map $\{\mathcal{H}_i\}_{i=2}^{N}$, find a triple $\{P_k,\ R_k,\ Q_k\}$ that satisfy (2.31) (2.32) (2.33). By using the realization theory, it is also possible to get the reduced triple $\left\{\hat{P}_k,\ \hat{R}_k,\ \hat{Q}_k\right\}$ that approximates the Hankel map $\mathcal{H}_i$ in (2.31).

To do this approximation, first we need to transform the map $\mathcal{O}_i$ to the form that has orthonormal columns and transform the map $\mathcal{C}_i$ to the form that has orthonormal rows. These two forms are called the left proper form and the right proper form [34], respectively. Next, we use the transform of $\mathcal{C}_i$ to introduce the basic idea. The first step is to do a singular value decomposition (SVD) at the starting point $\mathcal{C}_2$, which gives

$$\mathcal{C}_2 = U_2\Sigma_2 V_2^T,$$

and let $\bar{\mathcal{C}}_2 = V_2^T$. At this step, the map $\mathcal{C}_2$ is transformed to the form $\bar{\mathcal{C}}_2$ that has orthonormal rows. Due to the change of $\mathcal{C}_2$, to keep the Hankel map $\mathcal{H}_i = \mathcal{O}_i\mathcal{C}_i$ unchanged, the map $\mathcal{O}_2$ is given by

$$\bar{\mathcal{O}}_2 = \mathcal{O}_2 U_2\Sigma_2,$$

then the Hankel map

$$\bar{\mathcal{H}}_2 = \bar{\mathcal{O}}_2\bar{\mathcal{C}}_2 = \mathcal{O}_2 U_2\Sigma_2 V_2^T = \mathcal{O}_2\mathcal{C}_2 = \mathcal{H}_2.$$

Since all these transformations have to be done on the triple $\{P_k,\ R_k,\ Q_k\}$, not on the maps, we have

$$\bar{Q}_1 = \bar{\mathcal{C}}_2 = V_2^T,$$

and

$$\bar{\mathcal{O}}_2 = \mathcal{O}_2 U_2\Sigma_2 = \begin{bmatrix} P_2 \\ \mathcal{O}_3 R_2 \end{bmatrix} U_2\Sigma_2,$$

which gives $\bar{P}_2 = P_2 U_2\Sigma_2$ and $\bar{R}_2 = R_2 U_2\Sigma_2$.

Now, suppose at step i, the map $\bar{\mathcal{C}}_i$ already has orthonormal rows, then for $\mathcal{C}_{i+1}$, we have

$$\mathcal{C}_{i+1} = \begin{bmatrix} \bar{R}_i\bar{\mathcal{C}}_i & Q_i \end{bmatrix} = \begin{bmatrix} \bar{R}_i & Q_i \end{bmatrix} \begin{bmatrix} \bar{C}_i & \\ & I \end{bmatrix}. \tag{2.35}$$

By performing a singular value decomposition to $\begin{bmatrix} \bar{R}_i & Q_i \end{bmatrix}$, we have

$$\begin{bmatrix} \bar{R}_i & Q_i \end{bmatrix} = U_i\Sigma_i V_i^T, \tag{2.36}$$

let $\begin{bmatrix}\bar{\bar{R}}_i & \bar{Q}_i\end{bmatrix} = V_i^T$ and partition V_i such that $V_i^T = \begin{bmatrix}V_{i_1}^T & V_{i_2}^T\end{bmatrix}$ to make the size of $V_{i_2}^T$ match the size of Q_i. Then let

$$\bar{Q}_i = V_{i_2}^T,\ \bar{\bar{R}}_i = V_{i_1}^T.$$

To keep the use of notations consistent, we reuse $\bar{R}_i$ to denote the transformed $\bar{R}_i$, i.e., $\bar{\bar{R}}_i$, this gives $\bar{R}_i = V_{i_1}^T$. By doing this, we have the transformed map

$$\bar{\mathcal{C}}_{i+1} = \begin{bmatrix}\bar{R}_i\bar{\mathcal{C}}_i & \bar{Q}_i\end{bmatrix} = \begin{bmatrix}\bar{R}_i & \bar{Q}_i\end{bmatrix}\begin{bmatrix}\bar{\mathcal{C}}_i & \\ & I\end{bmatrix} = V_i^T\begin{bmatrix}\bar{\mathcal{C}}_i & \\ & I\end{bmatrix}, \tag{2.37}$$

which also has orthonormal rows. This is due to

$$\begin{aligned}\bar{\mathcal{C}}_{i+1}\bar{\mathcal{C}}_{i+1}^T &= V_i^T\begin{bmatrix}\bar{\mathcal{C}}_i & \\ & I\end{bmatrix}\begin{bmatrix}\bar{\mathcal{C}}_i^T & \\ & I\end{bmatrix}V_i \\ &= V_i^T\begin{bmatrix}\bar{\mathcal{C}}_i\bar{\mathcal{C}}_i^T & \\ & I\end{bmatrix}V_i = V_i^TV_i = I,\end{aligned}$$

since $\bar{\mathcal{C}}_i$ also has orthonormal rows. Then the Hankel map at time step $i+1$ before and after such transformation has the following relation,

$$\mathcal{C}_{i+1} = U_i\Sigma_i\bar{\mathcal{C}}_{i+1}, \tag{2.38}$$

which can be checked by associating (2.35) and (2.36) with (2.37).

To keep the Hankel map at time step $i+1$ unchanged, the following relation needs to hold,

$$\bar{\mathcal{O}}_{i+1} = \mathcal{O}_{i+1}U_i\Sigma_i. \tag{2.39}$$

Since $\mathcal{O}_{i+1} = \begin{bmatrix}P_{i+1} \\ \mathcal{O}_{i+2}R_{i+1}\end{bmatrix}$, by letting $\bar{P}_{i+1} = P_{i+1}U_i\Sigma_i$ and $\bar{R}_{i+1} = R_{i+1}U_i\Sigma_i$, we have the transformed map

$$\bar{\mathcal{O}}_{i+1} = \begin{bmatrix}\bar{P}_{i+1} \\ \mathcal{O}_{i+2}\bar{R}_{i+1}\end{bmatrix}. \tag{2.40}$$

And by checking (2.38)(2.39)(2.40), it is easy to get the unchanged Hankel map at time step $i+1$. Similar procedure can be applied to transform the map $\mathcal{O}_i$ to the form that has orthonormal columns.

After transforming the map $\mathcal{O}_i$ and $\mathcal{C}_i$ into the form with orthogonal column basis and row basis, we start to transform $\mathcal{C}_i$ to the form with orthonormal columns and compute the approximate Hankel map (blocks) $\hat{\mathcal{H}}_i = \hat{\mathcal{O}}_i\hat{\mathcal{C}}_i$. Therefore, this method is called the Hankel blocks approximation.

Remark 2.18 For the approximate balanced truncation,the map $\mathcal{O}_i$ and $\mathcal{C}_i$ are approximated separately via the low-rank approximation Algorithm 2.2. If the maps are not well balanced, the Hankel map $\mathcal{H}_i$ is not well approximated. We will introduce the error bound for the Hankel blocks approximation in Chapter 4.

Algorithm 2.4 Hankel Blocks Approximation

procedure HBA(P, R, Q)
  Input:LTV system $\{P_k\}_{k=2}^{N}$, $\{R_k\}_{k=2}^{N-1}$, $\{Q_k\}_{k=1}^{N-1}$, reduced system order m
  for $i = 2 : N$ **do**
    if $i == 2$ **then**
      $Q_{i-1} = U_i \Sigma_i V_i^T$ (singular value decomposition)
      Let $Q_{i-1} = V_i^T$, $P_i = P_i U_i \Sigma_i$, and $R_i = R_i U_i \Sigma_i$
    else if $i == N$ **then**
      $\begin{bmatrix} R_{i-1} & Q_{i-1} \end{bmatrix} = U_i \Sigma_i V_i^T$ (singular value decomposition)
      Partition $V_i^T = \begin{bmatrix} V_{i_1}^T & V_{i_2}^T \end{bmatrix}$ such that the size of Q_{i-1} and $V_{i_2}^T$ match
      Let $R_{i-1} = V_{i_1}^T$, $Q_{i-1} = V_{i_2}^T$, $P_i = P_i U_i \Sigma_i$
    else
      $\begin{bmatrix} R_{i-1} & Q_{i-1} \end{bmatrix} = U_i \Sigma_i V_i^T$ (singular value decomposition)
      Partition $V_i^T = \begin{bmatrix} V_{i_1}^T & V_{i_2}^T \end{bmatrix}$ such that the size of Q_{i-1} and $V_{i_2}^T$ match
      Let $R_{i-1} = V_{i_1}^T$, $Q_{i-1} = V_{i_2}^T$, $P_i = P_i U_i \Sigma_i$ and $R_i = R_i U_i \Sigma_i$
    end if
  end for
  for $i = N : 2$ **do**
    if $i == N$ **then**
      $P_i = U_i \Sigma_i V_i^T$ (singular value decomposition)
      Let $P_i = U_i$, $R_{i-1} = \Sigma_i V_i^T R_{i-1}$, $Q_{i-1} = \Sigma_i V_i^T Q_{i-1}$
    else if $i == 2$ **then**
      $\begin{bmatrix} P_i \\ R_i \end{bmatrix} = U_i \Sigma_i V_i^T$ (singular value decomposition)
      Partition $U_i = \begin{bmatrix} U_{i_1} \\ U_{i_2} \end{bmatrix}$ such that the size of U_{i_2} and R_i match
      Let $P_i = U_{i_1}$, $R_i = U_{i_2}$, $Q_{i-1} = \Sigma_i V_i Q_{i-1}$
    else
      $\begin{bmatrix} P_i \\ R_i \end{bmatrix} = U_i \Sigma_i V_i^T$ (singular value decomposition)
      Partition $U_i = \begin{bmatrix} U_{i_1} \\ U_{i_2} \end{bmatrix}$ such that the size of U_{i_2} and R_i match
      Let $P_i = U_{i_1}$, $R_i = U_{i_2}$, $Q_i = \Sigma_i V_i Q_i$
    end if
  end for

Algorithm 2.4 Hankel Blocks Approximation (continued)

for $i = 1 : N$ **do**
  if $i == 1$ **then**
    Partition $Q_i = \left[\begin{array}{c} Q_{i_1} \\ \hline (\cdot) \end{array}\right]$ with $Q_{i_1} \in \mathbb{R}^{m\times(\cdot)}$, let $\hat{Q}_i = Q_{i_1}$
  else if i==N **then**
    Partition $P_i = \left[\begin{array}{c} P_{i_1} \\ \hline (\cdot) \end{array}\right]$ with $P_{i_1} \in \mathbb{R}^{(\cdot)\times m}$, let $\hat{P}_i = P_{i_1}$
  else
    Partition $P_i = \left[\begin{array}{c} P_{i_1} \\ \hline (\cdot) \end{array}\right]$ with $P_{i_1} \in \mathbb{R}^{(\cdot)\times m}$, let $\hat{P}_i = P_{i_1}$
    Partition $R_i = \left[\begin{array}{c|c} R_{i_1} & (\cdot) \\ \hline (\cdot) & (\cdot) \end{array}\right]$ with $R_{i_1} \in \mathbb{R}^{m\times m}$, let $\hat{R}_i = R_{i_1}$
    Partition $Q_i = \left[\begin{array}{c} Q_{i_1} \\ \hline (\cdot) \end{array}\right]$ with $Q_{i_1} \in \mathbb{R}^{m\times(\cdot)}$, let $\hat{Q}_i = Q_{i_1}$
  end if
end for
end procedure

Output: Reduced LTV systems $\left\{\hat{P}_k\right\}_{k=2}^{N}$, $\left\{\hat{R}_k\right\}_{k=2}^{N-1}$, $\left\{\hat{Q}_k\right\}_{k=1}^{N-1}$

2.4.4 Operations Count for the Two Model Order Reduction Methods

Given an SSS matrix $A = \mathcal{SSS}(P_S,\ R_s,\ Q_s,\ D_s,\ U_s,\ W_s,\ V_s)$, to compare the operations count of the approximate balanced truncation described by Algorithm 2.2 - 2.3 and the Hankel blocks approximation introduced in Algorithm 2.4, we assume that the generators sizes in Table 2.1 are uniform, i.e., $m_i = n$ and $k_i = l_i = M$. Here N is the number of SSS blocks (LTV system time steps), M is the unreduced LTV system order, and $N \gg M \gg n$. For the reduced SSS matrix $\tilde{A} = \mathcal{SSS}(\hat{P}_s,\ \hat{R}_s,\ \hat{Q}_s,\ D_s,\ \hat{U}_s,\ \hat{W}_s,\ \hat{V}_s)$, let $\hat{k}_i = \hat{l}_i = m$, m is the reduced semiseparable order and $m \ll M$.

In this section, we measure the operations count by the floating-point operations (flops). To compute the operations count of the approximate balanced truncation, first we compute the operations count for the low-rank approximation in Algorithm 2.2. In the forward recursion, the flops count for the singular value decomposition is

$$m^2M + (m+n)^2M(N-2). \tag{2.41}$$

In this recursion, two matrix-matrix product are computed in each iteration, where the flops count is

$$mM^2(N-2) + m^2M(N-1). \tag{2.42}$$

Adding (2.41) and (2.42) gives the total flops count for the forward low-rank ap-

proximation,

$$m^2M + (m+n)^2M(N-2) + mM^2(N-2) + m^2M(N-1), \tag{2.43}$$

Since the forward low-rank approximation and the backward low-rank approximation are symmetric in computing, the flops count for the backward low-rank approximation is equal to (2.43). Then the flops count $\mathcal{F}_l$ for the low-rank approximation Algorithm 2.2 is

$$\mathcal{F}_l = 2mM^2N + 4m^2MN + 4mnMN + 2n^2MN - 4(m+n)^2M - 4mM^2. \tag{2.44}$$

Next we compute the operations count of the approximate balanced truncation Algorithm 2.3. First, to compute $\Pi_l(k)$ and $\Pi_r(k)$, the flops count is

$$\left(m^2M + m^3 + 2(m^2 + m^2M)\right)(N-1), \tag{2.45}$$

and the flops count to compute the reduced LTV system is

$$2mnM(N-1) + (mM^2 + m^2M)(N-2). \tag{2.46}$$

Thus the total flops count $\mathcal{F}_a$ for the approximate balanced truncation is the summation of (2.45) and (2.46), which is given by

$$\mathcal{F}_a = (M^2 + mM + 2nM)N - 2mn(M+m+n). \tag{2.47}$$

Then we have the total flops count $\mathcal{F}_{la}$ of the approximate balanced truncation by adding (2.44) to (2.47). Since we have $N \gg M \gg m,\ n$, we just use the $\mathcal{O}(\cdot)$ to denote the total flops count. Then, we get

$$\mathcal{F}_{la} = \mathcal{O}\left((2m+1)M^2N\right). \tag{2.48}$$

Similarly, we can compute the operations count $\mathcal{F}_h$ for the Hankel blocks approximation in Algorithm 2.4, which is given by

$$\mathcal{F}_h = 4M^3N + 6nM^2N + 2(n^2+2n)MN - 8M^3 - 12nM^2 + 2(n^2-3n)M + 2n^2, \tag{2.49}$$

and by using the $\mathcal{O}(\cdot)$ notation, we can write the flops count of the Hankel blocks approximation method as

$$\mathcal{F}_h = \mathcal{O}(4M^3N). \tag{2.50}$$

Since $N \gg M \gg m$, by comparing the flops count $\mathcal{F}_{la}$ for the approximate balanced truncation in (2.48) with the flops count $\mathcal{F}_h$ for the Hankel blocks approximation in (2.49), we see that the approximate balanced truncation algorithm is computationally cheaper than the Hankel blocks approximation for the model order reduction of SSS matrices.

Remark 2.19 By checking the flops count $\mathcal{F}_{la}$ for the approximate balanced truncation in (2.48) with the flops count $\mathcal{F}_h$ for the Hankel blocks approximation in

(2.49), we can see that the flops count is linear with N for both method, where N denotes the number of blocks of an SSS matrix. Moreover, the size of the SSS matrix equals to nN and $n \ll N$. Thus, both methods have computational complexity that is linear with the matrix size.

2.4.5 Flowchart of Preconditioning by Using MSSS Matrix Computations

We have already described the MSSS matrix computations and showed how to compute a preconditioner using such matrix computations. In this part, we use a flowchart to illustrate how to compute a preconditioner for the PDE-constrained optimization problem (2.1). This flowchart is shown in Figure 2.3.

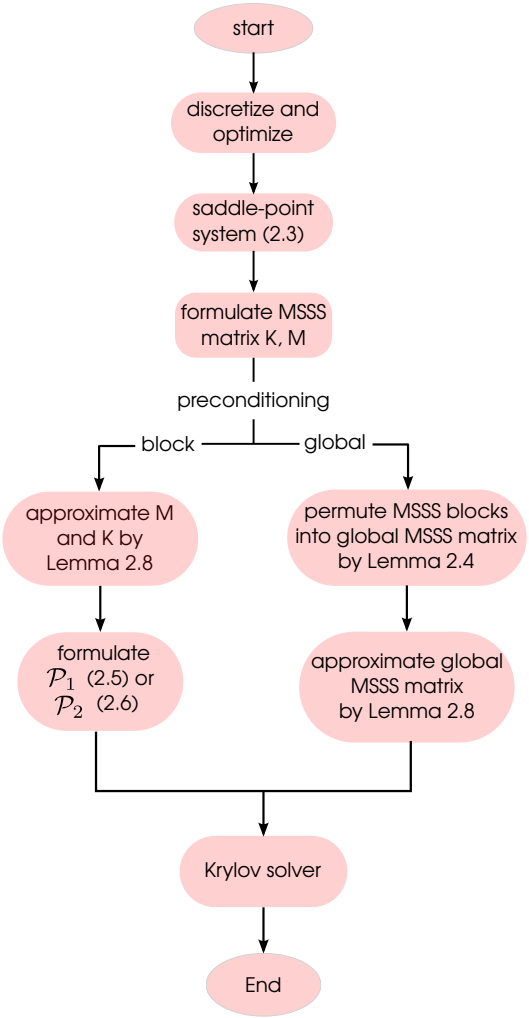


Figure 2.3: Flowchart for MSSS preconditioning of PDE-constrained optimization problem

2.5 Numerical Experiments

In this section, we study the problem of optimal control of the convection-diffusion equation that is introduced in Example 2.2. First, we compare the performance of our model order reduction algorithm with the conventional model order reduction algorithm. Next we test the global MSSS preconditioner and the block diagonal MSSS preconditioner. Numerical results in the appendix also show the advantage of the global MSSS preconditioner over the lower-triangular block MSSS preconditioner for the PDE-constrained optimization problem. The superiority of the global MSSS preconditioner to the block preconditioners that are computed by the multigrid methods for computational fluid dynamics (CFD) problems is illustrated in the next chapter.

Example 2.2 ([106]) *Let* $\Omega = \{(x,y)|0 \le x \le 1, 0 \le y \le 1\}$ *and consider the problem*

$$\begin{aligned} \min_{u,f} \frac{1}{2}\|u-\hat{u}\| + \frac{\beta}{2}\|f\|^2 \\ s.t. \ -\nu\nabla^2 u + \overrightarrow{\omega}\cdot\nabla u = f \quad in \quad \Omega \\ u = u_D \quad on \quad \partial\Omega, \end{aligned}$$

where $\partial\Omega$ *is the boundary of* Ω*, and*

$$u_D = \begin{cases} (2x-1)^2(2y-1)^2 & if \ \ 0 \le x \le \frac{1}{2}, \ and \ \ 0 \le y \le \frac{1}{2}, \\ 0 & otherwise. \end{cases}$$

ν *is a positive scalar,* $\overrightarrow{\omega} = (cos(\theta),\ sin(\theta))^T$ *is the unit directional vector and the prescribed state* $\hat{u} = 0$*.*

The numerical experiments are performed on a laptop of Intel Core 2 Duo P8700 CPU of 2.53 GHz and 8Gb memory with Matlab R2010b. The iterative solver is stopped by either reducing the 2-norm of the residual by a factor of 10^{-6} or reaching the maximum number of iterations that is set to be 100 in this section. Note that there are three unknowns on each grid point. The problem sizes 3.07e+03,1.23e+04, 4.92e+04 and 1.97e+05 correspond to the mesh sizes $h = 2^{-5}$, 2^{-6}, 2^{-7}, and 2^{-8}, respectively. The maximum semiseparable order for the model order reduction are given in the brackets following the problem sizes. The "preconditioning" columns in the tables report the time to compute the preconditioner while the "MINRES" or "IDR(s)" columns give the time to solve the saddle-point problem by using such Krylov solver, and the "total" columns is the summation of the time to compute the preconditioner and iteratively solve the saddle-point system. All the times are measured in seconds.

2.5.1 Comparison of Two Model Order Reduction Algorithms

In this part, we test the performance of the two model order reduction algorithms. Consider the preconditioning of optimal control of the convection-diffusion equation described by Example 2.2. For the block-diagonal preconditioner $\mathcal{P}_1$ that is computed by the approximate balanced truncation algorithm and the Hankel blocks approximation method, the results for different ν and β are shown in Table 2.2 - 2.9 while θ is set to be $\pi/5$ for all the experiments.

Table 2.2: Results for approximate balanced truncation for $\beta = 10^{-1}$, $\nu = 10^{-1}$

problem size	iterations	preconditioning (sec.)	MINRES (sec.)	total (sec.)
3.07e+03 (4)	10	0.43	0.88	1.31
1.23e+04 (6)	10	1.79	2.07	3.86
4.92e+04 (6)	10	4.11	5.95	10.06
1.97e+05 (7)	10	17.05	22.09	39.14

Table 2.3: Results for Hankel blocks approximation for $\beta = 10^{-1}$, $\nu = 10^{-1}$

problem size	iterations	preconditioning (sec.)	MINRES (sec.)	total (sec.)
3.07e+03 (4)	10	0.69	1.32	2.01
1.23e+04 (6)	10	2.59	2.38	4.97
4.92e+04 (6)	10	6.14	5.94	12.08
1.97e+05 (7)	10	26.11	21.59	47.70

Table 2.4: Results for approximate balanced truncation for $\beta = 10^{-1}$, $\nu = 10^{-2}$

problem size	iterations	preconditioning (sec.)	MINRES (sec.)	total (sec.)
3.07e+03 (3)	16	0.29	1.46	1.75
1.23e+04 (4)	14	0.96	3.01	3.97
4.92e+04 (4)	14	2.49	8.17	10.66
1.97e+05 (5)	14	9.43	29.57	39.00

Table 2.5: Results for Hankel blocks approximation for $\beta = 10^{-1}$, $\nu = 10^{-2}$

problem size	iterations	preconditioning (sec.)	MINRES (sec.)	total (sec.)
3.07e+03 (3)	16	0.46	1.48	1.94
1.23e+04 (4)	14	1.40	2.98	4.38
4.92e+04 (4)	14	4.85	7.99	12.84
1.97e+05 (5)	14	20.48	28.24	48.72

Table 2.6: Results for approximate balanced truncation for $\beta = 10^{-2}$, $\nu = 10^{-1}$

problem size	iterations	preconditioning (sec.)	MINRES (sec.)	total (sec.)
3.07e+03 (3)	18	0.28	1.59	1.87
1.23e+04 (3)	18	0.85	4.02	4.87
4.92e+04 (3)	18	2.26	10.79	13.05
1.97e+05 (5)	18	9.67	35.32	44.99

Table 2.7: Results for Hankel blocks approximation for $\beta = 10^{-2}$, $\nu = 10^{-1}$

problem size	iterations	preconditioning (sec.)	MINRES (sec.)	total (sec.)
3.07e+03 (3)	18	0.47	1.65	2.12
1.23e+04 (3)	18	1.28	3.95	5.23
4.92e+04 (3)	18	4.41	10.38	14.79
1.97e+05 (5)	18	21.14	35.12	56.26

Table 2.8: Results for approximate balanced truncation for $\beta = 10^{-2}$, $\nu = 10^{-2}$

problem size	iterations	preconditioning (sec.)	MINRES (sec.)	total (sec.)
3.07e+03 (3)	30	0.32	2.54	2.86
1.23e+04 (3)	30	0.81	6.04	6.85
4.92e+04 (3)	30	2.28	17.79	20.07
1.97e+05 (5)	30	9.42	58.01	67.43

Table 2.9: Results for Hankel blocks approximation for $\beta = 10^{-2}$, $\nu = 10^{-2}$

problem size	iterations	preconditioning (sec.)	MINRES (sec.)	total (sec.)
3.07e+03 (3)	30	0.49	2.62	3.11
1.23e+04 (3)	30	1.42	6.08	7.50
4.92e+04 (3)	30	4.46	17.43	21.89
1.97e+05 (5)	30	20.39	57.32	77.71

The results in Table 2.2 - 2.9 show that the time to compute the preconditioner and iteratively solve the saddle-point system is linear in the problem size, which verifies that the MSSS preconditioning technique has linear computational complexity. It shows that for the same group of ν and β, the block MSSS preconditioners computed by the approximate balanced truncation and Hankel blocks approximation methods give mesh size independent convergence. Moreover, the number of itera-

tions for the block MSSS preconditioners computed by both model order reduction algorithms are the same.

Remark 2.20 As shown by (2.48) and (2.49), the approximate balanced truncation is computationally cheaper than the Hankel blocks approximation and both algorithms have linear computational complexity. This is illustrated by the time to compute the preconditioners by these two methods for the same group of β and ν in Table 2.2 - 2.9.

2.5.2 Comparison of Preconditioners

In this part, we test the performance of the block-diagonal MSSS preconditioner and the global MSSS preconditioner. For the block diagonal MSSS preconditioner, from Table 2.2 - 2.9 we have seen that with the decrease of β, the number of iterations increases slightly for the same problem size and ν. This is due to the $^1/_{2\beta}M$ term, which plays an increasingly important role with the decrease of β, while this term is often neglected in the preconditioner $\mathcal{P}_1$ in (2.5) for big and middle value of β [106]. If we continue decreasing β, we obtain the computational results for the block-diagonal MSSS preconditioner in Table 2.10-2.11. For the preconditioners tested in this part, the Hankel blocks approximation method is chosen as the model order reduction algorithm. Results for the preconditioners computed by the approximate balanced truncation can be found in [102].

Table 2.10: Results for the block-diagonal MSSS preconditioner (2.5) for $\beta = 10^{-3}$, $\nu = 10^{-1}$

problem size	iterations	preconditioning (sec.)	MINRES (sec.)	total (sec.)
3.07e+03 (3)	34	0.43	2.91	3.34
1.23e+04 (3)	34	1.31	7.61	8.92
4.92e+04 (3)	34	4.26	19.83	24.09
1.97e+05 (5)	34	17.39	61.82	79.21

Table 2.11: Results for the block-diagonal MSSS preconditioner (2.5) for $\beta = 10^{-4}$, $\nu = 10^{-1}$

problem size	iterations	preconditioning (sec.)	MINRES (sec.)	total (sec.)
3.07e+03 (3)	82	0.45	4.91	5.36
1.23e+04 (3)	82	1.31	11.91	13.22
4.92e+04 (3)	80	4.34	34.83	39.17
1.97e+05 (5)	80	17.89	133.28	141.17

As shown in Table 2.10 - 2.11, with the decrease of β from 10^{-3} to 10^{-4}, the number of iterations more than doubles. Clearly if β is too small, the block-diagonal MSSS

preconditioner $\mathcal{P}_1$ cannot give satisfactory results. Alternatively, for small β, we can choose the block-diagonal MSSS preconditioner as follows

$$\mathcal{P}_1 = \begin{bmatrix} 2\beta\hat{M} & & \\ & \hat{M} & \\ & & -\frac{1}{2\beta}\hat{M} \end{bmatrix}. \tag{2.51}$$

The computational results of the preconditioner (2.51) for $\beta = 10^{-4}$ are given in Table 2.12. Here, we approximate the mass matrix M by performing approximate LU factorization using MSSS matrix computations.

Table 2.12: Results for the block-diagonal MSSS preconditioner (2.51) for $\beta = 10^{-4}$, $\nu = 10^{-1}$

problem size	iterations	preconditioning (sec.)	MINRES (sec.)	convergence
3.07e+03 (5)	100	0.35	6.73	no convergence
1.23e+04 (5)	100	1.17	17.97	no convergence
4.92e+04 (5)	100	4.19	44.93	no convergence
1.97e+05 (5)	100	15.72	156.89	no convergence

The results in Table 2.11 - 2.12 show that the block-diagonal MSSS preconditioner does not give satisfactory performance when β becomes so small. Here in the table, "no convergence" means that the 2-norm of the residual does not converge to desired accuracy within 100 iterations. This is due to the fact that $^1/_{2\beta}M$ cannot approximate the Schur complement efficiently for small β.

Recall from Section 2.3 that we can permute the saddle-point system with MSSS blocks into a global MSSS system. Due to the indefiniteness of the global MSSS preconditioner, `MINRES` is not suitable to iteratively solve the preconditioned saddle-point system, the induced dimension reduction (`IDR(s)`) method [127] is chosen as the Krylov solver. To compare with the results for the block-diagonal MSSS preconditioner in Table 2.10 - 2.12, we apply the global MSSS preconditioner to the same test case. The results are given in Table 2.13 - 2.14.

Table 2.13: Results for the global MSSS preconditioner for $\beta = 10^{-3}$ and $\nu = 10^{-1}$

problem size	iterations	preconditioning (sec.)	IDR(4) (sec.)	total (sec.)
3.07e+03 (4)	2	0.38	0.13	0.51
1.23e+04 (6)	2	1.56	0.24	1.80
4.92e+04 (8)	2	5.46	0.66	6.12
1.97e+05 (10)	2	21.29	2.21	23.50

Even though it takes slightly longer time to compute the global MSSS preconditioner than to compute the block-diagonal MSSS preconditioner, much less time

is needed for the `IDR(s)` method to solve the preconditioned system by the global MSSS preconditioner. Meanwhile, the time to compute both preconditioners and to solve the preconditioned system scales linearly with the problem size.

Table 2.14: Results for the global MSSS preconditioner for $\beta = 10^{-4}$ and $\nu = 10^{-1}$

problem size	iterations	preconditioning (sec.)	IDR(4) (sec.)	total (sec.)
3.07e+03 (4)	2	0.38	0.13	0.51
1.23e+04 (6)	2	1.55	0.24	1.79
4.92e+04 (7)	2	5.23	0.64	5.87
1.97e+05 (9)	2	21.87	2.18	24.05

Remark 2.21 By comparing the computational results of the global MSSS preconditioner with that of the block-diagonal MSSS preconditioner, we find that for the same numerical test with the same group of β and ν that the number of iterations is reduced significantly by the global MSSS preconditioner. Meanwhile, the global MSSS preconditioner gives both mesh size and β independent convergence. This makes the global MSSS preconditioner superior to the block preconditioners.

Remark 2.22 Although recent advance in block preconditioning techniques also has β independent convergence property [96], the block preconditioner developed in [96] failed to solve PDE-constrained optimization problems where control only act on parts of the domain, cf. [93, 107]. The block preconditioner we used for comparison with global MSSS preconditioner in this chapter still suits that type problem and is more general. Moreover, for the case of in-domain control of Navier-Stokes equation that is described in Chapter 5, both block preconditioning techniques introduced in this chapter and [96] fail to give satisfactory performance while global MSSS preconditioning technique is still robust and efficient. Details will be discussed in Chapter 5.

2.6 Optimal Control of 3D Problems

As analyzed in Section 2.4.1, to do an LU factorization of a k-level SSS matrix, the model order reduction of a $(k-1)$-level SSS matrix is needed. Therefore, to compute a preconditioner for 3D problems using MSSS matrix computations, model order reduction for 2-level SSS matrices is needed. Since the model order reduction for 2 and higher level SSS matrices is still an open problem, only preliminary results for optimal control of the 3D Poisson equation in Example 2.3 are given in this section.

Example 2.3 *Consider the following problem of optimal control of the 3D Poisson*

equation

$$\begin{aligned} \min_{u,f} \frac{1}{2}\|u-\hat{u}\| + \frac{\beta}{2}\|f\|^2 \\ s.t. -\nabla^2 u = f \ in\ \Omega \\ u = u_D \ on\ \partial\Omega, \end{aligned} \tag{2.52}$$

where $\Omega = \{(x,y,z)|0 \le x \le 1, 0 \le y \le 1, 0 \le z \le 1\}$ *and*

$$u_D = \begin{cases} \sin(2\pi y), & \text{if } x = 0,\ 0 \le y \le 1,\ z = 0; \\ -\sin(2\pi y), & \text{if } x = 1,\ 0 \le y \le 1,\ z = 0; \\ 0, & \text{elsewhere.} \end{cases}$$

The discretized analog of problem (2.52) is

$$\begin{aligned} \min_{u,\ f}\ \frac{1}{2}\|u-\hat{u}\|^2 + \beta\|f\|^2 \\ s.t.\ Ku = Mf + d, \end{aligned} \tag{2.53}$$

where

$$K = \begin{bmatrix} D & -L & & & \\ -L & D & -L & & \\ & -L & D & \ddots & \\ & & \ddots & \ddots & -L \\ & & & -L & D \end{bmatrix}, \tag{2.54}$$

the matrices D and L in K are 2-level SSS matrices, M is the 3D mass matrix that has the same structure as K, d is a vector that corresponds to the given boundary condition. To compute the optimal solution of Example 2.3, system of the form (2.3) needs to be solved. Here we again study two types of preconditioners: the block-diagonal MSSS preconditioner and the global MSSS preconditioner.

2.6.1 Block-Diagonal Preconditioner

In this subsection, we test the block-diagonal preconditioner for big and middle β, then the block-diagonal preconditioner $\mathcal{P}_1$ (2.5) is chosen. Here the Schur complement is approximated by $\hat{K}M^{-1}\hat{K}^T$ where $\hat{K}$ is an approximation of K by MSSS matrix computations.

To approximate the symmetric positive definite matrix K, we can compute its approximate Cholesky factorization with MSSS matrix computations. At the k-th step of the Cholesky factorization, the Schur complement is computed via

$$\begin{cases} S_k = D, & \text{if } k = 1, \\ S_k = D - LS_{k-1}^{-1}L, & \text{if } k \ge 2. \end{cases} \tag{2.55}$$

Since D and L are 2-level SSS matrices, S_k is also a 2-level SSS matrix. In the recurrence (2.55), both the 2-level and 1-level semiseparable orders of S_k increase as k goes up. Model order reduction for 2-level and 1-level SSS matrices are necessary, of which the model order reduction for 2-level SSS matrix is still an open problem. Here we use an alternative approach to approximate the Schur complement with lower 2-level semiseparable order.

As pointed out in [35], for a symmetric positive definite matrix from the discretization of PDEs with constant coefficients, all its subsequent Schur complements are also symmetric positive definite and converge to a fixed point matrix S_∞ with a fast rate. In [40], Dewilde et al. used a hierarchical partitioning for the 3D matrix K (2.54) and did computations on 2D matrices using the 1-level SSS matrix computations for preconditioning 3D Poisson equation on an $8 \times 8 \times 8$ regular grid. Due to the fact that 1-level SSS matrix computations were performed on 2D matrices, the linear computational complexity is lost. Note that there is no numerical experiment in [40] to study the performance of such preconditioning technique for a certain Krylov solver.

In this section, we extend the method in [40] to the optimal control of 3D Poisson equation in the following ways. Instead of using the hierarchical partitioning of a 3D matrix, we use the 3-level SSS matrix formulation. This avoids cutting on "layers" that is introduced in [40] to bound the 2-level semiseparable order. We exploit the fast convergence property of the Schur complements of symmetric positive definite matrices to bound the 2-level semiseparable order. As analyzed in Section 2.3, the 1-level and 2-level semiseparable order both grow in computing the Schur complements in (2.55). To reduce the 1-level semiseparable order, we can apply the approximate balanced truncation or the Hankel blocks approximation that are introduced in Section 2.4. We use an alternative approach to bound the 2-level semiseparable order. We compute the Schur complements of the first k_r steps using MSSS matrix computations. Here k_r determines the accuracy of the approximate block factorization. By balancing the accuracy of the preconditioner and computational cost, we usually choose k_r as a small number. Using the Schur complement at step k_r to replace the Schur complements, we have the following recursions for the Schur complements

$$\begin{cases} S_k = D, & \text{if } k = 1, \\ S_k = D - LS_{k-1}^{-1}L, & \text{if } 2 \le k \le k_r, \\ S_k = S_{k_r}, & \text{if } k > k_r. \end{cases} \tag{2.56}$$

Since only the Schur complements are computed in the first k_r steps, the 2-level semiseparable order is bounded. This also bounds the computational complexity. Due to the fast convergence property, the Schur complement at step k_r gives an efficient approximation of the Schur complements afterwards. We also extend the fast convergence property of the Schur complements from the symmetric positive definite matrix to the symmetric indefinite matrix case. This extension enables us to compute a good approximation of the 3D global saddle-point matrix, which gives an efficient global MSSS preconditioner.

In this part, we apply the block-diagonal MSSS preconditioner (2.5) and `MINRES`

method to iteratively solve the saddle-point system. The computational results are reported in Table 2.15 - 2.17. Note that if the mesh size h is halved, the problem size grows by a factor of 8. Besides, there are three unknowns per grid point. The 1-level semiseparable order is set to be 6 for all the numerical experiments in this part. The iterative solver is stopped if the 2-norm of the residual is reduced by a factor of 10^{-6}. The Hankel blocks approximation is chosen as the model order reduction method. The "preconditioning" column, "MINRES" column and the "total" column represent the same as the tables in Section 2.5.

According to the computational results for different β and h in Table 2.15 - 2.17, we can see that for fixed h and β, the number of iterations decreases as k_r goes up, and a small k_r is enough to compute an efficient preconditioner. Due to the growth of k_r, the time to compute the preconditioner increases. Since only the Schur complements in the first k_r steps are computed, the time to compute the preconditioner increases less than linear when halving the mesh size h. This is illustrated by the "preconditioning" columns in Table 2.15 - 2.17. Moreover, by choosing a small k_r that ranges from 3 to 8, the block MSSS preconditioner also gives nearly size independent convergence and regularization parameter almost independent convergence while the computational complexity is smaller than linear. This property is obtained by carefully checking the computational results given in Table 2.15 - 2.17.

Table 2.15: Block MSSS preconditioner for optimal 3D Poisson equation control with $\beta = 10^{-1}$

problem size	h	k_r	preconditioning (sec.)	iterations	MINRES (sec.)	total (sec.)
1.54e+03	2^{-3}	1	1.59	16	17.41	19.01
		2	2.76	10	11.09	13.85
		3	4.20	6	7.08	11.28
		4	5.68	6	7.15	12.82
1.23e+04	2^{-4}	1	3.35	30	139.81	143.16
		2	6.47	18	86.77	93.24
		3	9.88	12	59.30	69.18
		4	13.36	10	50.42	63.77
9.83e+04	2^{-5}	2	14.47	38	761.27	775.75
		3	22.95	24	503.24	526.18
		4	33.51	18	397.82	431.33
		5	42.83	14	321.34	364.17
7.86e+05	2^{-6}	7	215.42	20	2156.24	2371.66
		8	315.62	18	2024.42	2340.04

Table 2.16: Block MSSS preconditioner for optimal 3D Poisson equation control with $\beta = 10^{-2}$

problem size	h	k_r	preconditioning (sec.)	iterations	MINRES (sec.)	total (sec.)
1.54e+03	2^{-3}	1	1.48	14	15.49	16.97
		2	2.93	8	9.31	12.24
		3	4.29	8	9.22	13.51
		4	6.07	6	7.17	13.24
1.23e+04	2^{-4}	1	3.56	30	141.86	145.42
		2	7.26	16	86.04	86.04
		3	10.59	12	59.85	70.44
		4	13.36	8	42.63	56.82
9.83e+04	2^{-5}	2	15.86	36	726.65	742.51
		3	27.34	24	504.29	531.63
		4	35.72	18	408.10	443.82
		5	50.33	14	356.48	406.80
7.86e+05	2^{-6}	7	216.87	20	2154.61	2371.48
		8	314.44	18	2050.43	2364.87

Table 2.17: Block MSSS preconditioner for optimal 3D Poisson equation control with $\beta = 10^{-3}$

problem size	h	k_r	preconditioning (sec.)	iterations	MINRES (sec.)	total (sec.)
1.54e+03	2^{-3}	2	2.90	14	15.36	19.01
		3	4.44	14	15.60	13.85
		4	6.13	12	13.61	11.28
		5	7.68	12	13.39	12.82
1.23e+04	2^{-4}	2	6.80	14	70.27	143.16
		3	13.04	10	53.51	93.24
		4	20.34	10	53.79	69.18
		5	17.22	10	52.10	63.77
9.83e+04	2^{-5}	2	14.52	32	647.86	775.75
		3	22.43	22	459.30	526.18
		4	30.73	16	347.96	431.33
		5	40.11	12	273.07	364.17
7.86e+05	2^{-6}	6	183.13	22	2880.90	3064.03
		7	214.31	20	2419.73	2634.04
		8	315.58	16	1843.61	2159.19

2.6.2 Global Preconditioners

In the previous subsection, we studied the performance of the block MSSS preconditioner for the optimal control of 3D Poisson equation. In this subsection, we use the same technique with the previous subsection to bound the 2-level semiseparable order of the Schur complements of the symmetric indefinite system. Even the analysis in [35] only holds for the symmetric positive definite matrix case, our numerical experiments illustrate that the fast convergence property of the Schur complements also holds for symmetric indefinite case.

The saddle-point system obtained by using the *discretize-then-optimize* approach has exactly the same form as (2.3) and the system matrix can be represented by,

$$\mathcal{A} = \begin{bmatrix} 2\beta M & 0 & -M \\ 0 & M & K^T \\ -M & K & 0 \end{bmatrix}, \tag{2.57}$$

where K is the stiffness matrix in (2.54), and M is the mass matrix. Since all these matrices are from the discretization of 3D PDEs on a regular domain with uniform mesh, K and M have the same 3-level SSS structure as shown in (2.54). Here we can apply Lemma 2.4 again to permute the global saddle-point matrix $\mathcal{A}$ (2.57) into a global MSSS matrix $\bar{\mathcal{A}}$. The permuted global saddle-point matrix $\bar{\mathcal{A}}$ has the same MSSS structure as subblocks of $\mathcal{A}$, i.e.,

$$\bar{\mathcal{A}} = \begin{bmatrix} \bar{D} & \bar{L} & & & \\ \bar{L} & \bar{D} & \bar{L} & & \\ & \bar{L} & \bar{D} & \ddots & \\ & & \ddots & \ddots & \bar{L} \\ & & & \bar{L} & \bar{D} \end{bmatrix}, \tag{2.58}$$

where $\bar{D}$ and $\bar{L}$ are obtained via Lemma 2.4. To compute an LU factorization, the Schur complements are computed via the following recursions,

$$\begin{cases} \bar{S}_k = \bar{D}, & \text{if } k = 1, \\ \bar{S}_k = \bar{D} - \bar{L}\bar{S}_{k-1}^{-1}\bar{L}, & \text{if } k \geq 2. \end{cases} \tag{2.59}$$

Due to the indefiniteness of $\bar{D}$, the Schur complements are also indefinite. We apply the same method as introduced in the previous subsection: we compute the Schur complements of the first k_r steps and use the Schur complement at step k_r to approximate the Schur complements afterwards. This gives,

$$\begin{cases} \bar{S}_k = \bar{D}, & \text{if } k = 1, \\ \bar{S}_k = \bar{D} - \bar{L}\bar{S}_{k-1}^{-1}\bar{L}, & \text{if } 2 \leq k \leq k_r, \\ \bar{S}_k = \bar{S}_{k_r}, & \text{if } k > k_r. \end{cases} \tag{2.60}$$

By using this approximation for the permuted global system, we can compute the global MSSS preconditioner and apply it to iteratively solve the saddle-point

system. The computational results are given in Table 2.18 - 2.20, where the columns represent the same as Table 2.15 - 2.17.

Since we just compute the first few steps of the Schur complements, the computational complexity to compute the global MSSS preconditioner is smaller than linear. This is stated by the "preconditioning" columns for the same k_r in Table 2.18 - 2.20. The number of iterations decreases as k_r goes up for the same β and h. By using a small k_r, we have already reduced the number of iterations significantly by the global MSSS preconditioner compared with the block-diagonal MSSS preconditioner. Moreover, the global MSSS preconditioner gives virtually mesh size h and regularization parameter β independent convergence for properly chosen k_r.

Table 2.18: Global MSSS preconditioner for optimal 3D Poisson equation control with $\beta = 10^{-1}$

problem size	h	k_r	iterations	preconditioning (sec.)	IDR(4) (sec.)	total (sec.)
1.54e+03	2^{-3}	1	3	3.84	1.96	5.79
		2	3	4.95	1.59	6.53
		3	2	7.70	1.08	8.78
1.23e+04	2^{-4}	2	6	13.37	15.19	28.56
		3	4	22.39	10.79	33.17
		4	3	33.67	8.75	42.42
9.83e+04	2^{-5}	2	8	41.04	106.14	147.18
		3	7	78.87	109.18	188.05
		4	6	143.04	109.26	252.31
7.86e+05	2^{-6}	2	14	153.60	1174.12	1327.72
		3	9	245.24	1101.88	1347.12
		4	8	1152.20	1841.57	2993.78

Table 2.19: Global MSSS preconditioner for optimal 3D Poisson equation control with $\beta = 10^{-2}$

problem size	h	k_r	iterations	preconditioning (sec.)	IDR(4) (sec.)	total (sec.)
1.54e+03	2^{-3}	1	4	3.39	2.69	6.09
		2	3	4.92	1.61	6.53
		3	2	8.13	1.09	9.22
1.23e+04	2^{-4}	2	7	13.41	17.98	31.40
		3	4	22.39	10.78	33.17
		4	3	34.16	8.80	42.95
9.83e+04	2^{-5}	2	8	38.71	103.94	142.65
		3	6	77.30	111.30	188.61
		4	4	155.59	103.77	259.36
7.86e+05	2^{-6}	2	14	209.47	1362.70	1572.17
		3	9	290.69	1132.86	1423.55
		4	8	1181.81	2277.18	3458.99

Table 2.20: Global MSSS preconditioner for optimal 3D Poisson equation control with $\beta = 10^{-3}$

problem size	h	k_r	iterations	preconditioning (sec.)	IDR(4) (sec.)	total (sec.)
1.54e+03	2^{-3}	1	9	2.63	5.26	7.89
		2	4	5.30	2.72	8.03
		3	3	6.32	1.64	7.96
1.23e+04	2^{-4}	2	6	10.54	15.25	25.79
		3	4	19.41	14.26	33.68
		4	4	31.65	17.67	49.32
9.83e+04	2^{-5}	2	8	35.08	104.76	139.84
		3	7	78.38	108.77	187.15
		4	4	134.06	93.27	227.44
7.86e+05	2^{-6}	2	16	162.84	1594.91	1757.75
		3	9	322.00	1328.26	1650.26
		4	8	1503.76	2218.80	3722.56

Remark 2.23 Compare the results for the block-diagonal MSSS preconditioner in Table 2.15 - 2.17 with that of the global MSSS preconditioner in Table 2.18 - 2.20, the global MSSS preconditioner reduces the number of iterations significantly. Even though more time is spent in computing the global MSSS preconditioner for the same group of numerical experiment, the time to iteratively solve the preconditioned system is much reduced due to the fact that fewer iterations are needed. Moreover, the total computation time for the global MSSS preconditioner is less than that for the block MSSS preconditioner.

Remark 2.24 Since there is no efficient model order reduction to reduce the 2-level semiseparable order, the 2-level semiseparable order continues growing before k_r is reached. It is shown in Table 2.18 - 2.20, when k_r goes from 3 to 4 for $h = 2^{-6}$, the time to compute the global MSSS preconditioner increases dramatically. This is due to the fact that when k_r goes from 3 to 4, the 2-level semiseparable order is not bounded by a small number, but by a moderate constant. However, the computational complexity increases slightly more than linear when h goes from 2^{-5} to 2^{-6} for $k_r = 4$. Moreover, the global MSSS preconditioner already gives satisfactory performance by choosing $k_r = 3$ for $h = 2^{-6}$.

2.7 Conclusions

In this chapter, we have studied the multilevel sequentially semiseparable (MSSS) preconditioners for saddle-point systems that arise from the PDE-constrained optimization problems. By exploiting the MSSS structure of the blocks of the saddle-point system, we are able to construct the preconditioners and solve the preconditioned system in linear computational complexity for 2D problems and almost linear computational complexity for 3D problems. To reduce the computational

complexity of computing the preconditioners, we have proposed a new model order reduction algorithm based on the approximate balanced truncation for SSS matrices. We evaluated the performance of the new model order reduction algorithm by comparing with the standard model order reduction algorithm, which is called the Hankel blocks approximation. Numerical experiments illustrate that our model order reduction algorithm is computationally cheaper than the standard method. Besides, it shows that for the optimal control of 2D PDEs, the global preconditioner reduced the number of iterations significantly compared with the block preconditioners. Both preconditioners give mesh size independent convergence and have linear computational complexity. Moreover, the global MSSS preconditioner yields regularization parameter independent convergence while the block MSSS preconditioner does not have this property.

For PDE-constrained optimization problem in 3D, since efficient model order reduction algorithm for 2- or higher- level SSS matrices is still an open problem, we apply an alternative approach to bound the 2-level semiseparable order. Numerical experiments also illustrate that the global MSSS preconditioner gives mesh size and regularization parameter virtually independent convergence while the block MSSS preconditioner just yields mesh size almost independent convergence.

3 Chapter

Evaluation of MSSS Preconditioners On CFD Benchmark Problems Using IFISS

In this chapter, we evaluate the performance of MSSS preconditioners for computational fluid dynamics (CFD) problems. Through numerical experiments on standard CFD benchmark problems in IFISS, we show the performance of the MSSS preconditioners. Numerical results indicate that the global MSSS preconditioner not only yields mesh size independent convergence, but also gives Reynolds number independent convergence. Compared 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 GMG method, and considerably faster than the AMG method.

3.1 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

$$Ax = b \,, \tag{3.1}$$

where $A = [A_{ij}]$ is an $n \times n$ matrix and b is a given right-hand-side vector of compatible size [17]. Normally, the system matrix A is large and sparse. Many efforts have been dedicated to finding efficient solution methods for such systems. Krylov subspace methods such as the conjugate gradient (`CG`), minimal residual (`MINRES`), generalized minimal residual (`GMRES`) and induced dimension reduction (`IDR(s)`) methods are some of the most popular methods [60, 110, 119]. Efficiency and robustness of iterative methods can be improved dramatically by combining

the preconditioning techniques [136]. In this chapter, we study MSSS preconditioners for CFD problems and evaluate the performance of MSSS preconditioners on standard CFD benchmark problems using the Incompressible Flow and Iterative Solver Software (IFISS) [117]. 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.

In this chapter, 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 so avoids the computations of the Schur complements. 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 block preconditioning techniques that use the algebraic multigrid (AMG) method and the geometric multigrid (GMG) method to solve sub-problems which correspond to the $(1, 1)$ block of saddle-point systems. Numerical experiments illustrate that the MSSS preconditioning technique yields mesh size independence convergence and eliminates or reduces the convergence dependency on the Reynolds number, which is an important advantage over the AMG and GMG methods. In addition to robustness, it is shown that the MSSS preconditioning technique is much faster than the AMG method.

The outline of this chapter is as follows. In Section 3.2, we study the MSSS preconditioners for the convection-diffusion equations. We apply both the block MSSS preconditioner and global MSSS preconditioner to the Stokes equation in Section 3.3. In Section 3.4, we apply the global MSSS preconditioner to the linearized Navier-Stokes equation. Conclusions are given in Section 3.5.

3.2 Convection-Diffusion Problem

In this section, we test the performance of MSSS preconditioning techniques for the convection-diffusion problems. The algebraic multigrid (AMG) method and geometric multigrid (GMG) method in IFISS are also used to compare their performance with that of the MSSS preconditioners. The MSSS matrix computations are performed 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, is reached.

In the tables that give numerical results, the "precond." column 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. Induction dimension reduction (IDR(s)) [119, 127] 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 chapter are measured in seconds.

3.2.1 Moderate viscosity case

We first consider the convection-diffusion problem described in Example 3.1, which is given as the example 3.1.4 in [53]. The details of the discretization of the convection-diffusion equation can also be found in [53]. To investigate the performance of the MSSS preconditioning technique, we first consider a moderate viscosity $\nu = 1/200$. Next we consider the convection-dominated case, which has a viscosity parameter $\nu = 10^{-4}$. These experiments are also performed using the AMG and GMG method for comparison.

Example 3.1 ([53]) *Zero source term, recirculating wind, characteristic boundary layers.*

$$\begin{aligned} -\nu\nabla^2 u + \vec{\omega}\cdot\nabla u &= f \quad in \quad \Omega \\ u &= u_D \quad on \quad \partial\Omega \end{aligned}$$

where $\Omega = \{(x,y)| -1 < x < 1, -1 < y < 1\}$, $\vec{\omega} = \big(2y(1-x^2),\ -2x(1-y^2)\big)$, $f = 0$. *Homogeneous Dirichlet boundary are imposed everywhere except that* $u_D = 1$ *on* $\big[-1,1\big] \times 1$.

We use the Q_1 finite element method to discretize the convection-diffusion equation. First, we consider a moderate value for the viscosity parameter $\nu = 1/200$, the computational results for the MSSS preconditioner and the AMG and GMG method are listed in Table 3.3 - 3.2. The maximum semiseparable orders for the MSSS preconditioner are 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 3.1.

Table 3.1: AMG method for $\nu = 1/200$

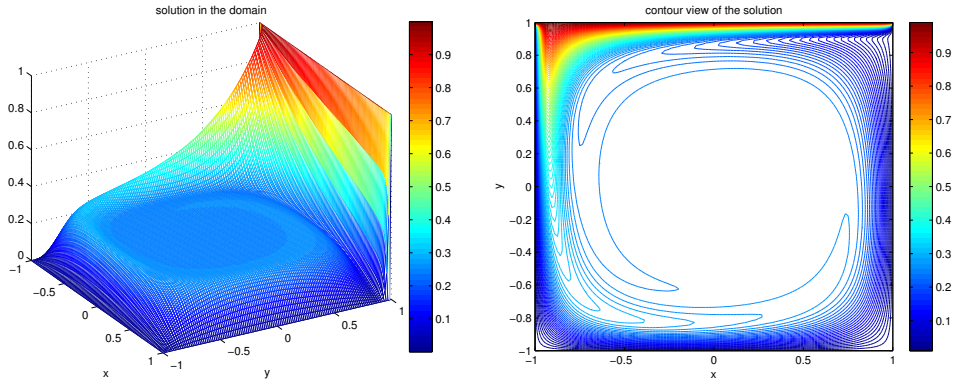
mesh size	problem size	#iter.	precond. (sec.)	IDR(4) (sec.)	total (sec.)
2^{-4}	1.09e+03	8	0.49	0.06	0.55
2^{-5}	4.23e+03	4	2.38	0.05	2.43
2^{-6}	1.66e+04	4	14.30	0.17	14.47
2^{-7}	6.60e+04	4	127.71	0.28	127.99
2^{-8}	2.63e+05	4	2513.11	1.53	2514.64

Table 3.2: GMG method for $\nu = \frac{1}{200}$

mesh size	problem size	#iter.	precond. (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

Table 3.3: MSSS Preconditioner for $\nu = {}^{1}/_{200}$

mesh size	problem size	#iter.	precond. (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



(a) Solution of u.

(b) Contour view.

Figure 3.1: Solution of test problem 3.1 for $\nu = {}^{1}/_{200}$ and $h = 2^{-7}$

According to Table 3.3, the MSSS preconditioner gives mesh size independent convergence for this 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 3.1 - 3.2 for the AMG and GMG methods, it is clear to see that the computational time of the AMG method setup is much bigger than 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 3.1 illustrates that the computational complexity for setting up the AMG method grows with the problem size and is bigger than linear. This is due to the fact that MATLAB is not competitive in speed and the MATLAB code in IFISS is not highly optimized. Or perhaps the AMG method implemented in IFISS is not of linear computational complexity.

It is clear to see from Table 3.1 - 3.3 that for the convection-diffusion problem with moderate viscosity, the GMG method is competitive.

3.2.2 Small viscosity case

Next, we test the convection-dominated case with the viscosity parameter $\nu = 10^{-4}$ for the MSSS preconditioner, the AMG and GMG method. The computational results are listed in Table 3.4 - 3.6. The solution for the mesh size $h = 2^{-7}$ is shown in Figure 3.2.

Table 3.4: MSSS preconditioner with $\nu = 10^{-4}$

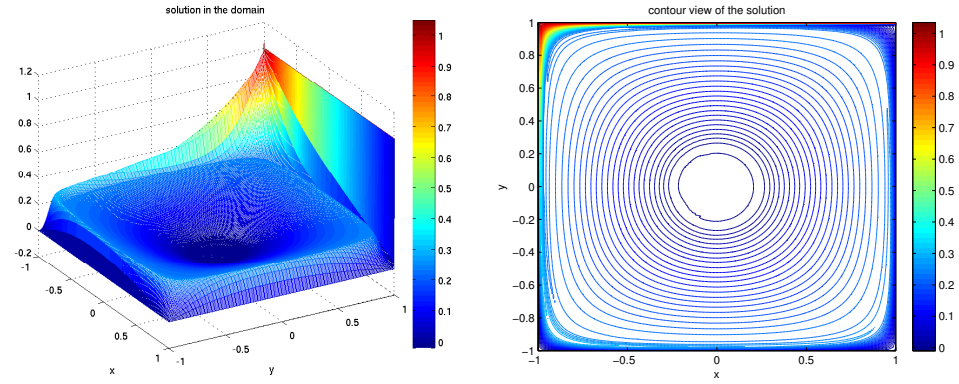
mesh size	problem size	#iter.	precond. (sec.)	IDR(4) (sec.)	total (sec.)
$2^{-4}(12)$	1.09e+03	14	0.46	0.84	1.30
$2^{-5}(24)$	4.23e+03	11	1.61	1.89	3.50
$2^{-6}(26)$	1.66e+04	12	6.68	6.80	13.48
$2^{-7}(26)$	6.60e+04	14	29.90	16.68	46.58
$2^{-8}(10)$	2.63e+05	5	66.63	38.22	104.85

Table 3.5: AMG method with $\nu = 10^{-4}$

mesh size	problem size	#iter.	precond. (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 3.6: GMG method with $\nu = 10^{-4}$

mesh size	problem size	#iter.	precond. (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	-



(a) Solution of u.

(b) Contour view.

Figure 3.2: Solution of test problem 3.1 for $\nu = 10^{-4}$ and $h = 2^{-7}$.

For the convection-dominated problem test case, the system is ill-conditioned. It is therefore more difficult to compute a good enough preconditioner, and a larger semiseparable order is needed to compute an accurate enough approximation compared with the case for $\nu = {}^{1}/_{200}$. This is illustrated by comparing the semiseparable orders in Table 3.3 and Table 3.4. 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 larger ν, the computational time still scales linearly with the problem size. Due to the ill-conditioning of the problem, both AMG and GMG method fail.

Remark 3.1 Compared with the AMG and GMG method, the MSSS preconditioner is more robust. Moreover, the MSSS preconditioning technique is considerably faster than the AMG method.

3.3 Stokes Problem

In this section, we evaluate the performance of the MSSS preconditioners for the lid-driven cavity problem described by the Stokes equation that is given by Example 3.2. This example is given as example 5.1.3 in [53]. Mixed finite elements are used for the discretization of the Stokes equation [53].

Example 3.2 ([53]) *Lid-driven cavity problem, enclosed flow boundary condition.*

$$\begin{aligned} -\nabla^2 \overrightarrow{u} + \nabla p &= 0 \\ \nabla \cdot \overrightarrow{u} &= 0 \end{aligned}$$

in a square domain $\Omega = \{(x, y) | -1 \leq x \leq 1, -1 \leq y \leq 1\}$, *where the regularized*

cavity condition is imposed, i.e.,

$$\overrightarrow{u} = \begin{cases} (1 - x^4,\ 0) & on \quad \partial\Omega_1 = [-1,\ 1] \times 1 \\ (0,\ 0) & on \quad \partial\Omega \backslash \partial\Omega_1 \end{cases}.$$

Here, $\partial\Omega$ denotes the boundary of the domain Ω.

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

$$\begin{bmatrix} K & B^T \\ B & -S_t \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}, \tag{3.2}$$

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.

3.3.1 Block Preconditioners

The Schur complement for CFD problems is even more difficult to compute or approximate than that for PDE-constrained optimization problems. A standard way is to compute an approximation that has an equivalent spectrum with the Schur complement. For the discrete Stokes equation, it is shown in [130] that the Schur complement has an equivalent spectrum with the pressure mass matrix M_p, i.e., the relation

$$\gamma^2 \leq \frac{x^T B K^{-1} B^T x}{x^T M_p x} \leq \Gamma^2, \quad \forall x \in \mathbb{R}^{n_p} \backslash \{\mathbf{0}\} \tag{3.3}$$

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

$$\mathcal{P}_1 = \begin{bmatrix} K & \\ & M_p \end{bmatrix}, \quad \mathcal{P}_2 = \begin{bmatrix} K & \\ B & -M_p \end{bmatrix}. \tag{3.4}$$

This type of preconditioners are called the Silvester-Wathen preconditioner and is widely studied for the Stokes problems in [53, 118, 130].

Since the diagonal blocks of the block preconditioners are MSSS matrices, an obvious way is to apply the Silvester-Wathen preconditioner to iteratively solve the discrete Stokes equation by MSSS matrix computations. In addition, the global system matrix of the discrete Stokes equation (3.2) has MSSS blocks, we can apply the global MSSS preconditioner to iteratively solve the global system (3.2) by Lemma 2.4. 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

$$\mathcal{P}_1 = \begin{bmatrix} \hat{K} & \\ & \hat{M}_p \end{bmatrix} \tag{3.5}$$

where $\hat{K}$ is the approximation of K, $\hat{M}_p$ is the lumped pressure mass matrix M_p. For comparison, the AMG and GMG methods, together with the block MSSS preconditioner, are performed to approximate K. Due to the symmetric definiteness of the block diagonal preconditioner and the symmetry and indefiniteness of the saddle point system, minimal residual (MINRES) method [92] is chosen as the iterative solver. The results for block MSSS preconditioner are listed in Table 3.7 and for the AMG and GMG method are given in Table 3.8 and Table 3.9. The smoother for the AMG and GMG method is chosen as the point damped Jacobi, which is very computationally cheap.

Table 3.7: Silvester-Wathen preconditioner for the Stokes equation by MSSS matrix computations

h	problem size	#iter.	precond. (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 3.8: Silvester-Wathen preconditioner for the Stokes equation by AMG method

h	problem size	#iter.	precond. (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 3.9: Silvester-Wathen preconditioner for the Stokes equation by GMG method

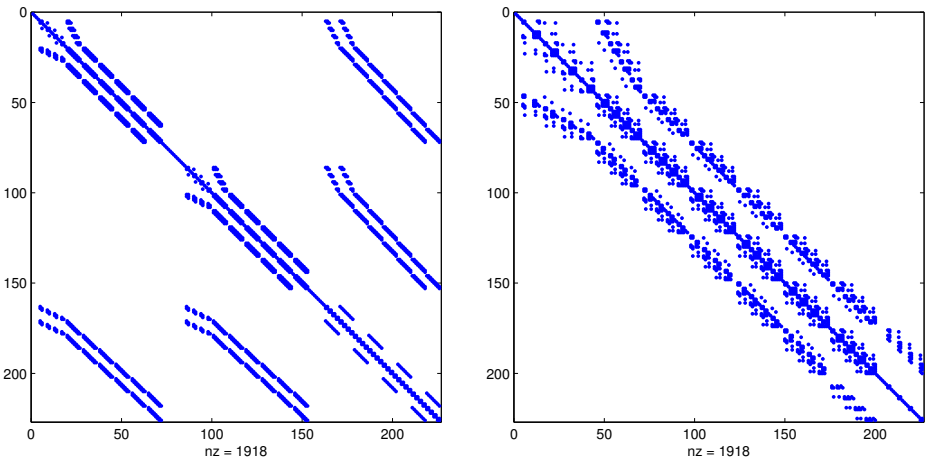
h	problem size	#iter.	precond. (sec.)	MINRES (sec.)	total (sec.)
2^{-4}	3.20e+03	34	0.09	0.09	0.18
2^{-5}	1.25e+04	34	0.14	0.28	0.42
2^{-6}	4.97e+04	32	0.61	0.58	1.19
2^{-7}	1.98e+05	32	2.01	2.00	4.01
2^{-8}	7.88e+05	30	3.26	7.38	10.64

Results in Table 3.7 - 3.9 illustrate that the block MSSS preconditioner, together with the AMG and GMG methods, gives mesh size independent convergence. The time to compute the block MSSS preconditioner and to solve the preconditioned system scale linearly with the problem size, which is verified in Table 3.7. 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.

For the block MSSS preconditioner, most time was spent to solve the preconditioned system. This is mainly due to the 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 (3.2).

3.3.2 Global Preconditioner

It is not difficult to verify that for the discrete Stokes system (3.2), all the matrix blocks K, B and S_t are MSSS matrices. Thus we can apply Lemma 2.4 to permute the saddle-point system (3.2) with MSSS blocks into a single MSSS system. Then, the LU factorization for the MSSS matrices in Lemma 2.8 can be performed. The spy plot of the matrix (3.2) before and after permutation are given by Figure 3.3. Here, "nz" represents the number of nonzeros.



(a) Before permutation (b) After permutation

Figure 3.3: Structure of system matrix (3.2) before and after permutation for $h = 2^{-2}$

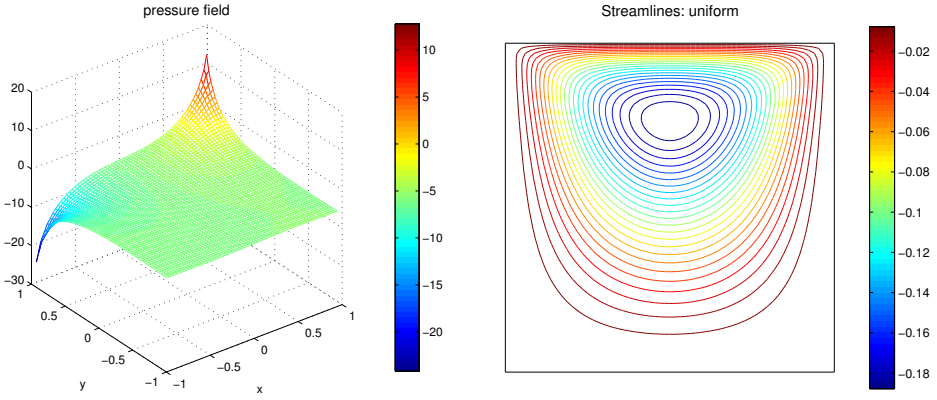
Due to the indefiniteness of the global preconditioner, `IDR(s)` is chosen as the iterative solver. The computational results for the global preconditioner are listed in Table 3.10. The solution of the pressure field and the streamlines are shown in Figure 3.4.

Table 3.10: 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

Computational results in Table 3.10 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 Table 3.7 - 3.9 with the results for the global MSSS preconditioners in Table 3.10, it is obvious that the number of iterations for global MSSS preconditioner is much more reduced. Thus, the time to solve the preconditioned system for the global MSSS preconditioner is also much less than the time for the block MSSS preconditioner.



(a) Pressure distribution.

(b) Streamline.

Figure 3.4: Solution of test example 3.2 for MSSS preconditioners

Remark 3.2 The global MSSS preconditioner performs much better than the Silvester-Wathen preconditioner by the AMG method for the solution of middle and large size discrete Stokes equation. Even though 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 problem.

3.4 Navier-Stokes Problem

The last example we consider is the lid-driven cavity problem of the Navier-Stokes equation that is given in Example 3.3, which is introduced as example 7.1.3 in [53].

Example 3.3 ([53]) *Lid-driven cavity problem, enclosed flow boundary condition.*

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

in a square domain $\Omega = \{(x,y)| -1 \le x \le 1, -1 \le y \le 1\}$*, where the regularized cavity condition is imposed, i.e.,*

$$\overrightarrow{u} = \begin{cases} (1-x^4,\ 0) & on \quad \partial\Omega_1 = [-1,\ 1]\times 1 \\ (0,\ 0) & on \quad \partial\Omega\backslash\partial\Omega_1 \end{cases}.$$

Here, $\partial\Omega$ *denotes the boundary of the domain* Ω*.*

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 [53]. In this section, we use the Newton method to linearize and $Q_1 - P_0$ finite element method to discretize. At each linearized step, we need to solve a linear system of the following form

$$\begin{bmatrix} \nu K_x + N + W_{xx} & W_{xy} & B_x^T \\ W_{yx} & \nu K_y + N + W_{yy} & B_y^T \\ B_x & B_y & -\frac{1}{\nu}S_t \end{bmatrix} \begin{bmatrix} \Delta u_x \\ \Delta u_y \\ \Delta p \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \\ g \end{bmatrix}, \tag{3.6}$$

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 (3.6), 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 [16, 53, 74, 90].

The generic form of system (3.6) can be written as

$$\begin{bmatrix} \mathbf{F} & \mathbf{B}^T \\ \mathbf{B} & -\frac{1}{\nu}S_t \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ p \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ g \end{bmatrix} \tag{3.7}$$

where $\mathbf{F}$, $\mathbf{B}$ in (3.7) satisfy some partition rules of the matrix in (3.6). One of the standard block preconditioners for the linearized Navier-Stokes equation (3.7) is called the pressure convection-diffusion (PCD) preconditioner, which is discussed

in [53, 74, 90]. The PCD preconditioner can be written as

$$\mathcal{P} = \begin{bmatrix} \mathbf{F} & \mathbf{B}^T \\ \mathbf{0} & -S \end{bmatrix} \tag{3.8}$$

where S is the approximate Schur complement and is given as

$$S = L_p A_p^{-1} M_p. \tag{3.9}$$

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. Details of the pressure convection-diffusion preconditioner can be found in [53, 74, 90].

We also note that all the matrix blocks in (3.6) have MSSS structure, thus we can apply Lemma 2.4 to permute the block system (3.6) into a single MSSS 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 (3.6) at the second Newton step. For comparison, we also carry out numerical experiments to solve Example 3.3 at the second Newton step by the PCD preconditioner (3.8).

Since 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 Table 3.11 - 3.12.

Numerical results in Table 3.11 - 3.12 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 3.12. We can also find that the global MSSS preconditioner is much faster than the PCD preconditioner by the AMG method.

Table 3.11: Global MSSS preconditioner for the 2nd Newton Step with $\nu = 10^{-1}$

h	problem size	#iter.	precond. (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

Table 3.12: PCD preconditioner by the AMG method for the 2nd Newton Step with $\nu = 10^{-1}$

h	problem size	#iter.	precond. (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

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 Table 3.13 - 3.14.

Table 3.13: Global MSSS preconditioner for the 2nd Newton Step with $\nu = 10^{-2}$

h	problem size	#iter.	precond. (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

Table 3.14: PCD preconditioner by the AMG method for the 2nd Newton Step with $\nu = 10^{-2}$

h	problem size	#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

For bigger Reynolds number, both the global MSSS preconditioner and PCD preconditioner gives mesh size independent convergence. This is verified by the numerical results listed in Table 3.13 - 3.14. In addition, the computational time for the global preconditioner is linear with the problem size while the PCD preconditioner does not have such linear computational complexity. Table 3.13 - 3.14 show that the global MSSS preconditioner is much faster than the PCD preconditioner by the AMG method.

Remark 3.3 According to the computational results for different Reynolds number in Table 3.11 - 3.14, 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 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.

3.5 Conclusions

In this chapter, we have studied a new class of preconditioners for computational fluid dynamics (CFD) problems. This type of preconditioners exploits the multilevel sequentially semiseparable (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 algebraic multigrid (AMG) method and geometric multigrid (GMG) method to the CFD benchmark problems in IFISS to evaluate the performance of the MSSS preconditioners. Numerical experiments show that the 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-diffusion problem with small viscosity parameter. 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.

4 Chapter

Convergence Analysis of MSSS Preconditioners

Multilevel sequentially semiseparable (MSSS) matrices are a class of structured matrices that have low-rank off-diagonal structure, which allows that matrix-matrix operations can be performed in linear computational complexity. MSSS preconditioners are computed via replacing the Schur complements in the block LU factorization of the global linear system by MSSS matrix approximations with low off-diagonal rank. In this chapter, we analyze the convergence properties of such preconditioners. We show that the spectrum of the preconditioned system is contained in a circle centered at $(1, 0)$ and give an analytic bound of the radius of this circle. This radius can be made arbitrarily small by properly setting a parameter in the MSSS preconditioner. Our results apply to a wide class of linear systems. The system matrix can be either symmetric or unsymmetric, definite or indefinite. We demonstrate our analysis by numerical experiments.

4.1 Introduction

In this chapter, we present a full convergence analysis of the MSSS preconditioners for a wide class of linear systems. The system matrix can be either symmetric or unsymmetric, definite or indefinite, where saddle-point systems, discretized Helmholtz equations, and discretized convection-diffusion equations are automatically covered. Our analysis gives an analytic bound for the spectrum of the preconditioned system. We show that the spectrum of the preconditioned system is contained in a circle centered at $(1, 0)$ and give an analytic bound for the radius of this circle. This radius can be made arbitrarily small by properly setting a parameter in the MSSS preconditioner.

Some related work includes [11] and [87]. Both analyses apply only to symmetric positive definite systems. The analysis for MSSS preconditioners in [87] is restricted to 1-level MSSS matrix computations, while our analysis can be applied to 2-level MSSS matrix computations. Our work in this chapter is closely related to [11].

Our contributions include: (1) We extend the work in [11, 87] for the symmetric positive definite case to the general linear systems. (2) Our analysis can also be applied to block linear systems from discretization of coupled PDEs, while the analysis in [11, 87] only applies to symmetric positive definite linear systems that arise from discretization of scalar PDEs. (3) We give an analytic bound for the error introduced by the model order reduction that is necessary for the MSSS preconditioning technique. (4) The analysis for MSSS preconditioning in [87] only concerns 1-level MSSS matrix computations, while our analysis also includes the 2-level MSSS cases. (5) For the first time, we apply this MSSS preconditioning technique to the Helmholtz equation.

The structure of this chapter is as follows. We give a brief introduction of the MSSS preconditioning technique in Section 4.2, and analyze its convergence in Section 4.3. Section 4.4 studies the numerical stability of this preconditioning technique and gives a sufficient condition to avoid breakdown. The underlying condition can be satisfied by properly setting a parameter. We show how to choose this parameter in Section 4.5. Numerical experiments are given in Section 4.6, while conclusions are drawn in the final part.

4.2 MSSS Preconditioners for Discretized PDEs

Consider the following PDE

$$\mathcal{L}u = f, \text{ with } u = u_D \text{ on } \Gamma_D,$$

on a square domain $\Omega \in \mathbb{R}^d$ with $d = 2,$ or 3, $\mathcal{L}$ is a linear differential operator, and $\Gamma_D = \partial\Omega$. Discretizing the PDE using lower order finite difference or finite element methods and using lexicographical to order the grid points gives the following linear system,

$$Kx = b,$$

where the stiffness matrix K is of the block tridiagonal form (4.1).

$$K = \begin{bmatrix} K_{1,1} & K_{1,2} & & & \\ K_{2,1} & K_{2,2} & K_{2,3} & & \\ & K_{3,2} & K_{3,3} & \ddots & \\ & & \ddots & \ddots & \ddots \\ & & & \ddots & K_{N,N} \end{bmatrix}. \tag{4.1}$$

Here $K_{i,j}$ is again a tridiagonal matrix for $d = 2$ and block tridiagonal matrix for $d = 3$. When higher order finite difference or finite element methods are used to discretize the PDE, we can reorder the grid points so that we can still get the stiffness matrix of the form (4.1).

For discretized scalar PDEs using uniform mesh, it is quite natural to infer that the stiffness matrix K has an MSSS matrix structure, i.e., a 2-level MSSS structure for $d = 2$ and a 3-level MSSS structure for $d = 3$. Discretized coupled PDEs, such

as the Stokes equation, and linearized Navier-Stokes equation yield a linear system of saddle-point type. It is shown in Chapter 2, that all the sub-matrices in the saddle-point matrix have an MSSS structure and can be permuted into a global MSSS structure that has the same form as (4.1) with $K_{i,j}$ a 2-level SSS matrix for $d = 3$ and 1-level SSS matrix for $d = 2$.

For a strongly regular $N \times N$ block matrix K, it admits the block factorization that is given by $K = LSU$. Here we say that a matrix is strongly regular if all the leading principle sub-matrices are nonsingular. S is a block diagonal matrix with its i-th diagonal block given by

$$S_i = \begin{cases} K_{i,i} & \text{if } i = 1 \\ K_{i,i} - K_{i,i-1} S_{i-1}^{-1} K_{i-1,i} & \text{if } 2 \le i \le N \end{cases}, \tag{4.2}$$

where S_i is the Schur complement at the i-th step. The matrices L and U are block bidiagonal matrices of lower-triangular form and upper-triangular form, respectively. They are obtained by computing

$$L_{i,j} = \begin{cases} I & \text{if } i = j \\ K_{i,j} S_j^{-1} & \text{if } i = j + 1 \end{cases}, \text{ and } U_{i,j} = \begin{cases} I & \text{if } j = i \\ S_i^{-1} K_{i,j} & \text{if } j = i + 1 \end{cases}.$$

To compute such a factorization, one needs to compute the Schur complements via (4.2). This is computationally expensive both in time and memory since the Schur complement S_i is a full matrix. Some earlier papers [37, 83] propose for symmetric positive definite systems to approximate S_i by using the off-diagonal decay property of the inverse of a symmetric positive definite tridiagonal matrix. Alternatively, an incomplete factorization can be made to reduce the fill-in within the bandwidth for such a factorization [110]. However, these methods do not yield satisfactory performance. In [12, 35], it is stated that the Schur complement S_i from the factorization of a discretized symmetric positive definite differential operator has low off-diagonal rank. Therefore, S_i can be approximated by a matrix with low off-diagonal rank, where the $\mathcal{H}$-matrix and SSS matrix belong to this class of matrices with low off-diagonal rank blocks. In this chapter, we use SSS matrices to approximate the Schur complements in the above factorization for a general class of linear systems. Note that $\mathcal{H}$-matrix computations have mainly been applied to approximate the Schur complements that are either symmetric positive definite [11, 12] or unsymmetric, whose eigenvalues have positive real part [78]. Some recent efforts have been made to solve the symmetric indefinite Helmholtz equation [54].

For the approximated Schur complement by MSSS matrix computations denoted by $\tilde{S}_i, (i = 1,\ 2,\ \cdots,\ N)$, we have the MSSS preconditioner $\tilde{K}$ that is given by

$$\tilde{K} = \tilde{L}\tilde{S}\tilde{U}. \tag{4.3}$$

Here

$$\tilde{L}_{i,j} = \begin{cases} I & \text{if } i = j \\ K_{i,j}\tilde{S}_j^{-1} & \text{if } i = j+1 \end{cases}, \quad \tilde{U}_{i,j} = \begin{cases} I & \text{if } j = i \\ \tilde{S}_i^{-1}K_{i,j} & \text{if } j = i+1 \end{cases},$$

and $\tilde{S}$ is a block-diagonal matrix with $\tilde{S}_i, (i = 1,\ 2,\ \cdots,\ N)$ as its diagonal blocks.

The Schur complement always corresponds to a problem that is one dimension lower than the linear system, i.e., for a 2D system, the Schur complement is of 1D, and 2D for a 3D system. When applying MSSS preconditioners to 3D problems, one needs 2-level MSSS matrix computations to approximate the Schur complement. How to reduce the off-diagonal rank of a 2-level MSSS matrix efficiently is still an open problem [35, 40, 103]. This makes extending MSSS preconditioning technique from 2D to 3D nontrivial, and some extra effort needs to be devoted, cf. [40, 103]. To keep the consistency of this chapter, we only focus on the convergence analysis of the MSSS preconditioner for 2D systems.

4.3 Convergence Analysis

In this section, we analyze the convergence of the MSSS preconditioner. Some recent work devoted to the analysis of structured preconditioners are in [11, 87]. In [87], the nested dissection method is used to order the unknowns of the discretized diffusion-reaction equation in 2D and the symmetric positive definite Schur complements are approximated by SSS matrix and HSS matrix computations, respectively. Analytic bounds of the spectrum of the preconditioned system are given. In [11], $\mathcal{H}$-matrix computations are applied to preconditioning the 2D symmetric positive definite Helmholtz equation. Both studies focus on the symmetric positive definite case.

In [11], it is stated that the key point for the $\mathcal{H}$-matrix preconditioner for symmetric positive definite systems is not how well the approximate Schur complement denoted by $\tilde{S}_i$ approximates the exact Schur complement S_i, but how small the distance between $\tilde{S}_i$ and $K_{i,i} - K_{i,i-1}\tilde{S}_{i-1}^{-1}K_{i-1,i}$ is. This statement is denoted by the so-called "condition ε" in [11]. In this chapter, we also make use of this condition for convergence analysis, which is given by the following definition.

Definition 4.1 (Condition ε [11]) *For the approximate block factorization* (4.3) *of the matrix K in* (4.1)*, there exists a constant ε such that*

$$\begin{aligned} \left\| \tilde{S}_1 - K_{1,1} \right\|_2 &\leq \varepsilon, \\ \left\| \tilde{S}_i - \left(K_{i,i} - K_{i,i-1}\tilde{S}_{i-1}^{-1}K_{i-1,i} \right) \right\|_2 &\leq \varepsilon, \end{aligned} \tag{4.4}$$

hold for $2 \leq i \leq N$.

If condition ε in Definition 4.1 holds, we have the following lemma that gives the distance between the preconditioner $\tilde{K}$ and the original system K.

Lemma 4.1 *Let K be a nonsingular matrix that has the form of* (4.1)*, suppose* $\tilde{S}_i,\ i = 1, 2, \ldots, N$ *in* (4.4) *are nonsingular and condition ε holds. The MSSS preconditioner is given by* (4.3)*, and*

$$\left\| K - \tilde{K} \right\|_2 \leq \varepsilon.$$

Proof: Define the matrix E_i $(i = 1,\ 2,\ \ldots\ ,\ N)$ as

$$E_i = \begin{cases} \tilde{S}_i - K_{i,i} & \text{if} \quad i = 1, \\ \tilde{S}_i - \left(K_{i,i} - K_{i,i-1} \tilde{S}_{i-1}^{-1} K_{i-1,i} \right) & \text{if} \quad 2 \leq i \leq N. \end{cases}$$

This in turn gives

$$\tilde{K} = \begin{bmatrix} K_{1,1} + E_1 & K_{1,2} & & & \\ K_{2,1} & K_{2,2} + E_2 & K_{2,3} & & \\ & K_{3,2} & K_{3,3} + E_3 & \ddots & \\ & & \ddots & \ddots & \ddots \\ & & & \ddots & \ddots \end{bmatrix}. \tag{4.5}$$

Then we have,

$$K - \tilde{K} = \begin{bmatrix} E_1 & & & \\ & E_2 & & \\ & & \ddots & \\ & & & \ddots \end{bmatrix}.$$

Given any vector x of compatible size for operations of Kx, by proper partitioning $x = \left[x_1^T,\ x_2^T,\ \ldots\ ,\ x_N^T \right]^T$, we have

$$\begin{aligned} \left\| (K - \tilde{K})x \right\|_2 &= \sum_{i=1}^{N} \|E_i x_i\|_2^2 \\ &\leq \sum_{i=1}^{N} \|E_i\|_2^2 \, \|x_i\|_2^2 . \end{aligned}$$

Since the condition ε is satisfied, we have $\|E_i\|_2 \leq \varepsilon$, this yields

$$\left\| (K - \tilde{K})x \right\|_2^2 \leq \sum_{i=1}^{N} \varepsilon^2 \, \|x_i\|_2^2 = \varepsilon^2 \sum_{i=1}^{N} \|x_i\|_2^2 = \varepsilon^2 \, \|x\|_2^2 ,$$

i.e., for the induced 2-norm, we have

$$\left\| K - \tilde{K} \right\|_2 \leq \varepsilon. \qquad \square$$

Next, we introduce how to compute an MSSS preconditioner that satisfies Lemma 4.1. Here, we assume the exact arithmetic.

Lemma 4.2 *Let the lower semiseparable order and upper semiseparable order be defined by Definition 2.3 in Chapter 2. For a nonsingular SSS matrix A with lower semiseparable order r_l and upper semiseparable order r_u, the exact inverse of A can be computed using SSS matrix computations in linear computational complexity provided that r_l and r_u are much smaller than the size of A. The inverse of A is again an SSS matrix with r_l and r_u as its lower and upper semiseparable order, respectively.*

Proof: This can be shown by carefully checking the arithmetic for inverting SSS matrices described in [33, 44]. □

Lemma 4.3 *Let A and B be SSS matrices with compatible sizes and properly partitioned blocks for matrix-matrix addition and multiplication, then $A + B$ and AB can be computed exactly using SSS matrix computations in linear computational complexity if no model order reduction is performed.*

Proof: Proof of this lemma is given by checking the algorithms for SSS matrices introduced in [33, 46]. □

For the 2D matrix K of the form in (4.1), all its sub-blocks are SSS matrices, therefore we have the following corollary that shows how the condition ε in Definition 4.1 can be satisfied.

Corollary 4.4 *Suppose $\tilde{S}_i,\ i = 1, 2, \ldots, N$ in (4.4) are nonsingular, then the condition ε can be satisfied by applying the following procedure.*

1. *Invert $\tilde{S}_{i-1}$ using the SSS inversion algorithm.*
2. *Compute $K_{i,i} - K_{i,i-1}\tilde{S}_{i-1}^{-1}K_{i-1,i}$ using SSS matrix computations without performing the model order reduction.*
3. *Perform the model order reduction for $K_{i,i} - K_{i,i-1}\tilde{S}_{i-1}^{-1}K_{i-1,i}$ by choosing a proper semiseparable order r_k or a proper bound τ for the discarded singular values, which will be introduced in Section 4.5, such that the condition*

$$\left\| \tilde{S}_i - \left(K_{i,i} - K_{i,i-1}\tilde{S}_{i-1}^{-1}K_{i-1,i} \right) \right\|_2 \leq \varepsilon$$

is satisfied.

Proof: According to Lemma 4.2 and Lemma 4.3, both step 1 and 2 can be performed exactly. By applying the Hankel blocks approximation introduced in [33, 103], $K_{i,i} - K_{i,i-1}\tilde{S}_{i-1}^{-1}K_{i-1,i}$ can be approximated by an SSS matrix $\tilde{S}_i$ in a prescribed accuracy ε measured in the matrix 2-norm by properly chosen parameters in the Hankel block approximations.

Details of the Hankel blocks approximation error bound control will be introduced in Section 4.5. □

Remark 4.5 To satisfy condition ε, we need to apply Corollary 4.4 to compute the MSSS preconditioner. This is computationally cheaper than full matrix computations and is also feasible in practice, because the semiseparable orders of $K_{i,i}$, $\tilde{S}_{i-1}$, $K_{i,i-1}$ and $K_{i-1,i}$ are small. Performing step 2 in Corollary 4.4 just increases the semiseparable order slightly and the bound for the growth of the semiseparable order is given by Lemma 2.10 in Chapter 2. After performing the model order reduction in step 3, the semiseparable order is reduced and bounded. This gives the approximate Schur complements $\tilde{S}_i$ with small semiseparable order.

Lemma 4.1 gives the distance between the preconditioner and the original system matrix while Corollary 4.4 illustrates how to satisfy the condition ε. Normally, we do not consider this distance, but the distance between the preconditioned matrix and the identity matrix. Next, we give an analytic bound for this distance.

Theorem 4.6 *Let a nonsingular matrix K be of the form* (4.1) *and let the condition ε in Definition 4.1 hold. If $\tilde{S}_i$ $(i = 1,\ 2,\ \ldots\ ,\ N)$ are nonsingular and $\varepsilon < \varepsilon_0$, then the MSSS preconditioner is given by* (4.3) *and*

$$\left\| I - \tilde{K}^{-1}K \right\|_2 \leq \frac{\varepsilon}{\varepsilon_0 - \varepsilon}.$$

Here ε_0 is the smallest singular value of K.

Proof: Since $\tilde{S}_i$ $(i = 1, 2, \ldots, N)$ are nonsingular, $\tilde{K}$ is nonsingular. Then,

$$\left\| I - K^{-1}\tilde{K} \right\|_2 = \left\| K^{-1}(K - \tilde{K}) \right\|_2 \leq \left\| K^{-1} \right\|_2 \left\| K - \tilde{K} \right\|_2 \leq \varepsilon \left\| K^{-1} \right\|_2 = \frac{\varepsilon}{\varepsilon_0}.$$

Since $\varepsilon < \varepsilon_0$, we have $\dfrac{\varepsilon}{\varepsilon_0} < 1$, then the Neumann series

$$I + \left(I - K^{-1}\tilde{K} \right) + \left(I - K^{-1}\tilde{K} \right)^2 + \cdots\cdots$$

converges to $\left(I - (I - K^{-1}\tilde{K}) \right)^{-1} = \tilde{K}^{-1}K$. This in turn gives,

$$\begin{aligned}
\left\| \tilde{K}^{-1}K \right\|_2 &= \left\| I + \left(I - K^{-1}\tilde{K} \right) + \left(I - K^{-1}\tilde{K} \right)^2 + \cdots \right\|_2 \\
&\leq 1 + \left\| I - K^{-1}\tilde{K} \right\|_2 + \left\| \left(I - K^{-1}\tilde{K} \right)^2 \right\|_2 + \cdots\cdots \\
&\leq 1 + \frac{\varepsilon}{\varepsilon_0} + \left(\frac{\varepsilon}{\varepsilon_0} \right)^2 + \cdots\cdots \\
&= \frac{\varepsilon_0}{\varepsilon_0 - \varepsilon}.
\end{aligned}$$

Then, we can obtain

$$\left\| \tilde{K}^{-1} \right\|_2 = \left\| \tilde{K}^{-1}KK^{-1} \right\|_2 \leq \left\| \tilde{K}^{-1}K \right\|_2 \left\| K^{-1} \right\|_2 \leq \frac{\varepsilon_0}{\varepsilon_0 - \varepsilon} \times \frac{1}{\varepsilon_0} = \frac{1}{\varepsilon_0 - \varepsilon}.$$

Using Lemma 4.1 in turn yields

$$\left\|I - \tilde{K}^{-1}K\right\|_2 = \left\|\tilde{K}^{-1}(\tilde{K} - K)\right\|_2 \leq \left\|\tilde{K}^{-1}\right\|_2 \left\|\tilde{K} - K\right\|_2 \leq \frac{\varepsilon}{\varepsilon_0 - \varepsilon}. \qquad \square$$

According to Theorem 4.6 we have the following proposition that gives the condition number of the preconditioned matrix.

Proposition 4.7 *Let a nonsingular matrix K be of the form* (4.1) *and let the condition ε in Definition 4.1 hold. If $\tilde{S}_i$ $(i = 1, 2, \ldots, N)$ are nonsingular and $\varepsilon < \frac{1}{2}\varepsilon_0$, then we have the MSSS preconditioner $\tilde{K}$ is of the form* (4.3) *and*

$$\kappa_2(\tilde{K}^{-1}K) \leq \frac{\varepsilon_0}{\varepsilon_0 - 2\varepsilon}.$$

Here ε_0 is the smallest singular value of K, and $\kappa_2(\cdot)$ represents the condition number measured using the matrix 2-norm.

Proof: According to Theorem 4.6, we have

$$\left\|I - \tilde{K}^{-1}K\right\|_2 \leq \frac{\varepsilon}{\varepsilon_0 - \varepsilon},$$

associated with $\varepsilon < \frac{1}{2}\varepsilon_0$, we get $\dfrac{\varepsilon}{\varepsilon_0 - \varepsilon} < 1$. Then the Neumann series

$$I + \left(I - \tilde{K}^{-1}K\right) + \left(I - \tilde{K}^{-1}K\right)^2 + \cdots\cdots$$

converges to $\left(I - (I - \tilde{K}^{-1}K)\right)^{-1} = K^{-1}\tilde{K}$. This yields

$$\begin{aligned}
\left\|K^{-1}\tilde{K}\right\|_2 &= \left\|I + \left(I - \tilde{K}^{-1}K\right) + \left(I - \tilde{K}^{-1}K\right)^2 + \cdots\cdots\right\|_2 \\
&\leq 1 + \left\|I - \tilde{K}^{-1}K\right\|_2 + \left\|\left(I - \tilde{K}^{-1}K\right)^2\right\|_2 + \cdots\cdots \\
&\leq 1 + \frac{\varepsilon}{\varepsilon_0 - \varepsilon} + \left(\frac{\varepsilon}{\varepsilon_0 - \varepsilon}\right)^2 + \cdots\cdots \\
&= \frac{\varepsilon_0 - \varepsilon}{\varepsilon_0 - 2\varepsilon}.
\end{aligned}$$

According to Theorem 4.6, we have

$$\left\|\tilde{K}^{-1}K\right\|_2 \leq \frac{\varepsilon_0}{\varepsilon_0 - \varepsilon},$$

then we obtain

$$\kappa_2(\tilde{K}^{-1}K) = \left\|\tilde{K}^{-1}K\right\|_2 \left\|K^{-1}\tilde{K}\right\|_2 \leq \frac{\varepsilon_0}{\varepsilon_0 - 2\varepsilon} \qquad \square$$

According to Theorem 4.6, we can also give an analytic bound on the spectrum of the preconditioned matrix.

Proposition 4.8 *Let a nonsingular matrix K be of the form* (4.1) *and let the condition ε in Definition 4.1 hold. If $\tilde{S}_i$ $(i = 1, 2, \ldots, N)$ are nonsingular, then we have the MSSS preconditioner $\tilde{K}$ is of the form* (4.3)*. Denote the eigenvalues of the preconditioned matrix by $\lambda(\tilde{K}^{-1}K)$. If $\varepsilon < \varepsilon_0$, we have*

$$\left|\lambda(\tilde{K}^{-1}K) - 1\right| \leq \frac{\varepsilon}{\varepsilon_0 - \varepsilon}.$$

Here ε_0 is the smallest singular value of K.

Proof: According to Theorem 4.6, we have

$$\left\|I - \tilde{K}^{-1}K\right\|_2 \leq \frac{\varepsilon}{\varepsilon_0 - \varepsilon}.$$

Therefore, we can obtain

$$\left|\lambda(I - \tilde{K}^{-1}K)\right| \leq \frac{\varepsilon}{\varepsilon_0 - \varepsilon},$$

owing to $\left|\lambda(I - \tilde{K}^{-1}K)\right| \leq \left\|I - \tilde{K}^{-1}K\right\|_2$. Since $\lambda(I - \tilde{K}^{-1}K) = 1 - \lambda(\tilde{K}^{-1}K)$, we get

$$\left|\lambda(\tilde{K}^{-1}K) - 1\right| \leq \frac{\varepsilon}{\varepsilon_0 - \varepsilon}. \qquad \square$$

Remark 4.9 Proposition 4.8 states that the spectrum of the preconditioned system is contained in a circle centered at $(1, 0)$ with a maximum radius $\frac{\varepsilon}{\varepsilon_0 - \varepsilon}$. Therefore, the smaller ε is, the closer the eigenvalues of the preconditioned system are to $(1, 0)$. This in turn gives better convergence for a wide class of Krylov solvers to solve the preconditioned system by applying the MSSS preconditioner.

According to Theorem 4.6, Proposition 4.7, and Proposition 4.8, we conclude that the smaller the value of ε is, the better-conditioned the preconditioned matrix is. For the extreme case $\varepsilon = 0$ when there is no approximation of the Schur complement, this factorization is exact. This is in turn verified by Theorem 4.6, Proposition 4.7, and Proposition 4.8. In Section 4.5, we will show that ε can be made arbitrarily small by setting a parameter in the MSSS preconditioner.

4.4 Breakdown Free Condition

In the previous section, we have analyzed the conditioning and spectrum of the preconditioned matrix. In this section, we discuss how to compute the MSSS preconditioner without breakdown, i.e., how to set the bound ε to compute the nonsingular $\tilde{S}_i$ $(i = 1, 2, \ldots, N)$. To start with, we give the following lemmas that are necessary for the analysis.

Lemma 4.10 ([60]) *Let A be an $m \times n$ matrix with, say, $m \geq n$. Sort its singular values in a non-increasing order by*

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n.$$

Let $\tilde{A} = A + E$ be a perturbation of A, and sort its singular values in a non-increasing order by

$$\tilde{\sigma}_1 \geq \tilde{\sigma}_2 \geq \cdots \geq \tilde{\sigma}_n.$$

Then, we have

$$|\tilde{\sigma}_i - \sigma_i| \leq \|E\|_2, \quad i = 1,\ 2,\ \cdots,\ n.$$

Proof: For the proof of this lemma, we refer to Corollary 8.6.2 of [60]. □

By applying this lemma, we have the following lemma that gives a sufficient condition for a nonsingular perturbation, i.e., for a full rank matrix A, its perturbed analog $\tilde{A}$ is still of full rank.

Lemma 4.11 *Let A be an $m \times n$ full rank matrix with, say, $m \geq n$, and $\tilde{A} = A + E$ be a perturbation of A. If*

$$\|E\|_2 < \sigma_n,$$

where σ_n is the smallest singular value of A, then the perturbed matrix $\tilde{A}$ is still of full rank.

Proof: Denote the smallest singular value of $\tilde{A}$ by $\tilde{\sigma}_n$, then according to Lemma 4.10, we have

$$|\sigma_n - \tilde{\sigma}_n| \leq \|E\|_2.$$

Since $\|E\|_2 < \sigma_n$, this yields,

$$|\sigma_n - \tilde{\sigma}_n| < \sigma_n.$$

We can obtain

$$0 < \tilde{\sigma}_n < 2\sigma_n,$$

which states that $\tilde{A}$ is still of full rank. □

With these lemmas, we can give a sufficient condition for ε that guarantees nonsingular approximate Schur complements $\tilde{S}_i$ $(i = 1,\ 2,\ \ldots,\ N)$, which satisfy condition ε in Definition 4.1.

Theorem 4.12 *Let K_p be the leading principle submatrix of K of size $pM \times pM$ for $p = 1,\ 2,\ \cdots\ N$, where M is the size of diagonal blocks of K. And let σ_0 be the minimum of all the smallest singular values of K_p for $p = 1,\ 2,\ \cdots\ N$, which is denoted by*

$$\sigma_0 \triangleq \min_{p=1}^{N} \Big\{ \min\Big(\sigma(K_p)\Big) \Big\}.$$

Here, we use the symbol " $\triangleq$ " to represent defined to be, and $\min(\sigma(K_p))$ denotes the smallest singular value of K_p. If ε satisfies the following inequality,

$$\varepsilon < \sigma_0,$$

then the condition ε can be satisfied and all the approximate Schur complements $\tilde{S}_i$ $(i = 1,\ 2,\ \ldots,\ N)$ are nonsingular.

Proof: At each step j $(j \geq 2)$ of the approximate factorization, we use $\tilde{S}_j$ to approximate $K_{j,j} - K_{j,j-1}\tilde{S}_{j-1}^{-1}K_{j-1,j}$ by performing a model order reduction according to Corollary 4.4 to satisfy the condition ε. This introduces a small perturbation E_j that is given by

$$E_j = \begin{cases} \tilde{S}_j - K_{j,j} & \text{if} \quad j = 1, \\ \tilde{S}_j - \left(K_{j,j} - K_{j,j-1}\tilde{S}_{j-1}^{-1}K_{j-1,j}\right) & \text{if} \quad 2 \leq j \leq N. \end{cases}$$

and $\|E_j\|_2 \leq \varepsilon$ $(2 \leq j \leq N)$. Since $S_1 = K_{1,1}$ is an SSS matrix with small off-diagonal rank, no model order reduction is performed at the first step, we have $E_1 = 0$.

Denote the $(j-1)$-th leading principle submatrix of K by $\bar{K}_{j-1}$, according to (4.5) we have

$$\bar{K}_j = \begin{bmatrix} \bar{K}_{j-1} & \bar{K}_{j-1,j} \\ \bar{K}_{j,j-1} & K_{j,j} \end{bmatrix}, \quad \tilde{\bar{K}}_j = \begin{bmatrix} \tilde{\bar{K}}_{j-1} & \bar{K}_{j-1,j} \\ \bar{K}_{j,j-1} & K_{j,j} + E_j \end{bmatrix}, \tag{4.6}$$

where $\bar{K}_j$ is the j-th principle leading sub-matrix of K and $\tilde{\bar{K}}_j$ is its approximation. $\tilde{\bar{K}}_j$ is also the j-th leading principle sub-matrix of $\tilde{K}$, and $\bar{K}_{j,j-1} = \begin{bmatrix} \mathbf{0} & K_{j,j-1} \end{bmatrix}$, $\bar{K}_{j-1,j} = \begin{bmatrix} \mathbf{0} \\ K_{j-1,j} \end{bmatrix}$, where $\mathbf{0}$ is a zero vector of suitable size.

Moreover, according to (4.5) we have

$$\tilde{\bar{K}}_j - \bar{K}_j = \begin{bmatrix} E_1 & & & \\ & E_2 & & \\ & & \ddots & \\ & & & E_j \end{bmatrix},$$

where $\|E_i\|_2 \leq \varepsilon$ $(1 \leq i \leq j)$. Then for any vector x of compatible size such that Kx can be performed, we can obtain,

$$\begin{aligned} \left\| \left(\tilde{\bar{K}}_j - \bar{K}_j\right) x \right\|_2^2 &= \|E_1 x_1\|_2^2 + \|E_2 x_2\|_2^2 + \cdots + \|E_j x_j\|_2^2 \\ &\leq \|E_1\|_2^2 \|x_1\|_2^2 + \|E_2\|_2^2 \|x_2\|_2^2 + \cdots + \|E_j\|_2^2 \|x_j\|_2^2 \\ &\leq \varepsilon^2 \sum_{k=1}^{j} \|x_k\|_2^2 = \varepsilon^2 \|x\|_2^2 . \end{aligned}$$

This in turn yields

$$\left\| \tilde{\bar{K}}_j - \bar{K}_j \right\|_2 \leq \varepsilon,\ j = 1,\ 2,\ \cdots,\ N.$$

According to Lemma 4.11, if $\varepsilon < \min_{j=1}^{N} \left\{ \min \left(\sigma(\bar{K}_j) \right) \right\}$, then $\tilde{\bar{K}}_j$ $(j = 1,\ 2,\ \cdots,\ N)$ is nonsingular.

Since $\tilde{\bar{K}}_j$ and $\tilde{\bar{K}}_{j-1}$ are nonsingular, according to (4.6), the Schur complement of $\tilde{\bar{K}}_{j-1}$ in $\tilde{\bar{K}}_j$ is also nonsingular and is given by

$$\tilde{\bar{S}}_j = K_{j,j} + E_j - \bar{K}_{j,j-1} \tilde{\bar{K}}_{j-1}^{-1} \bar{K}_{j-1,j},$$

which is exactly $\tilde{S}_j$ for $j = 2,\ \cdots,\ N$, while $\tilde{S}_1 = K_{1,1}$ is also nonsingular. □

Theorem 4.12 gives a sufficient condition for ε in order to compute nonsingular approximate Schur complements for a general class of linear systems. For the symmetric positive definite case, this sufficient condition can be simplified by the following lemma.

Lemma 4.13 *Let K be an $m \times m$ symmetric positive definite matrix of the form* (4.1) *and denote its smallest eigenvalue by $\lambda_{\min}(K)$. If $\varepsilon < \lambda_{\min}(K)$, then all the approximated Schur complements $\tilde{S}_i$ $(i = 1,\ 2,\ \ldots,\ N)$ are nonsingular.*

Before giving the proof of Lemma 4.13, we first introduce the following lemma that is necessary for the proof.

Lemma 4.14 (Corollary 8.4.6 in [19]) *Let A be an $m \times m$ Hermitian matrix, and A_0 be a $k \times k$ principle sub-matrix of A with $k < m$. Then,*

$$\lambda_{\min}(A) \le \lambda_{\min}(A_0) \le \lambda_{\max}(A_0) \le \lambda_{\max}(A),$$

and

$$\lambda_{\min}(A_0) \le \lambda_k(A).$$

With Lemma 4.14, we give the proof of Lemma 4.13 as follows.

Proof: According to Theorem 4.12, if $\varepsilon < \min_{p=1}^{N} \left\{ \min \left(\sigma(K_p) \right) \right\}$, the approximate Schur complements $\tilde{S}_i$ $(i = 1,\ 2,\ \ldots,\ N)$ are nonsingular. For the symmetric positive definite matrix K, its eigenvalues and singular values are identical. Then the condition for ε is given by

$$\varepsilon < \min_{p=1}^{N} \left\{ \min \left(\lambda(K_p) \right) \right\}.$$

According to Lemma 4.14, we have

$$\min \left(\lambda(K_p) \right) \ge \lambda_{\min}(K),$$

this in turn gives the condition for ε, that is

$$\varepsilon < \lambda_{\min}(K). \qquad \square$$

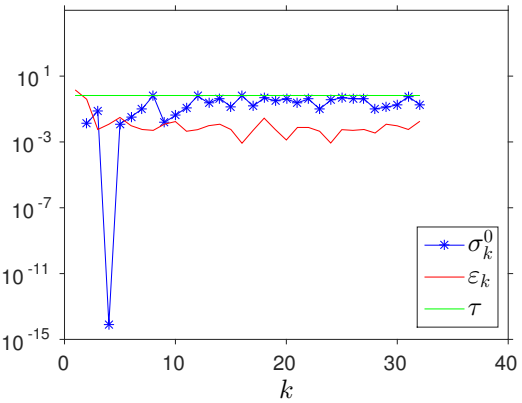
Theorem 4.12 gives a sufficient condition for ε in order to obtain nonsingular approximate Schur complements. Next, we use a simple example to illustrate this condition.

Example 4.1 *Consider the 2D stationary Schrödinger equation*

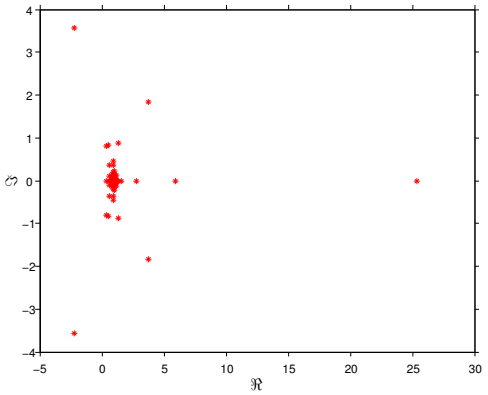
$$\nabla^2\Psi(x,y) + k^2\Psi(x,y) = 0,$$

with homogeneous Dirichlet boundary condition on a unit square domain $\Omega = [0,\ 1] \times [0,\ 1]$. *Using 5-point stencil finite difference discretization on a uniform grid with gird size* $h = 2^{-5}$ *gives a linear system that has a 2-level SSS structure, here* $k^2h^2 = {}^{\pi^2}/_{16}$. *Factorization of the linear system by using MSSS matrix computations that satisfy condition* ε *in Definition 4.1 gives an MSSS preconditioner* $\tilde{K}$.

For different settings of ε, the smallest singular value σ_k^0 of the leading principle sub-matrix K_k of size $kN \times kN$, the approximation error ε_k at each step to compute the approximate Schur complement, the bound of ε_k which is denoted by $\varepsilon_{\max}$, and the preconditioned spectrum are plotted in Figure 4.1 - 4.3.



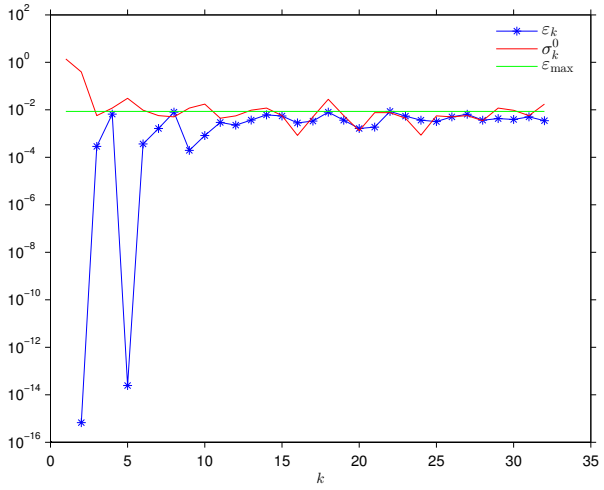
(a) Smallest singular values and ε_k



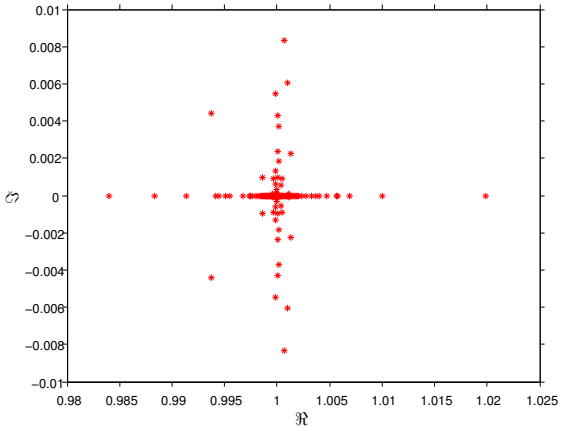
(b) Preconditioned spectrum

Figure 4.1: Condition ε and preconditioned spectrum for $\varepsilon = \mathcal{O}(0.5)$

We start decreasing ε from 0.5 to 10^{-3}, this corresponds to $\varepsilon > \sigma_0$ for relatively big ε. For the case $\varepsilon > \sigma_0$, Theorem 4.12 does not hold, which means that we may fail to get nonsingular approximate Schur complements $\tilde{S}_i$. However, we succeed in computing the nonsingular Schur complements $\tilde{S}_i$. In fact, the possibility of perturbing a matrix from nonsingularity to singularity is quite small. Although we get nonsingular approximate Schur complements for $\varepsilon > \sigma_0$, our analysis is not suited to analyzing the preconditioned spectrum for such case. This is illustrated by the spectrum of the preconditioned system in Figure 4.1(b). The preconditioned spectrum corresponds to $\varepsilon = \mathcal{O}(0.5)$ is not well clustered and a portion of the eigenvalues is far away from $(1, 0)$.



(a) Smallest singular values and ε_k

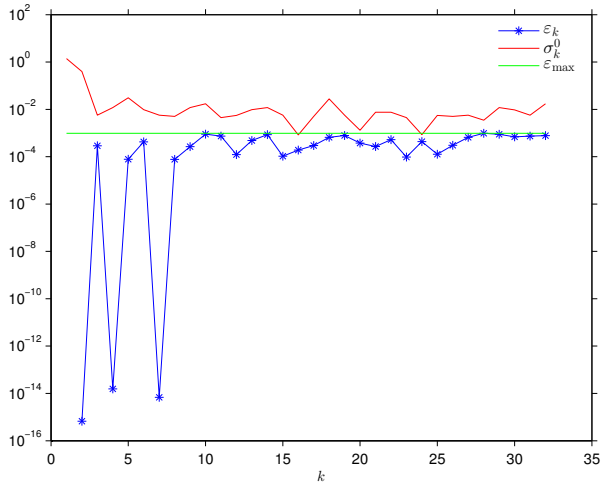


(b) Preconditioned spectrum

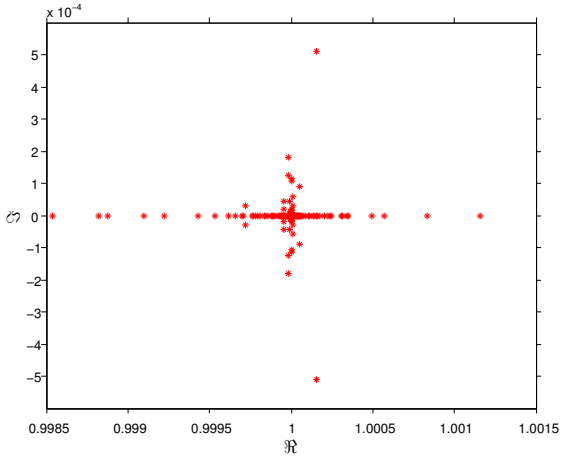
Figure 4.2: Condition ε and preconditioned spectrum for $\varepsilon = \mathcal{O}(10^{-2})$

For the cases that ε is slightly bigger than σ_0, the preconditioned spectrum is already contained in a quite small circle, cf. Figure 4.2(b). When ε is of the same order as σ_0, the circle is even smaller, cf. Figure 4.2(b) and Figure 4.3(b). Continue decreasing ε, the radius of the circle can be made arbitrarily small. At a certain

moment, the MSSS factorization can be used as a direct solver for small enough ε.



(a) Smallest singular values and ε_k



(b) Preconditioned spectrum

Figure 4.3: Condition ε and preconditioned spectrum for $\varepsilon = \mathcal{O}(10^{-3})$

Remark 4.15 We observed that for the case $\varepsilon < \sigma_0$ and ε is of the same order as σ_0, if ε decreases by a factor 10, the radius of the circle that contains the preconditioned spectrum is also reduced by a factor of around 10. This verifies the statement of the radius of the circle in Proposition 4.8.

The results for different ε given by Figure 4.1 - 4.3 verify our analysis of the convergence property of the MSSS preconditioner and the spectrum of the preconditioned system in Section 4.3. Both theoretical analysis and numerical experiments state that a small ε is preferred. Normally, decreasing ε will increase the computational complexity to a small extent. It is therefore favorable to choose a moderate ε to compute an MSSS factorization and use it as a preconditioner. This gives linear

computational complexity and satisfactory convergence for a wide class of Krylov solvers. Details will be discussed in Section 4.6.

Both theoretical analysis and numerical experiments indicate that a small ε is preferred. In the next section, we will discuss how to perform the model order reduction to make ε up to a prescribed accuracy.

4.5 Approximation Error of Model Order Reduction

We assume by Corollary 4.4 that the condition ε is satisfied via a model order reduction operation and the error of the model order reduction should be bounded by ε. To start, we use the model order reduction algorithm which is called Hankel blocks approximation that is studied in [33]. In the following part, we will show how to do this model order reduction to make the approximation error up to a prescribed accuracy ε. In this section, we use the algorithm style notation, i.e., by letting $a = b$, we assign the variable a with the value of b.

Recall from Chapter 2 that for a 4×4 block SSS matrix A, the Hankel blocks for the strictly lower-triangular part are shown in Figure 4.4 by $\mathcal{H}_2$, $\mathcal{H}_3$ and $\mathcal{H}_4$.

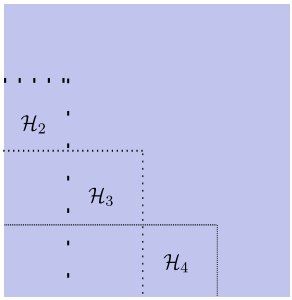


Figure 4.4: Hankel blocks of a 4×4 block SSS matrix

For the Hankel blocks $\mathcal{H}_k$ of the SSS matrix A, it has the following low-rank factorization,

$$\mathcal{H}_k = \mathcal{O}_k \mathcal{C}_k,$$

where the low-rank factors $\mathcal{O}_k$ and $\mathcal{C}_k$ have a backward and forward recursion, respectively. They are given by

$$\begin{cases} \mathcal{O}_k &= P_N, \text{ if } \quad k = N, \\ \mathcal{O}_k &= \begin{bmatrix} P_k \\ \mathcal{O}_{k+1} R_k \end{bmatrix}, \text{ if } \quad 2 \le k < N, \end{cases}$$

and

$$\begin{cases} \mathcal{C}_k &= Q_1, \text{ if } \quad k = 2, \\ \mathcal{C}_k &= \begin{bmatrix} R_{k-1} \mathcal{C}_{k-1} & Q_{k-1} \end{bmatrix}, \text{ if } \quad 2 < k \le N. \end{cases}$$

The rank r_k of the Hankel block $\mathcal{H}_k$ has the following equality

$$\operatorname{rank}(\mathcal{H}_k) = \operatorname{rank}(\mathcal{O}_k) = \operatorname{rank}(\mathcal{C}_k) = r_k.$$

The low-rank factors $\mathcal{O}_k$ and $\mathcal{C}_k$ are the observability factor and controllability factor of a linear time-varying (LTV) system that corresponds to the SSS matrix A. Moreover, the Hankel block $\mathcal{H}_k$ corresponds to the discrete Hankel map of a LTV system. SSS matrices and their relations with LTV system are studied in [41].

The basic idea for the model order reduction of SSS matrices is to reduce the rank of the Hankel blocks $\mathcal{H}_k$ from r_k to $\tilde{r}_k$ with $\tilde{r}_k < r_k$, where $\tilde{r}_k$ is the rank of the approximated Hankel map $\tilde{\mathcal{H}}_k$ and

$$\operatorname{rank}(\tilde{\mathcal{H}}_k) = \operatorname{rank}(\tilde{\mathcal{O}}_k) = \operatorname{rank}(\tilde{\mathcal{C}}_k) = \tilde{r}_k.$$

To start the model order reduction, first we need to transform $\mathcal{C}_i$ to the form that has orthonormal rows, which is called the right-proper form in [33]. This is obtained by performing a singular value decomposition (SVD) on $\mathcal{C}_i$. For $i = 2$,

$$\mathcal{C}_2 = U_1 \Sigma_1 V_1^T,$$

and let $\mathcal{C}_2 = Q_1 = V_1^T$. To keep the Hankel map (block) $\mathcal{H}_2$ unchanged, we let $\mathcal{O}_2 = \mathcal{O}_2 U_1 \Sigma_1$. This gives

$$P_2 = P_2 U_1 \Sigma_1, \; R_2 = R_2 U_1 \Sigma_1.$$

From step i to $i+1$, we have

$$\mathcal{C}_{i+1} = \begin{bmatrix} R_i \mathcal{C}_i & Q_i \end{bmatrix} = \begin{bmatrix} R_i & Q_i \end{bmatrix} \begin{bmatrix} \mathcal{C}_i & \\ & I \end{bmatrix}.$$

Since $\mathcal{C}_i$ has orthonormal rows, $\begin{bmatrix} \mathcal{C}_i & \\ & I \end{bmatrix}$ also has orthonormal rows. To complete this procedure for $\mathcal{C}_{i+1}$, perform the following SVD

$$\begin{bmatrix} R_i & Q_i \end{bmatrix} = U_i \Sigma_i V_i^T,$$

and let $\begin{bmatrix} R_i & Q_i \end{bmatrix} = V_i^T$. To keep the Hankel map at step $i+1$ unchanged, we let $\mathcal{O}_{i+1} = \mathcal{O}_{i+1} U_i \Sigma_i$. After finishing the above procedure, we can make all the factors $\mathcal{C}_i$ have orthonormal rows.

The next step of the model order reduction is to transform the low-rank factors $\mathcal{O}_i$ to the form with orthonormal columns, which is called the left-proper form [33]. While performing the transformation to the left proper form, we reduce the rank of the Hankel map (blocks). Since the recursion for the factor $\mathcal{O}_i$ is performed backward, we start from $i = N$.

First we approximate $\mathcal{O}_N$ by $\tilde{\mathcal{O}}_N$ using a low-rank approximation, this gives the factor $\tilde{\mathcal{O}}_{N-1}^1$ for the next step. Here $\tilde{\mathcal{O}}_{N-1}^1$ denotes the approximated factor $\mathcal{O}_{N-1}$ because of the propagation of the approximation error of $\mathcal{O}_N$. Then we compute a low-rank approximation of $\mathcal{O}_{N-1}^1$, which gives $\tilde{\mathcal{O}}_{N-1}^2$. We continue this procedure

till step $i = 2$. At step i, we use $\tilde{\mathcal{O}}_i^2$ to approximate $\tilde{\mathcal{O}}_i^1$, this introduces an approximation error that is bounded by τ. We use Figure 4.5 to depict this backward recursion of approximation. The details for the backward approximation of $\mathcal{O}_i$ and $\mathcal{H}_i$ are introduced in the proof of Lemma 4.16, which are given at Section 4.8 as appendix of this chapter. For the details of the Hankel blocks approximation algorithm, cf. [33, 103].

$$\begin{array}{ccccccccc} \mathcal{O}_N & \longrightarrow & \mathcal{O}_{N-1} & \longrightarrow & \mathcal{O}_{N-2} & \cdots & \longrightarrow & \mathcal{O}_2 \\ \downarrow\tau & & & & & & & \\ \tilde{\mathcal{O}}_N & \rightarrow & \tilde{\mathcal{O}}_{N-1}^1 & \xrightarrow{\tau} & \tilde{\mathcal{O}}_{N-1}^2 & \rightarrow & \tilde{\mathcal{O}}_{N-2}^1 \xrightarrow{\tau} \tilde{\mathcal{O}}_{N-2}^2 & \cdots \tilde{\mathcal{O}}_2^2 \end{array}$$

Figure 4.5: Low-rank approximation of $\mathcal{O}_k$

Here $\tilde{\mathcal{O}}_k^1$ represents the approximation of $\mathcal{O}_k$ by considering the propagation of the error introduced in the previous steps, and $\tilde{\mathcal{O}}_k^1$ is further approximated by $\tilde{\mathcal{O}}_k^2$ by performing a low-rank approximation. Then we have the following lemma that underlies the error between the original Hankel map and the approximate Hankel map.

Lemma 4.16 *Let A be a block $N \times N$ SSS matrix and its Hankel blocks $\mathcal{H}_i$ be approximated by $\tilde{\mathcal{H}}_i$ using the Hankel blocks approximation described by the above procedure, then*

$$\left\| \mathcal{H}_i - \tilde{\mathcal{H}}_i \right\|_2 \leq (N - i + 1)\tau, \quad 2 \leq i \leq N. \tag{4.7}$$

where τ is the upper bound for the discarded singular values that is applied in the singular value decomposition of the Hankel factors. For the approximated Hankel factors $\mathcal{O}_i$ that are illustrated in Figure 4.5, we have

$$\left\| \mathcal{O}_i - \tilde{\mathcal{O}}_i^1 \right\|_2 \leq (N - i)\tau, \quad 2 \leq i \leq N - 1. \tag{4.8}$$

and

$$\left\| \tilde{\mathcal{O}}_i^2 \tilde{Q}_{i-1} - \mathcal{O}_i Q_{i-1} \right\|_2 \leq (N - i + 1)\tau, \quad 2 \leq i \leq N. \tag{4.9}$$

Here we use the "˜ " notation to denote a factor or matrix after approximation.

Proof: For the proof of this lemma, cf. Section 4.8. □

Remark 4.17 To perform the Hankel blocks approximation to reduce the off-diagonal rank of an SSS matrix, the reduction of the strictly lower-triangular part is clearly covered by the analysis above. To reduce the off-diagonal rank of the strictly upper-triangular part, we can first transpose it to the strictly lower-triangular form. Then perform the Hankel blocks approximation to the strictly lower-triangular part and transpose back to the strictly upper-triangular form. This gives strictly upper-triangular part with reduced off-diagonal rank.

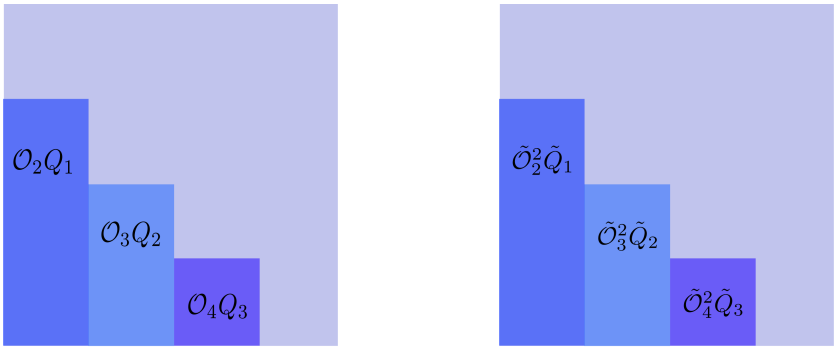
In Section 4.3, we gave an analytic bound of the radius of the circle that contains the preconditioned spectrum. This analytic bound is closely related to the approximation error ε by the model order reduction for SSS matrices, cf. Corollary 4.4 and Proposition 4.8. This model order reduction error ε can be made arbitrarily small by setting a parameter in the MSSS preconditioner. Now, we have all the ingredients to help to compute a controllable ε. Next, we will give the main theorem of this section to show how to compute the controllable ε.

Theorem 4.18 *Let the $N \times N$ block SSS matrix A be approximated by $\tilde{A}$ with lower off-diagonal rank using the Hankel blocks approximation, then*

$$\left\| A - \tilde{A} \right\|_2 \leq 2\sqrt{N}(N-1)\tau,$$

where τ is the upper bound of the discarded singular values for the singular value decomposition that is performed in approximating the Hankel blocks.

Proof: To prove this theorem, we use Figure 4.6 to illustrate the column structure of the off-diagonal blocks for an SSS matrix. Since the strictly upper-triangular part and the strictly lower-triangular part have similar structure, here we just take the strictly lower-triangular part for example.



(a) Hankel columns (b) Approximate Hankel columns

Figure 4.6: Lower-triangular Hankel columns before and after approximation

It is not difficult to verify that the i-th off-diagonal column of the strictly lower triangular part of an SSS matrix, denoted by C_i, can be represented by

$$C_i = \mathcal{O}_{i+1} Q_i, \quad (i = 1, 2, \cdots, N-1).$$

After performing the Hankel blocks approximation, C_i is approximated by

$$\tilde{C}_i = \tilde{\mathcal{O}}_{i+1}^2 \tilde{Q}_i, \quad (i = 1, 2, \cdots, N-1).$$

Denote $\Delta C_i = C_i - \tilde{C}_i$, then we have

$$\|\Delta C_i\|_2 = \left\| \mathcal{O}_{i+1} Q_i - \tilde{\mathcal{O}}_{i+1}^2 \tilde{Q}_i \right\|_2 \leq (N-i)\tau. \quad \text{(Lemma 4.16)}$$

We can write the SSS matrix A by $A = L + D + U$, where L is the strictly lower-triangular part, D is the block diagonal part, and U is the strictly upper-triangular part of A, respectively. Performing the Hankel blocks approximation on the strictly lower-triangular part and the strictly upper-triangular part, we obtain

$$\tilde{A} = \tilde{L} + D + \tilde{U},$$

where $\tilde{L}$ and $\tilde{U}$ are approximated independently. Moreover, we have

$$\tilde{L} - L = \begin{bmatrix} \Delta C_1 & \Delta C_2 & \cdots & \Delta C_{N-1} & \mathbf{0} \end{bmatrix}.$$

And

$$\begin{aligned} \left\| \left(\tilde{L} - L \right) x \right\|_2 = \left\| \sum_{i=1}^{N-1} \Delta C_i x_i \right\|_2 &\leq \sum_{i=1}^{N-1} \|\Delta C_i\|_2 \|x_i\|_2 \leq \max_{i=1}^{N-1} \|\Delta C_i\|_2 \sum_{i=1}^{N-1} \|x_i\|_2 \\ &\leq (N-1)\tau \sum_{i=1}^{N-1} \|x_i\|_2 \leq (N-1)\tau \sum_{i=1}^{N} \|x_i\|_2 \\ &\leq (N-1)\tau \sqrt{N \sum_{i=1}^{N} \|x_i\|_2^2} = \sqrt{N}(N-1)\tau \|x\|_2 . \end{aligned}$$

This yields

$$\left\| L - \tilde{L} \right\|_2 \triangleq \max_{x \neq \mathbf{0}} \frac{\left\| (L - \tilde{L})x \right\|_2}{\|x\|_2} \leq \sqrt{N}(N-1)\tau.$$

Here τ is the upper bound for the discarded singular values for the singular value decomposition that is performed in the Hankel blocks approximation. Similarly, we have $\left\| U - \tilde{U} \right\|_2 \leq \sqrt{N}(N-1)\tau$, this gives

$$\begin{aligned} \left\| A - \tilde{A} \right\|_2 &= \left\| (L - \tilde{L}) + (U - \tilde{U}) \right\|_2 \\ &\leq \left\| L - \tilde{L} \right\|_2 + \left\| U - \tilde{U} \right\|_2 \\ &\leq 2\sqrt{N}(N-1)\tau. \end{aligned}$$

□

Remark 4.19 Theorem 4.18 gives an analytical bound of the error introduced by the model order reduction. This error is only related to the maximum discarded singular value τ for the singular value decomposition that is performed in the Hankel blocks approximation and the matrix dimension. This states that this approximation error can be made arbitrarily small by setting τ small enough. This in turn gives a relatively bigger off-diagonal rank compared with moderate τ. A trade-off has to be made between the computational complexity and accuracy.

Remark 4.20 The model order reduction can be also performed by setting a fixed reduced off-diagonal rank. This is convenient in practice and makes the computational complexity and memory consumption easily predictable. The disadvantage,

however, is that the approximation error is unpredictable. In contrast, by setting a fixed τ for the model order reduction, we can easily control the error bound and get an adaptive reduced off-diagonal rank. However, the disadvantage is that the computational complexity is difficult to estimate. In practice, we observe that for many applications, a properly chosen τ also gives small enough off-diagonal rank, which in turn gives predictable computational complexity. This will be highlighted in the numerical experiments part.

Remark 4.21 In many applications, the error introduced by the model order reduction using the Hankel blocks approximation is of $\mathcal{O}(\tau)$, which is quite small compared with the bound given by Theorem 4.18. Only in some extreme cases, the bound given by Theorem 4.18 is sharp. However, it is quite difficult to prove for which case the error bound given by Theorem 4.18 is sharp. Normally, a small τ still results a small reduced off-diagonal rank, which yields linear computational complexity. This will be illustrated by numerical experiments in the next section.

Remark 4.22 The Hankel-norm approximation algorithm [125], which is more computationally expensive than the Hankel blocks approximation [33], approximates the Hankel blocks with optimality in the Hankel norm. This makes the Hankel-norm approximation appealing when using MSSS *LU* factorization as a direct solver.

In practice, it would be desirable to estimate the semiseparable order of the Schur complement that corresponds to a given τ. Normally, this is quite challenging since the off-diagonal rank depends not only on the differential operator of the PDE, but also on the coefficients of the PDE. Only some preliminary results can be found in the literature. These results are summarized by Lemma 4.23.

Lemma 4.23 ([35]) *Let the symmetric positive definite block tridiagonal system K arise from the discretization of PDEs with Laplacian operator, constant coefficients, and Dirichlet boundary condition everywhere on the boundary. Then the Schur complement S_i has a monotonic convergence rate and the limit of S_i, i.e., S_∞ is also symmetric positive definite. The τ-rank of the Hankel blocks of S_∞ are bounded by*

$$r\left(1 + 8\ln^4\left(\frac{3\,\|D\|}{\tau}\right)\right),$$

where r is the maximal Hankel block rank of $K_{i,i}$ and $K_{i,i-1}$, D is the diagonal block of K. Here, the τ-rank of a matrix is defined by the number of singular values that are bigger than or equal to τ.

Lemma 4.23 gives an upper bound on the limit of the Schur complement for the infinite dimensional symmetric positive definite systems. For finite dimensional symmetric positive definite systems with constant coefficients, similar results hold. For detailed discussion, cf. [35]. Note that this bound is not sharp because the term $\ln^4\left(\frac{3\,\|D\|}{\tau}\right)$ can be much bigger than the size of K.

Recall Lemma 4.13 states that for the symmetric positive definite system K, τ can be often chosen as $\tau < \lambda_{\min}(K)$. In Example 4.1, it is shown that usually we can choose $\tau = \mathcal{O}(\lambda_{\min}(K))$. If $\|D\| = \mathcal{O}(\|K\|)$, then we get the bound of the rank of the Hankel blocks is of $\mathcal{O}\left(r \ln^4 \kappa_2(K)\right)$. Even this bound is not sharp, it states that for ill-conditioned system, a bigger semiseparable order is needed to get a considerably good approximation, which will be shown in Section 4.6.

For the symmetric positive definite systems from the discretization of PDEs with variable coefficients and the indefinite systems, the analytic bound of the rank of the Hankel blocks of the Schur complement is quite difficult to analyze. Relevant work on analyzing the off-diagonal rank of the Schur complement of the symmetric positive definite type by using hierarchical matrix computations is done in [12].

Remark 4.24 The τ rank of the off-diagonal blocks of the Schur complement for symmetric positive definite systems studied in Lemma 4.23 is not sharp. In many applications, it can be made quite small and even bounded by a small number for a wide class of linear systems. This will be illustrated by numerical experiments in the next section.

4.6 Numerical Experiments

In this section, we use numerical experiments to investigate our analysis in the previous sections. We use three types of experiments, which include unsymmetric systems, symmetric indefinite systems from discretization of scalar PDEs and saddle-point systems, to demonstrate our results. For all the numerical experiments performed in this section, the induced dimension reduction (`IDR(s)`)[127] is used as a Krylov solver. The `IDR(s)` solver is terminated when the 2-norm of the residual is reduced by a factor of 10^{-6}. 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 8.0 system.

4.6.1 Unsymmetric System

In this subsection, we use the convection-diffusion equation as a numerical example to demonstrate our analysis for the unsymmetric case. The convection-diffusion problem is described by Example 4.2, which is given as the example 3.1.4 in [53]. The details of the discretization of the convection-diffusion equation can be also found in [53]. We generate the linear system in this example using the Incompressible Flow and Iterative Solver Software [117] (IFISS). To investigate the performance of the MSSS preconditioning technique, we consider the case for a moderate ν and a small ν.

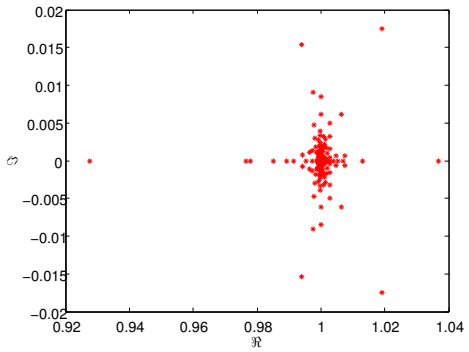
Example 4.2 ([53]) *Zero source term, recirculating wind, characteristic boundary layers.*

$$\begin{aligned} -\nu\nabla^2 u + \overrightarrow{\omega}\cdot\nabla u &= f \quad in \quad \Omega \\ u &= u_D \quad on \quad \partial\Omega \end{aligned} \tag{4.10}$$

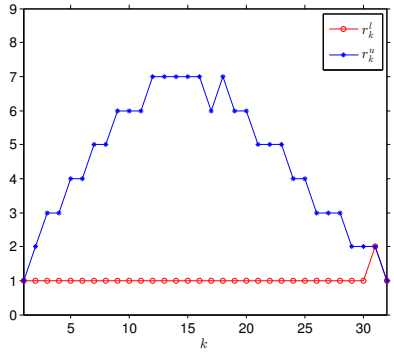
where $\Omega = \{(x,y)| -1 < x < 1, -1 < y < 1\}$, $\overrightarrow{\omega} = \big(2y(1-x^2),\ -2x(1-y^2)\big)$, $f = 0$. *Homogeneous Dirichlet boundary are imposed everywhere except that* $u_D = 1$ *on* $\big[-1,1\big] \times 1$.

We use the Q_1 finite element method to discretize the convection-diffusion equation. First, we consider a moderate value for the viscosity parameter $\nu = {}^1\!/_{200}$.

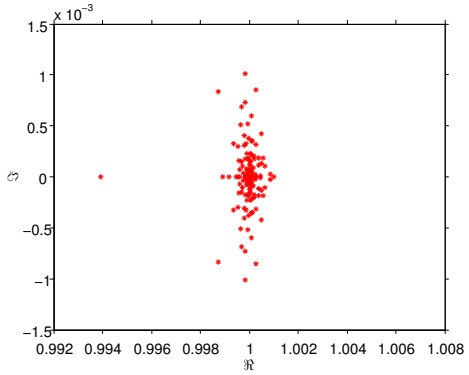
According to Proposition 4.8, the preconditioned spectrum is contained in a circle centered at $(1,0)$ and the radius of this circle is directly related to the approximation error ε introduced by the model order reduction for SSS matrix computations. In Section 4.5, we show that ε can be made arbitrarily small by setting the bound of the discarded singular values τ properly. We give detailed information of the spectrum of the preconditioned matrix of dimension 1089×1089, which corresponds to a mesh size $h = 2^{-4}$. For different values of τ, the preconditioned spectrum and the adaptive semiseparable order are plotted in Figure 4.7. For all the numerical results that show the adaptive semiseparable order in this chapter, we use red line to illustrate the lower-semiseparable order r_k^l and blue line to indicate the upper-semiseparable order r_k^u for unsymmetric linear systems while the red line is used to indicate the semiseparable order r_k for symmetric systems.



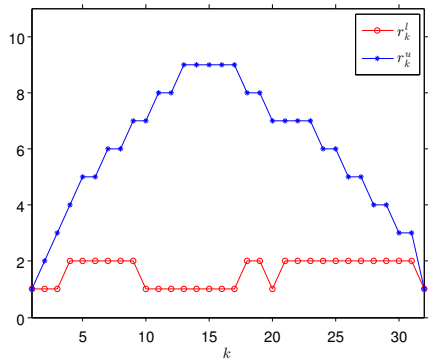
(a) Preconditioned spectrum, $\tau = 10^{-3}$



(b) Adaptive semiseparable order, $\tau = 10^{-3}$



(c) Preconditioned spectrum, $\tau = 10^{-4}$



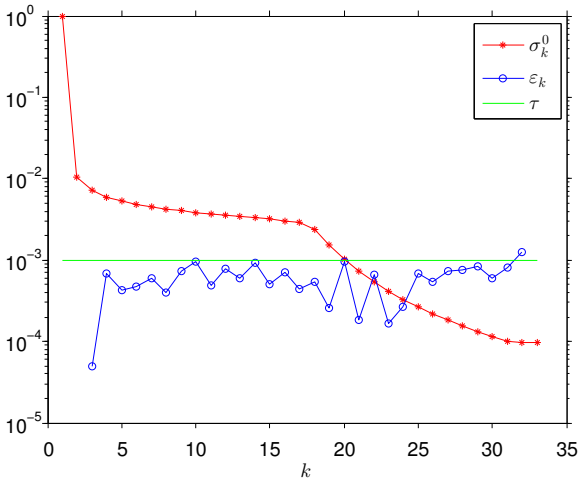
(d) Adaptive semiseparable order, $\tau = 10^{-4}$

Figure 4.7: Preconditioned spectrum and adaptive semiseparable order

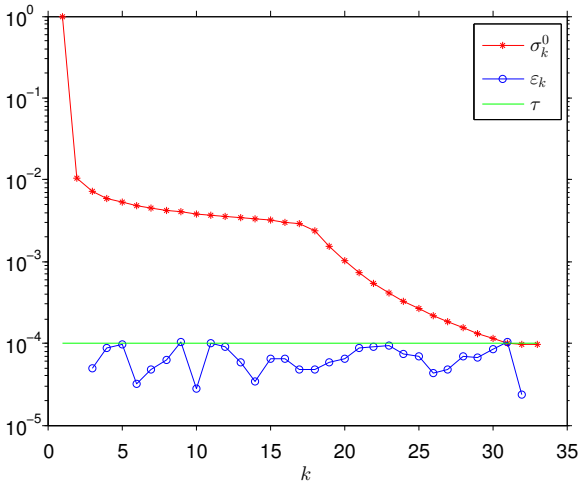
Figure 4.8(a) and Figure 4.8(b) illustrate that the error introduced by the model order reduction at step k in computing the MSSS preconditioner, which is denoted by ε_k and measured by the matrix 2-norm, is of the same order as τ. Here ε_k is computed by

$$\varepsilon_k = \left\| \tilde{S}_k - \left(K_{k,k} - K_{k,k-1} \tilde{S}_{k-1}^{-1} K_{k-1,k} \right) \right\|_2 . \tag{4.11}$$

It also illustrates that by setting the approximation error of the model order reduction with the same order as the smallest singular value σ_0, we can compute a nonsingular preconditioner and get satisfactory convergence.



(a) $\tau = 10^{-3}$



(b) $\tau = 10^{-4}$

Figure 4.8: Minimal singular value and ε_k

By decreasing τ, we get a smaller approximation error, which corresponds to smaller ε. According to our analysis, the circle that contains the preconditioned spectrum

is even smaller. This is depicted by Figure 4.7(a) and Figure 4.7(c). Moreover, it is shown that by decreasing τ by a factor of 10, the radius of the circle that contains the preconditioned spectrum also decreases by a factor of about 10. This validates our bound of the radius in Proposition 4.8. In fact, for the preconditioned spectrum in Figure 4.7(a), only 4 iterations are needed to compute the solution by the `IDR(4)` solver and only 2 iterations are needed to solve the system corresponding to the preconditioned spectrum in Figure 4.7(c).

Figure 4.7(b) and Figure 4.7(d) give the maximum adaptive rank for the off-diagonal blocks of the Schur complement at step k to compute the MSSS preconditioner. Since we have an unsymmetric matrix, the τ-rank for the upper-triangular part and the lower-triangular part are different from each other. Here the τ-rank represents the number of singular values that is bigger than or equal to τ for a matrix. Figure 4.7(b) and Figure 4.7(d) illustrate that the upper semiseparable order r^u is bigger than the lower semiseparable order r^l which states that the upper-triangular part is more difficult to approximate. Both r^l and r^u are small and this gives small average semiseparable order, which yields linear computational complexity.

We plot the spectrum of the system without preconditioning in Figure 4.9 to compare with the preconditioned spectrum in Figure 4.7(a) and Figure 4.7(c).

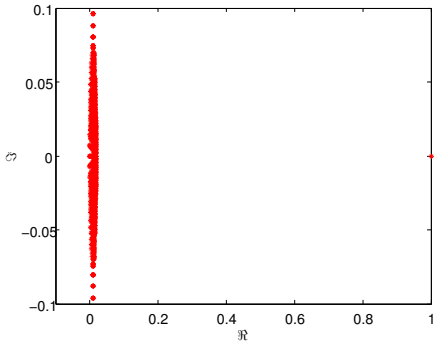
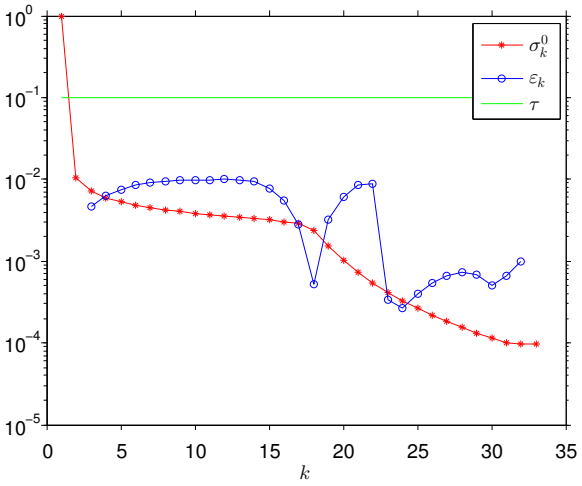


Figure 4.9: Spectrum of the system without preconditioning

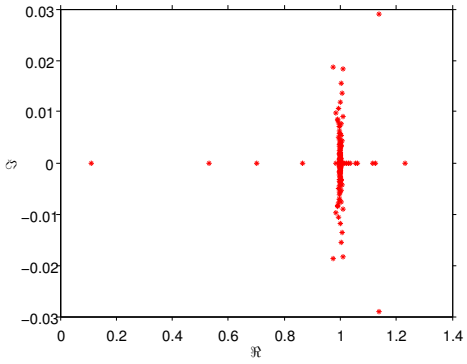
The driving force of preconditioning is to push the eigenvalues away from 0 and make them cluster. We have already seen that moderate or small setting of the model reduction error give satisfactory results. Next, we use a big model order reduction error bound by setting $\tau = 10^{-1}$ and test the performance of the MSSS preconditioner. The approximation error at each step in computing the MSSS preconditioner is plotted in Figure 4.10(a), the preconditioned spectrum is given in Figure 4.10(b), and the adaptive semiseparable order is given in Figure 4.10(c).

Note that even this setting of the error bound is much bigger than the smallest singular value of the leading principle sub-matrices, which is used to guarantee to compute a nonsingular preconditioner, we still compute an MSSS preconditioner. This is because the possibility of perturbing a nonsingular matrix to singularity is quite small. Since we have a preconditioner that is less accurate because we use a relatively big error bound, the radius of the circle that contains the spectrum of the preconditioned matrix in Figure 4.10(b) is not as small as the radius in

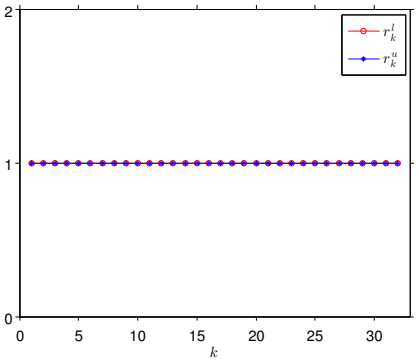
Figure 4.7(a) and Figure 4.7(c). However, the spectrum is away from 0 and only a few eigenvalues are out of this cluster. `IDR(4)` computes the solution in just 8 iterations. Moreover, the semiseparable order for such computations is just 1 as shown in Figure 4.10(c), which makes the computational complexity even smaller.



(a) Minimal singular value and ε_k



(b) Preconditioned spectrum

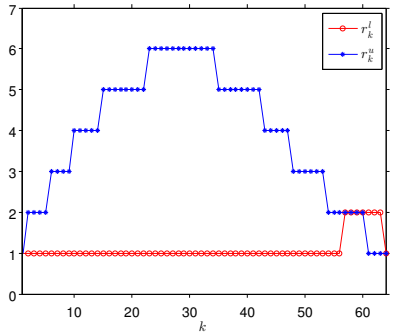


(c) Adaptive semiseparable order

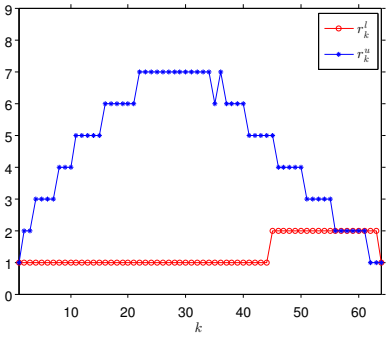
Figure 4.10: Preconditioned spectrum and adaptive semiseparable order for $\tau = 10^{-1}$

The performance of MSSS preconditioners for different mesh sizes h and τ are reported in Table 4.1. For different settings of τ, the adaptive semiseparable order is given in Figure 4.11.

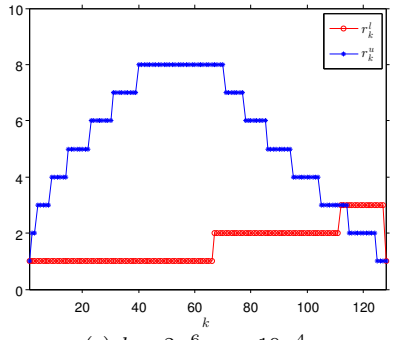
The results reported in Table 4.1 and Figure 4.11 state that by choosing a proper error bound for the model order reduction, we can compute an MSSS preconditioner that gives satisfactory convergence. This convergence can be made independent of the mesh size, while the adaptive semiseparable order only slightly increases with the problem size. This is demonstrated by Figure 4.11. The average of the upper and lower semiseparable order is still quite small and can be almost kept constant. We can also choose a fixed semiseparable order to compute an MSSS preconditioner. This is studied in [104]. Both ways to compute the MSSS preconditioner give satisfactory results.



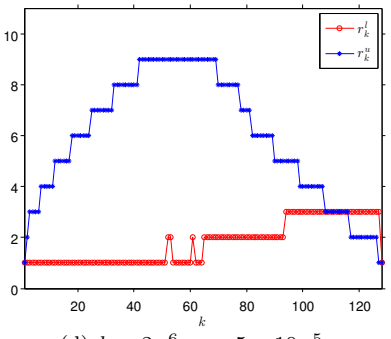
(a) $h = 2^{-5}$, $\tau = 10^{-3}$



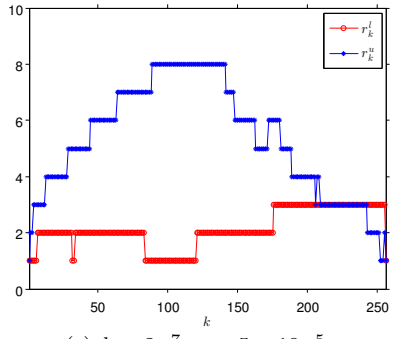
(b) $h = 2^{-5}$, $\tau = 5 \times 10^{-4}$



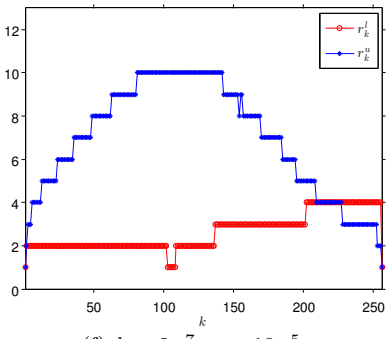
(c) $h = 2^{-6}$, $\tau = 10^{-4}$



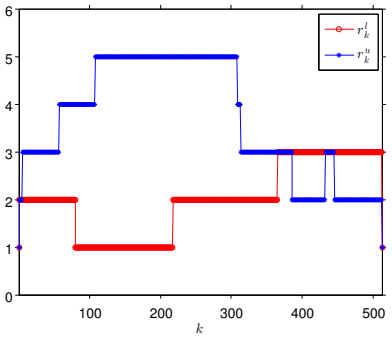
(d) $h = 2^{-6}$, $\tau = 5 \times 10^{-5}$



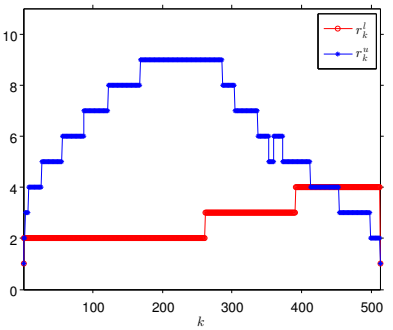
(e) $h = 2^{-7}$, $\tau = 5 \times 10^{-5}$



(f) $h = 2^{-7}$, $\tau = 10^{-5}$



(g) $h = 2^{-8}$, $\tau = 5 \times 10^{-5}$

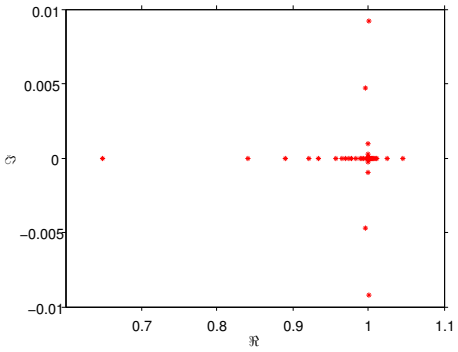


(h) $h = 2^{-8}$, $\tau = 10^{-5}$

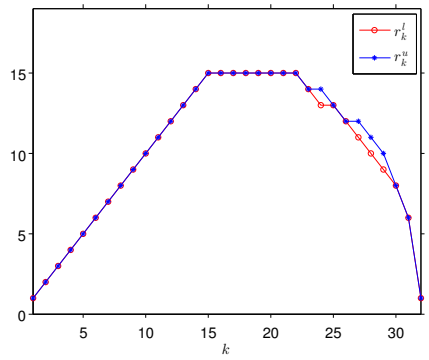
Figure 4.11: Adaptive semiseparable order for convection-diffusion equation with $\nu = {}^1/_{200}$

Table 4.1: MSSS preconditioner for the convection-diffusion equation with $\nu = {}^1/_{200}$

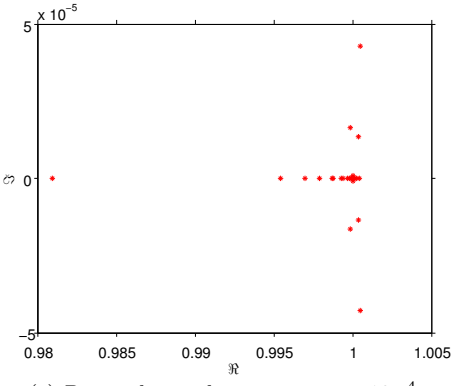
h	N^2	τ	# iter.
2^{-4}	$1.09e+03$	5×10^{-3}	6
		10^{-3}	3
2^{-5}	$4.23e+03$	10^{-3}	7
		5×10^{-4}	4
2^{-6}	$1.66e+04$	10^{-4}	5
		5×10^{-5}	4
2^{-7}	$6.61e+04$	5×10^{-5}	6
		10^{-5}	3
2^{-8}	$2.63e+05$	5×10^{-5}	10
		10^{-5}	5



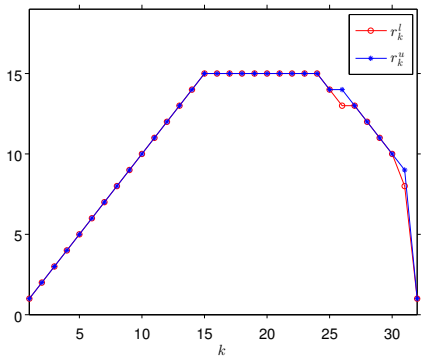
(a) Preconditioned spectrum, $\tau = 10^{-3}$



(b) Adaptive semiseparable order, $\tau = 10^{-3}$



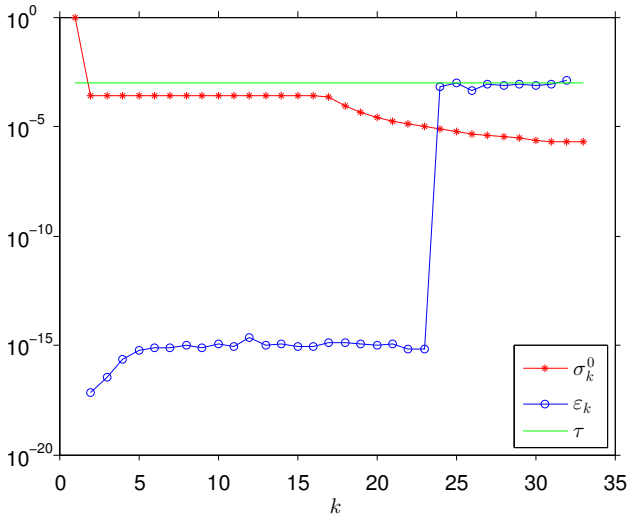
(c) Preconditioned spectrum, $\tau = 10^{-4}$



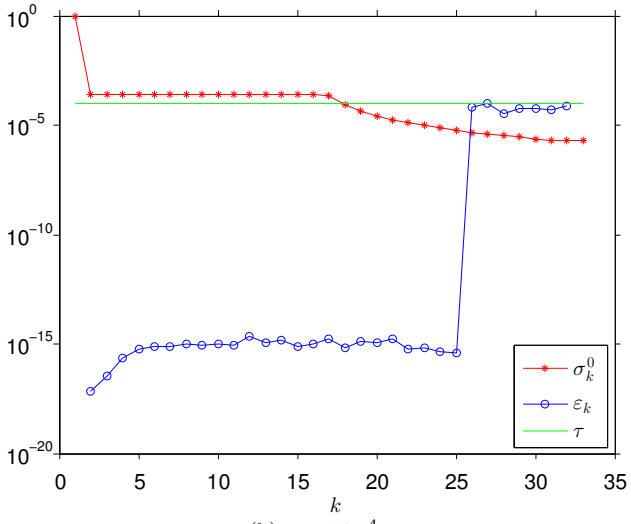
(d) Adaptive semiseparable order, $\tau = 10^{-4}$

Figure 4.12: Preconditioned spectrum and adaptive semiseparable order

Next we set $\nu = 10^{-4}$ for the convection-diffusion equation, which corresponds to the convection-dominated case. For such test case, the finite element discretization is not stable anymore, an up-wind scheme should be applied to get a stable discretization. Due to the ill-conditioning of the system, a bigger semiseparable order is needed to compute the MSSS preconditioner to get better performance. Note that for this case, the multigrid methods (both AMG and GMG) fail to solve such system without up-wind scheme while the MSSS preconditioner can still solve these ill-conditioned systems [104]. Here we report detailed numerical results for the test case with mesh size $h = 2^{-4}$. We first set $\tau = 10^{-3}$ and $\tau = 10^{-4}$, the computational results are reported in Figure 4.12 and Figure 4.13.



(a) $\tau = 10^{-3}$



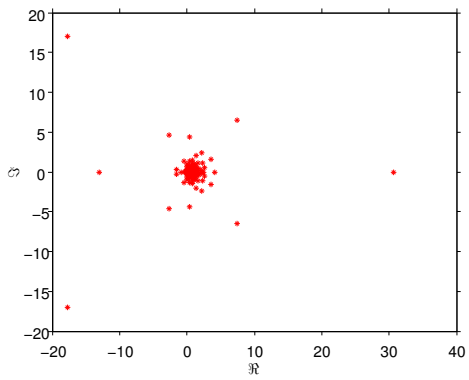
(b) $\tau = 10^{-4}$

Figure 4.13: Minimal singular value and ε_k

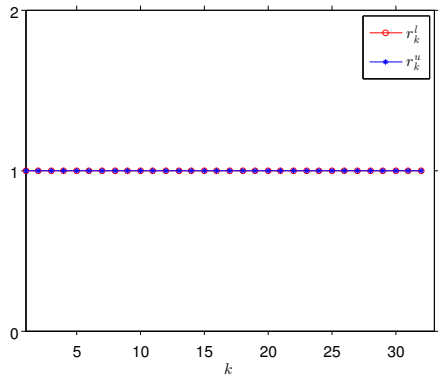
Figure 4.12(a) and Figure 4.12(c) show that by reducing τ with a factor of 10, the radius of the circle that contains the preconditioned spectrum is also reduced by a factor around 10. This validates our bound of the radius of circle that contains the preconditioned spectrum in Proposition 4.8 again. The preconditioned system can be solved by `IDR(4)` using 4 iterations for $\tau = 10^{-3}$ and 2 iterations for $\tau = 10^{-4}$.

Remark 4.25 The MSSS preconditioners computed by setting $\tau = 10^{-3}$ and $\tau = 10^{-4}$ give different clustering of the spectrum, as shown in Figure 4.12(a) and Figure 4.12(c). However, the adaptive semiseparable order for different τ is almost the same. This is primarily because the Schur complements in computing the factorization are very ill-conditioned and difficult to approximate. A slight change of the semiseparable order results in relatively big difference of the approximation accuracy. This also explains why a bigger adaptive semiseparable order is needed compared with the test case for $\nu = {}^{1}/_{200}$.

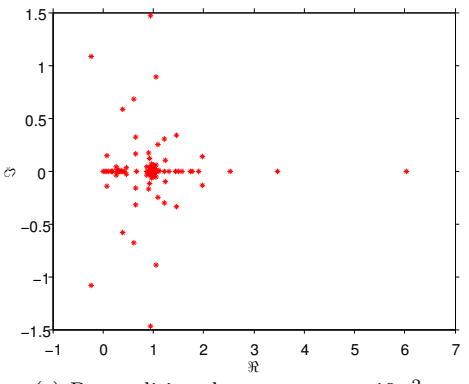
We also test the convergence of the MSSS preconditioned system by setting $\tau = 10^{-1}$ and $\tau = 10^{-2}$. The computational results are given in Figure 4.14 - 4.15.



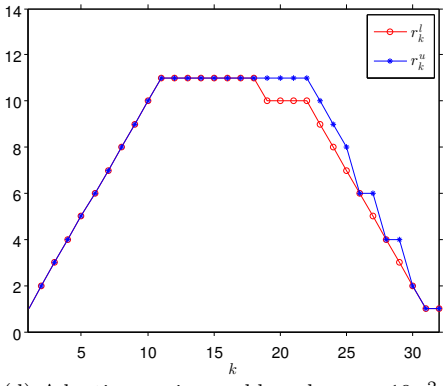
(a) Preconditioned spectrum, $\tau = 10^{-1}$



(b) Adaptive semiseparable order, $\tau = 10^{-1}$



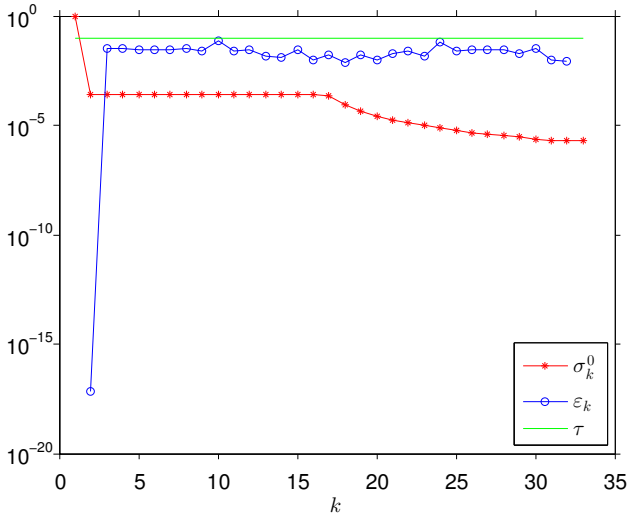
(c) Preconditioned spectrum, $\tau = 10^{-2}$



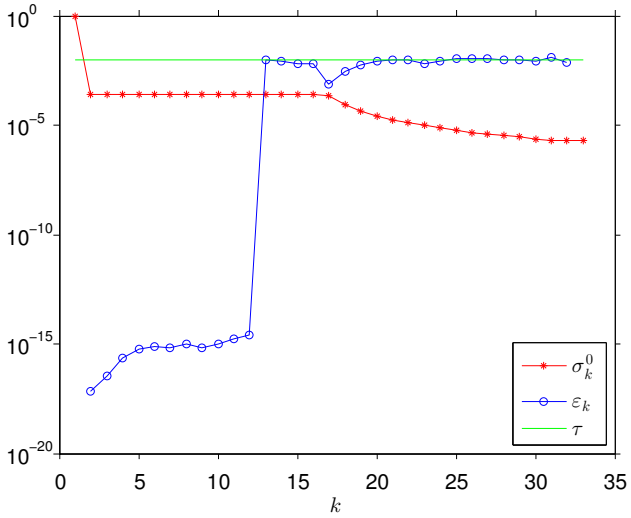
(d) Adaptive semiseparable order, $\tau = 10^{-2}$

Figure 4.14: Preconditioned spectrum and adaptive semiseparable order for $\tau = 10^{-1}$

It is shown in Figure 4.14 - 4.15 that although the MSSS preconditioner is nonsingular by choosing relatively bigger τ compared to σ_0, the preconditioned spectrum is contained in a circle centered at $(1, 0)$ with a much bigger radius and $(0, 0)$ is included in the circle. The preconditioned system for $\tau = 10^{-2}$ is solved by `IDR(4)` in 61 iterations while `IDR(4)` fails to solve the preconditioned system for $\tau = 10^{-1}$ within 80 iterations.



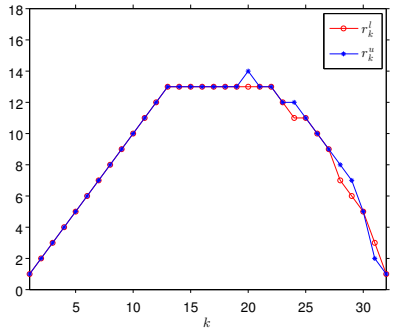
(a) $\tau = 10^{-1}$



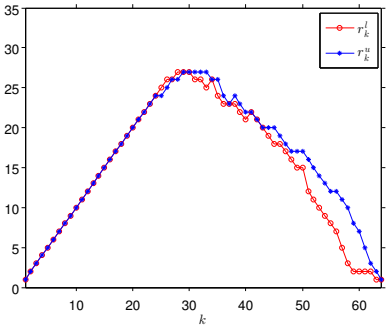
(b) $\tau = 10^{-2}$

Figure 4.15: Minimal singular value and ε_k

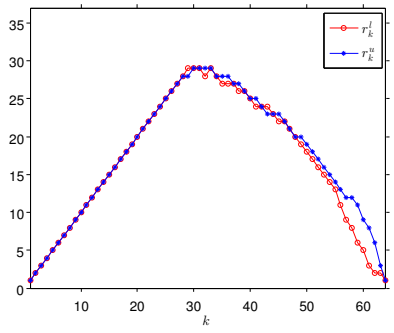
For different mesh sizes h and settings of τ, the computational results are reported in Table 4.2. The adaptive semiseparable order for different mesh sizes h and settings of τ are plotted in Figure 4.16.



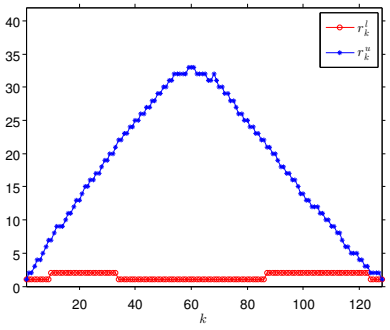
(a) $h = 2^{-4}$, $\tau = 5 \times 10^{-3}$



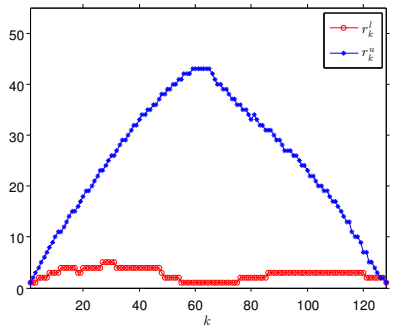
(b) $h = 2^{-5}$, $\tau = 10^{-3}$



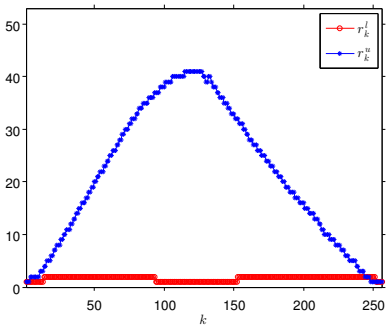
(c) $h = 2^{-5}$, $\tau = 5 \times 10^{-4}$



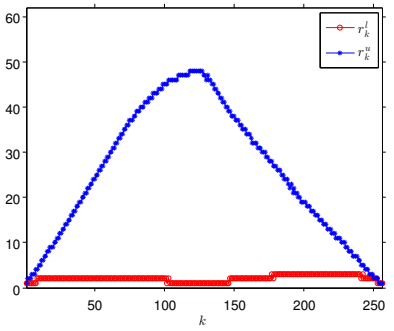
(d) $h = 2^{-6}$, $\tau = 5 \times 10^{-4}$



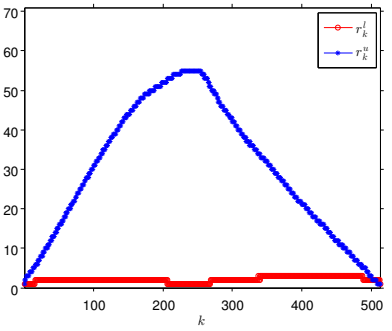
(e) $h = 2^{-6}$, $\tau = 10^{-4}$



(f) $h = 2^{-7}$, $\tau = 10^{-4}$



(g) $h = 2^{-7}$, $\tau = 5 \times 10^{-5}$



(h) $h = 2^{-8}$, $\tau = 10^{-5}$

Figure 4.16: Adaptive semiseparable order for convection-diffusion equation with $\nu = 10^{-4}$

Table 4.2: Computational results of the MSSS preconditioner for the convection-diffusion equation with $\nu = 10^{-4}$

h	N^2	τ	# iter.
2^{-4}	$1.09e+03$	5×10^{-3}	23
		10^{-3}	4
2^{-5}	$4.23e+03$	10^{-3}	18
		5×10^{-4}	11
2^{-6}	$1.66e+04$	5×10^{-4}	20
		10^{-4}	6
2^{-7}	$6.61e+04$	10^{-4}	13
		5×10^{-5}	8
2^{-8}	$2.63e+05$	5×10^{-5}	17
		10^{-5}	6

Since this convection-dominated test case is ill-conditioned, it is quite difficult to compute its factorization (inverse). It takes more effort to compute a good approximation compared with the case $\nu = 1/200$. This is illustrated by comparing the adaptive semiseparable order in Figure 4.16 with that in Figure 4.11. For such case, the average of the upper and lower semiseparable order is considerably bigger. Since the average semiseparable order may not be bounded by a small constant, this makes the computational complexity of the MSSS preconditioning technique slightly bigger than linear. Details and remarks on the computational complexity for moderate semiseparable order will be discussed in Section 4.6.2.

4.6.2 Symmetric Indefinite Systems from Discretized Helmholtz Equation

In this subsection, we study the convergence of the MSSS preconditioners for the symmetric indefinite systems from the discretization of scalar PDEs, where the Schrödinger equation in Example 4.1 and the Helmholtz equation belong to this type. In this part, we mainly focus on the performance of the MSSS preconditioner for the Helmholtz equation that is given by Example 4.3.

Example 4.3 ([91]) *Consider the following Helmholtz equation,*

$$-\nabla^2 u(x,\omega) - \frac{\omega^2}{c^2(x)} u(x,\omega) = g(x,\omega), \ \ x \in [0,1] \times [0,1],$$

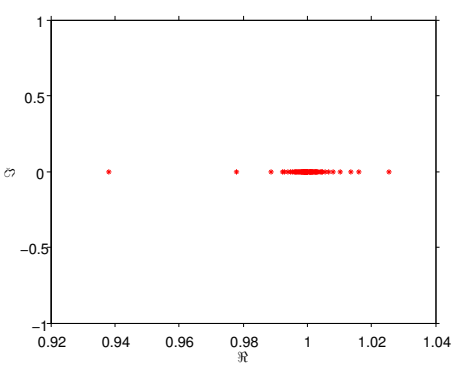
with homogeneous Dirichlet boundary condition. Here $u(x,\omega)$ represents the pressure field in the frequency domain, ∇^2 is the Laplacian operator, ω is the angular frequency, and $c(x)$, is the acoustic-wave velocity that varies with position x.

Standard five-point stencil finite difference method is used to discretize the Helmholtz equation. We use the rule of thumb that at least 10 nodes per wavelength should be employed, which leads to the restriction

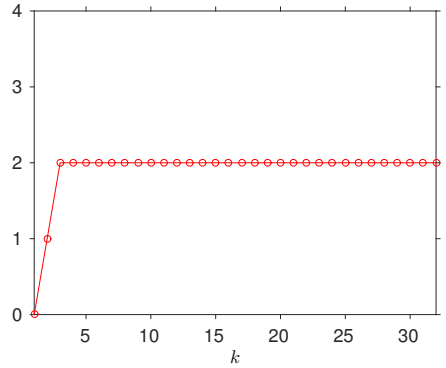
$$\kappa h \leq \frac{\pi}{5} \approx 0.628, \tag{4.12}$$

for the standard five-point stencil finite difference discretization [91]. Here $\kappa = {}^{\omega}/_{c(x)}$ is the wave number. We apply the MSSS preconditioner to the Helmholtz equation. The pulse source term $g(x, \omega)$ is chosen as the scaled delta function that is located at $\left({}^{1}/_{32}, {}^{1}/_{2}\right)$.

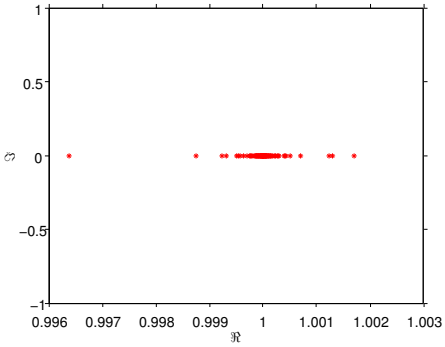
To test the performance of the MSSS preconditioner, we first set a moderate value for κh, say ${}^{1}/_{16}$. The preconditioned spectrum and semiseparable order for mesh size $h = 2^{-5}$ and different settings of τ are plotted in Figure 4.17 and Figure 4.18.



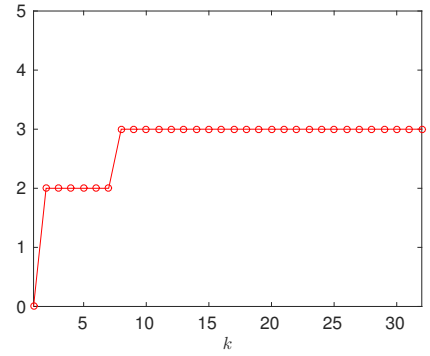
(a) Preconditioned spectrum, $\tau = 10^{-2}$



(b) Adaptive semiseparable order, $\tau = 10^{-2}$



(c) Preconditioned spectrum, $\tau = 10^{-3}$



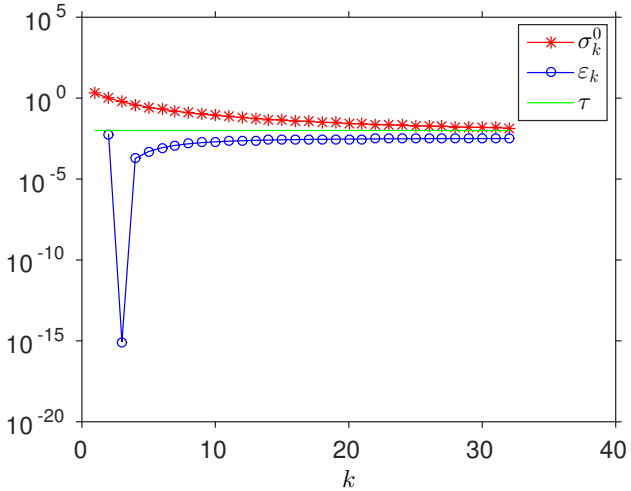
(d) Adaptive semiseparable order, $\tau = 10^{-3}$

Figure 4.17: Preconditioned spectrum and adaptive semiseparable order for $\kappa = 2$

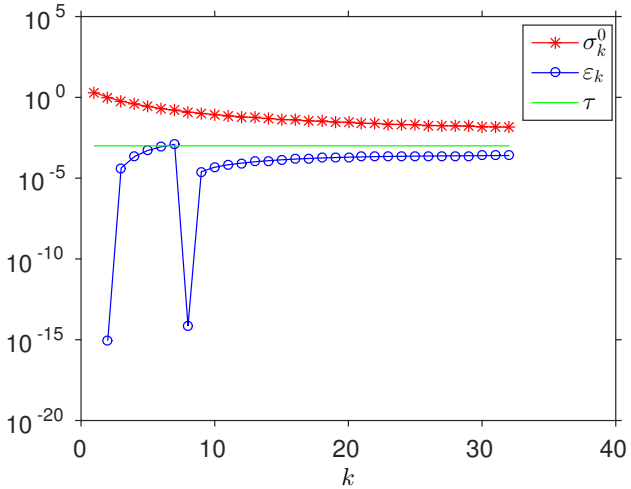
For $\tau = 10^{-2}$, the error introduced by the model order reduction is already smaller than σ_0 as illustrated by Figure 4.18(a), we deduce that the preconditioned spectrum is contained in a circle that is small enough according to Proposition 4.8. This is well illustrated by Figure 4.17(a). If we reduce τ to 10^{-3}, then we get

even smaller circle that contains the preconditioned spectrum in Figure 4.17(c). Figure 4.17(a) and Figure 4.17(c) indicate that by reducing τ with a factor of 10, the radius of the circle that contains the preconditioned spectrum also decreases by a factor around 10. This again verifies our analysis on the radius of the circle in Proposition 4.8.

Both settings of τ give small enough circle to yield very fast convergence for Krylov solvers. `IDR(4)` computes the solution in 4 iterations for $\tau = 10^{-2}$ and 2 iterations for $\tau = 10^{-3}$. For both cases, the semiseparable order is small enough to yield linear computational complexity of the MSSS preconditioners.



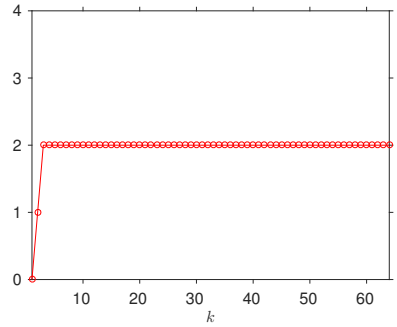
(a) $\tau = 10^{-2}$



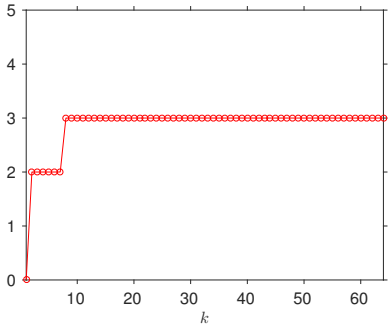
(b) $\tau = 10^{-3}$

Figure 4.18: Minimal singular value and ε_k for $\kappa = 2$

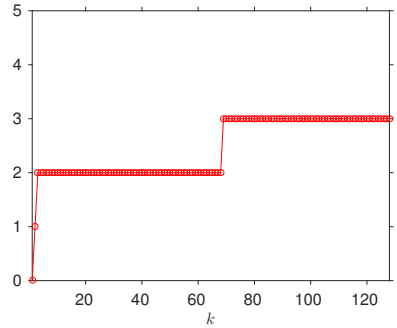
For the settings of different τ and h, the computational results are reported in Table 4.3. The adaptive semiseparable order are plotted in Figure 4.19.



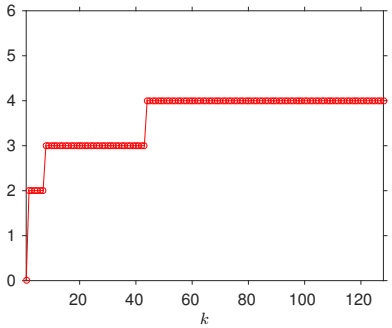
(a) $h = 2^{-6}$, $\tau = 10^{-2}$, $\kappa = 4$



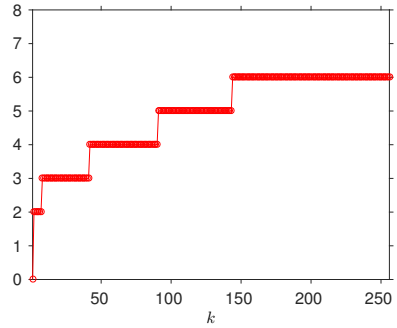
(b) $h = 2^{-6}$, $\tau = 10^{-3}$, $\kappa = 4$



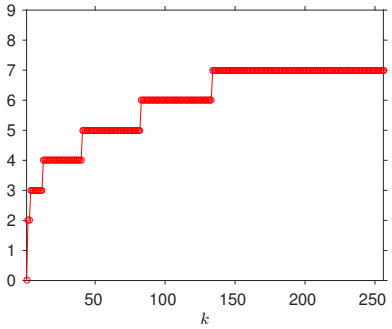
(c) $h = 2^{-7}$, $\tau = 10^{-2}$, $\kappa = 8$



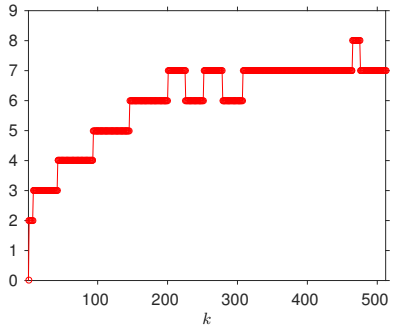
(d) $h = 2^{-7}$, $\tau = 10^{-3}$, $\kappa = 8$



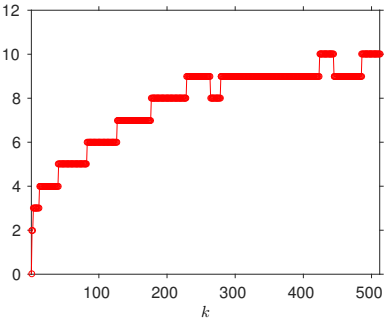
(e) $h = 2^{-8}$, $\tau = 10^{-3}$, $\kappa = 16$



(f) $h = 2^{-8}$, $\tau = 10^{-4}$, $\kappa = 16$



(g) $h = 2^{-9}$, $\tau = 10^{-3}$, $\kappa = 32$



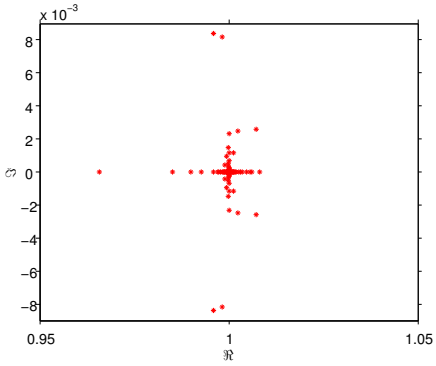
(h) $h = 2^{-9}$, $\tau = 10^{-4}$, $\kappa = 32$

Figure 4.19: Adaptive semiseparable order for the Helmholtz equation with $\kappa h = {}^{1}/_{16}$

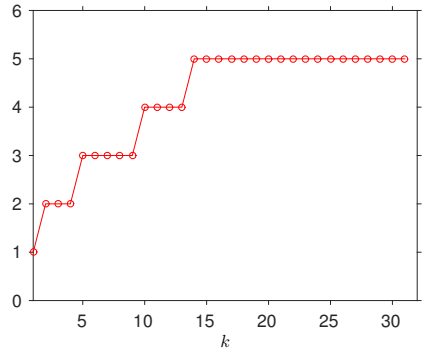
The computational results in Table 4.3 show that the number of iterations can be kept constant by setting a proper τ. We keep κh constant for numerical experiments. This makes the Helmholtz equation even more difficult to solve, and results in a slight increase of the semiseparable order for big κ in Figure 4.19. However, the semiseparable order is still bounded by a small number.

Table 4.3: Performance of MSSS preconditioner for the Helmholtz equation with $\kappa h = {}^1/_{16}$

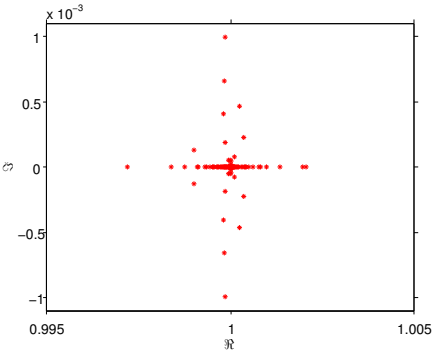
h	κ	N^2	τ	# iter.
2^{-5}	2	$1.09e+03$	10^{-2}	4
			10^{-3}	2
2^{-6}	4	$4.23e+03$	10^{-2}	6
			10^{-3}	3
2^{-7}	8	$1.66e+04$	10^{-2}	8
			10^{-3}	4
2^{-8}	16	$6.61e+04$	10^{-3}	7
			10^{-4}	3
2^{-9}	32	$2.63e+05$	10^{-3}	14
			10^{-4}	3



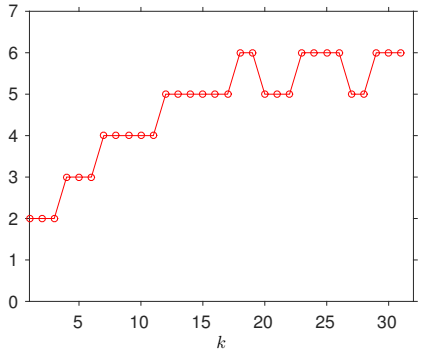
(a) Preconditioned spectrum, $\tau = 10^{-2}$



(b) Adaptive semiseparable order, $\tau = 10^{-2}$



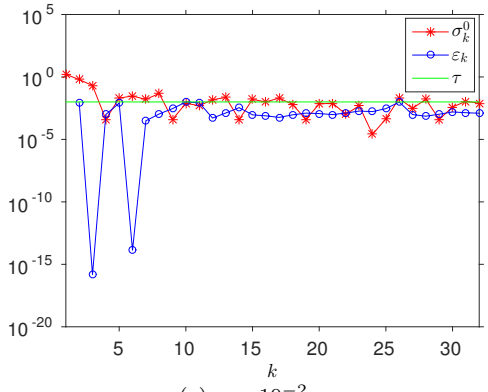
(c) Preconditioned spectrum, $\tau = 10^{-3}$



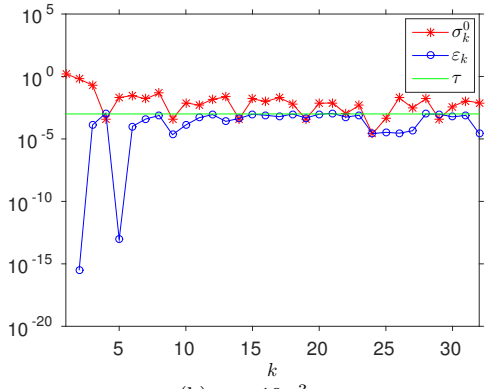
(d) Adaptive semiseparable order, $\tau = 10^{-3}$

Figure 4.20: Preconditioned spectrum and adaptive semiseparable order for $\kappa = 20$

Next, we set $\kappa h = 0.625$ that almost reaches the limit of condition (4.12). We first report the preconditioned spectrum to demonstrate our analysis. By choosing different settings of τ, the computational results for the mesh size $h = 2^{-5}$ are given in Figure 4.20 - 4.21.



(a) $\tau = 10^{-2}$



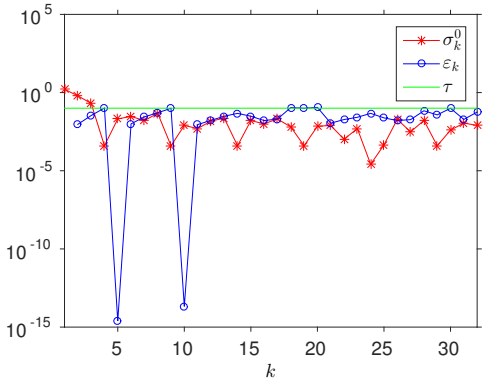
(b) $\tau = 10^{-3}$

Figure 4.21: Minimal singular value and ε for $\kappa = 20$

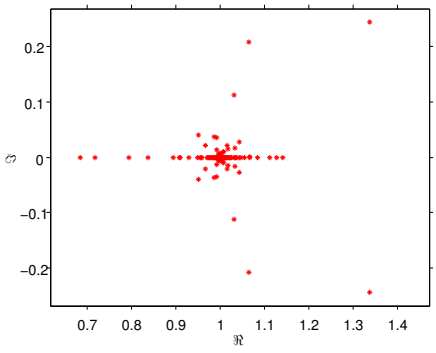
τ is first chosen as 10^{-2}, which is of $\mathcal{O}(\sigma_0)$. This gives the computational results in Figure 4.20. The preconditioned spectrum in Figure 4.20(a) is contained in a circle with very small radius. By decreasing τ, we get a even smaller radius of the circle in Figure 4.20(c). It is shown in Figure 4.20(a) and Figure 4.20(c) that if τ is decreased by a factor of 10, the radius of the circle that contains the preconditioned spectrum is also reduced by a factor around 10. Both settings give super fast convergence. Here the Helmholtz problem is solved by `IDR(4)` in only 3 iterations for $\tau = 10^{-2}$ and 2 iterations for $\tau = 10^{-3}$.

We can even relax the settings of τ and still compute an efficient MSSS preconditioner. This setting gives us smaller semiseparable order that is preferable. The computational results are shown in Figure 4.22. The preconditioned spectrum is contained in a circle with a bigger radius compared with the case $\tau = 10^{-2}$. Compare Figure 4.22(b) with Figure 4.20(a), we see that by increasing τ with a factor of 10, the radius of the circle that contains the preconditioned spectrum also increases

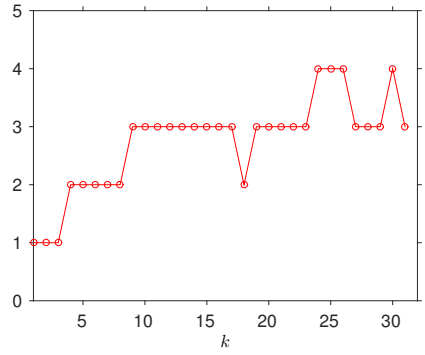
by a factor around 10. This is stated in Proposition 4.8. The circle that contains the preconditioned spectrum still has a small radius for $\tau = 10^{-1}$. Therefore, `IDR(4)` computes the solution in only 9 iterations.



(a) Minimal singular value and ε_k, $\kappa = 20$



(b) Preconditioned spectrum, $\kappa = 20$



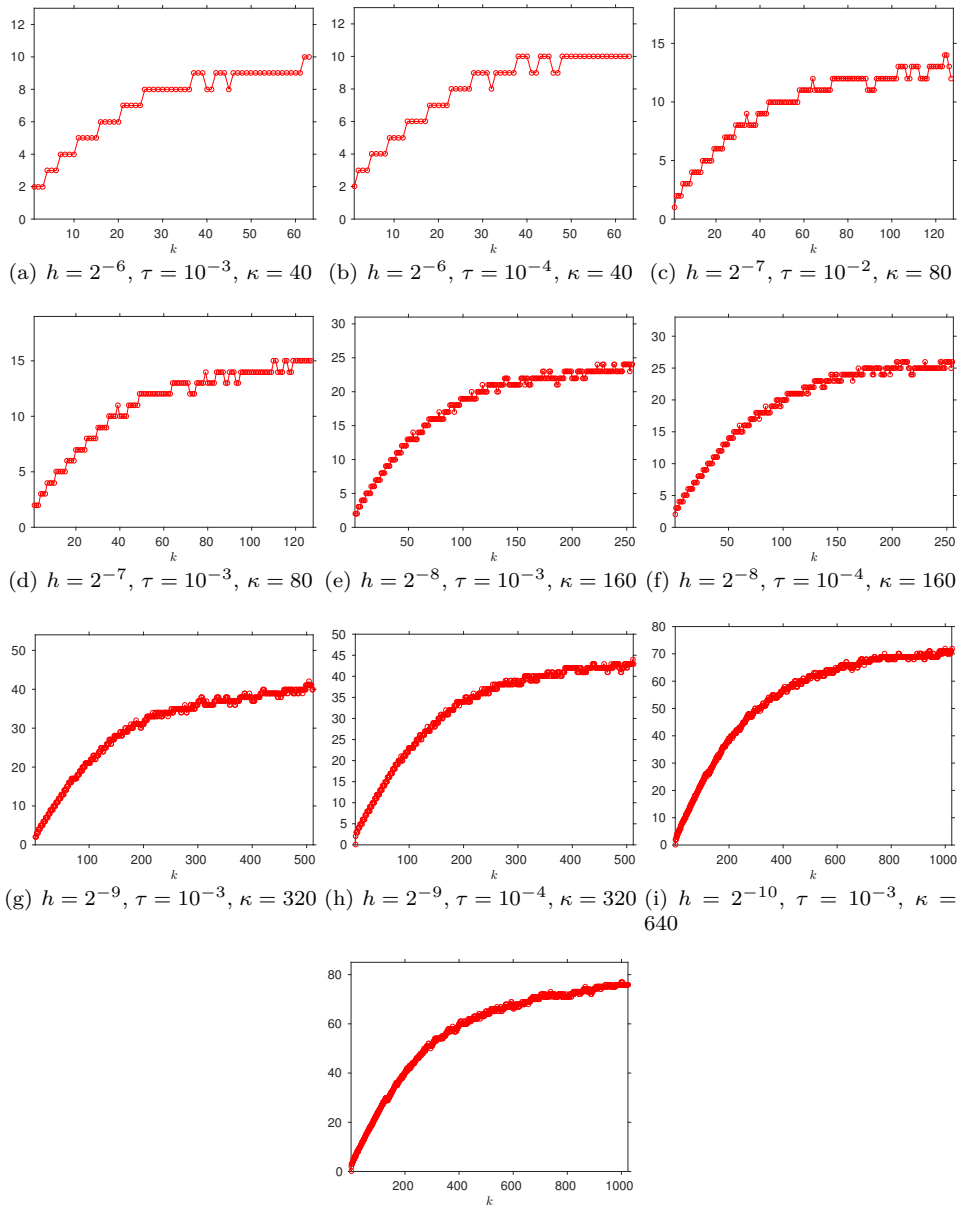
(c) Adaptive semiseparable order, $\kappa = 20$

Figure 4.22: Preconditioned spectrum and adaptive semiseparable order for $\tau = 10^{-1}$

Table 4.4: Performance of MSSS preconditioner for the Helmholtz equation with $\kappa h = 0.625$

h	κ	N^2	τ	# iter.
2^{-5}	20	$1.09e+03$	10^{-2}	3
			10^{-3}	2
2^{-6}	40	$4.23e+03$	10^{-3}	3
			10^{-4}	3
2^{-7}	80	$1.66e+04$	10^{-2}	8
			10^{-3}	4
2^{-8}	160	$6.61e+04$	10^{-3}	5
			10^{-4}	3
2^{-9}	320	$2.63e+05$	10^{-3}	5
			10^{-4}	3
2^{-10}	640	$1.05e+06$	10^{-3}	6
			10^{-4}	3

We report the computational results of the MSSS preconditioner for different mesh sizes h and settings of τ in Table 4.4. The adaptive semiseparable order for the MSSS preconditioners are plotted in Figure 4.23.



(a) $h = 2^{-6}$, $\tau = 10^{-3}$, $\kappa = 40$ (b) $h = 2^{-6}$, $\tau = 10^{-4}$, $\kappa = 40$ (c) $h = 2^{-7}$, $\tau = 10^{-2}$, $\kappa = 80$

(d) $h = 2^{-7}$, $\tau = 10^{-3}$, $\kappa = 80$ (e) $h = 2^{-8}$, $\tau = 10^{-3}$, $\kappa = 160$ (f) $h = 2^{-8}$, $\tau = 10^{-4}$, $\kappa = 160$

(g) $h = 2^{-9}$, $\tau = 10^{-3}$, $\kappa = 320$ (h) $h = 2^{-9}$, $\tau = 10^{-4}$, $\kappa = 320$ (i) $h = 2^{-10}$, $\tau = 10^{-3}$, $\kappa = 640$

(j) $h = 2^{-10}$, $\tau = 10^{-4}$, $\kappa = 640$

Figure 4.23: Adaptive semiseparable order for the Helmholtz equation with $\kappa h = 0.625$

Results listed in Table 4.4 illustrate that by properly setting τ, we obtain mesh size h independent and wave number k independent convergence. The number of itera-

tions can be kept virtually constant. For the shifted Laplacian preconditioner [55], which is the state-of-the-art preconditioning technique for the Helmholtz equation, the number of iterations scales linearly with the wave number κ [126]. Some recent effort dedicated to reduce the dependency on wave number of the number of iterations is carried out by making use of the deflation technique, cf. [114].

With the refinement of the mesh, the wave number κ increases linearly. This makes the Helmholtz problem difficult to solve for small mesh size, which is illustrated by the increase of semiseparable order. Figure 4.23 shows that the semiseparable order has a considerable increase with the refinement of the mesh and is not bounded by a small number, but a moderate number of $\mathcal{O}(\sqrt{N})$. As stated in [33] that the computational complexity for the SSS matrix computations is linear with respect to the problem size with a prefactor r_k^3 provided that r_k is small. Next, we analyze the computational complexity of SSS matrix computations for big r_k but $r_k \ll N$, which corresponds to the case of the Helmholtz equation for $\kappa h = 0.625$.

For an SSS matrix A of $N \times N$ with semiseparable order r_k, the size of its diagonal blocks is denoted by n, then A has N/n blocks. The computational complexity for the matrix-matrix operations and the model order reduction of SSS matrices is bounded by

$$\mathcal{O}(\max\left\{n^3,\ n^2 r_k,\ r_k^2 n,\ r_k^3\right\}\frac{N}{n}), \tag{4.13}$$

which can be obtained by checking the SSS matrix computations in [33]. For small r_k, we also set n small enough. This gives the computational complexity of $\mathcal{O}(r_k^3 N)$, i.e., linear with respect to the problem size. For moderate r_k, the term r_k^3 becomes big. According to (4.13), the settings of n can be adjusted such that a proper computational complexity can be reached. Usually, we choose n and r_k of the same order, say $r_k = n$. This in turn gives the computational complexity for SSS matrix computations of $\mathcal{O}(r_k^2 N)$ for moderate r_k. Note that the setting of n does not change r_k, since r_k is the rank of the off-diagonal blocks, which only depends on the property of the matrix.

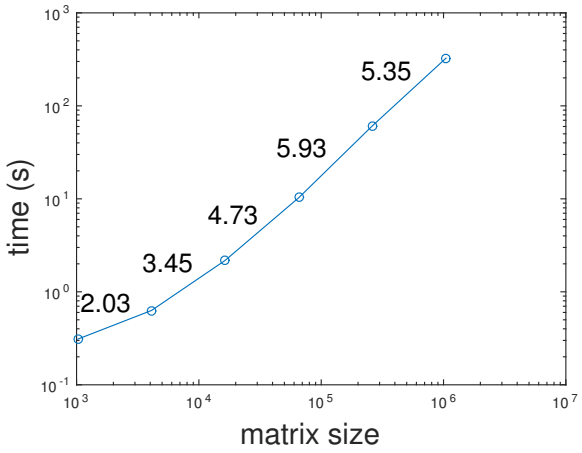


Figure 4.24: MSSS preconditioning time for $\kappa h = 0.625$

For the computations of the MSSS preconditioners with moderate r_k, N Schur complements are computed while each Schur complement is computed in $\mathcal{O}(r_k^2 N)$. This in turn gives the total computational complexity $\mathcal{O}(r_k^2 N^2)$, where N^2 is the problem size. For the case $\kappa h = 0.625$, r_k is roughly bounded by $\mathcal{O}(\sqrt{N})$, this in turn results the total computational complexity bounded by $\mathcal{O}(N^2)^{\frac{3}{2}}$. This is comparable with the computational complexity of a multi-frontal solver for 2D problems [54]. We use Figure 4.24 to show the growth factor of time to compute the MSSS preconditioner with the mesh refinement.

Here we set $n = 4,\ 8,\ 8,\ 16,\ 32,\ 128$ with the refinement of the mesh and $\tau = 10^{-3}$ for all the mesh sizes except $\tau = 10^{-4}$ for $h = 2^{-9}$. All the time are measured in seconds. The number over the line shows the growth factor of the time to compute the MSSS preconditioner. It is clear that the growth factor for time is below $4^{\frac{3}{2}} = 8$. Note that we use a non-equidistant axis for Figure 4.24.

4.6.3 Saddle-Point Systems

We study the convergence property of MSSS preconditioners for the saddle-point systems in this part. Consider the following PDE-constrained optimization problem given by Example 4.4.

Example 4.4 ([102]) *Let* $\Omega = [0,\ 1]^2$ *and consider the problem*

$$\begin{aligned} \min_{u,f}\ & \frac{1}{2}\|u - \hat{u}\| + \frac{\beta}{2}\|f\|^2 \\ s.t.\ & -\nabla^2 u = f \ \text{in}\ \Omega \\ & u = u_D \ \text{on}\ \Gamma_D, \end{aligned}$$

where $\Gamma_D = \partial\Omega$, $\beta > 0$, $\hat{u} = 0$ *is the prescribed system state, and*

$$u_D = \begin{cases} -\sin(2\pi y) & \text{if}\ \ x = 1, 0 \le y \le 1, \\ \sin(2\pi y) & \text{if}\ \ x = 0, 0 \le y \le 1, \\ 0 & \text{otherwise.} \end{cases}$$

Discretize the cost function and the PDE constraints by using the Galerkin method and then compute the optimality condition gives the following linear saddle-point system to solve

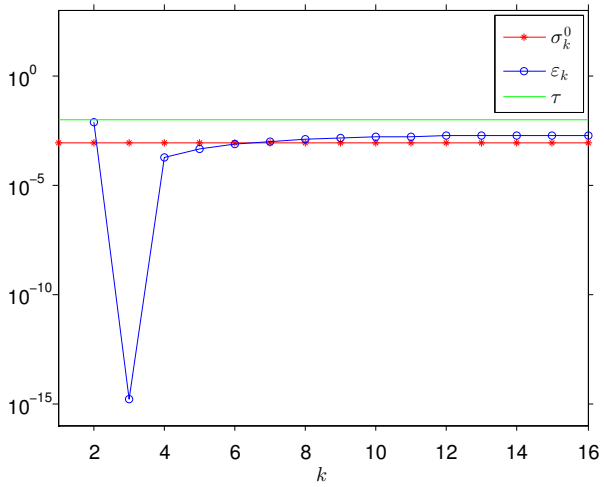
$$\begin{bmatrix} 2\beta M & 0 & -M \\ 0 & M & K^T \\ -M & K & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ b \\ d \end{bmatrix}. \tag{4.14}$$

Here M is the mass matrix, K is the stiffness matrix, x and y are the discrete analog of f and u, λ is the Lagrangian multiplier, b and d are obtained by discretizing the cost function and boundary conditions, respectively.

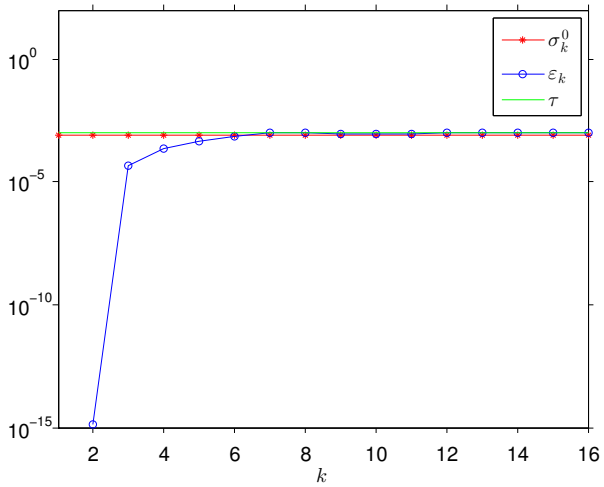
All the sub-blocks of the saddle-point system (4.14) have an MSSS structure and can be exploited to get a global MSSS structure. By exploiting the global MSSS

structure of the saddle-point system, we can compute a global MSSS preconditioner. This is discussed in great detail in [103]. Here, we use the numerical experiments of preconditioning the saddle-point system (4.14) to demonstrate the convergence analysis in Section 4.3.

We report the computational results for the mesh size $h = 2^{-4}$ with a wide range settings of β and τ. First, we test a moderate set of $\tau = 10^{-2}$ and $\tau = 10^{-3}$ for $\beta = 10^{-1}$. The computational results are given in Figure 4.25 and Figure 4.26.



(a) $\tau = 10^{-2}$

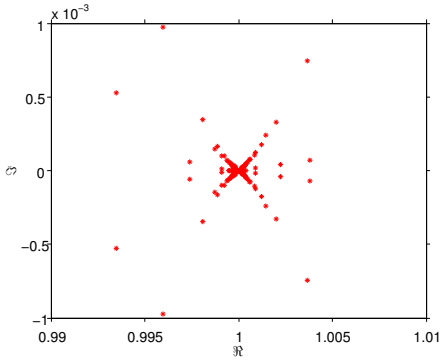


(b) $\tau = 10^{-3}$

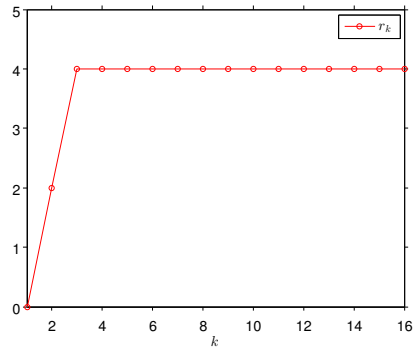
Figure 4.25: Minimal singular value and approximation error for $h = 2^{-4}$, $\beta = 10^{-1}$

Figure 4.26(a) and Figure 4.26(c) show that a moderate setting of τ gives a small radius of the circle that contains the preconditioned spectrum, and the smaller τ is, the smaller the radius is. Both settings give adequately small circle and the decrease of τ just yields a slightly increase of the semiseparable order, which is

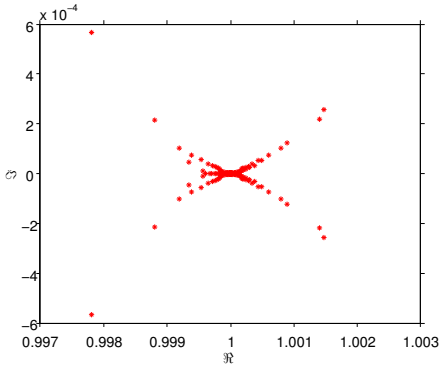
shown in Figure 4.26(b) and Figure 4.26(d). The `IDR(4)` solves the preconditioned system for $\tau = 10^{-2}$ in only 3 iterations and only 2 iterations for $\tau = 10^{-3}$.



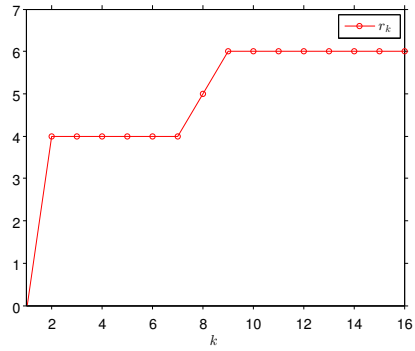
(a) Preconditioned spectrum, $\tau = 10^{-2}$



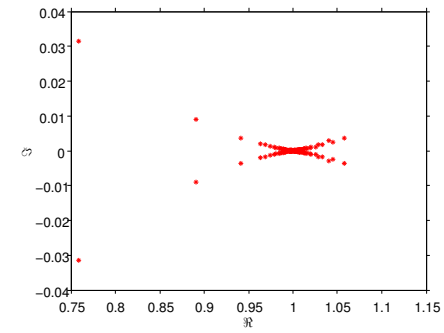
(b) Adaptive semiseparable order, $\tau = 10^{-2}$



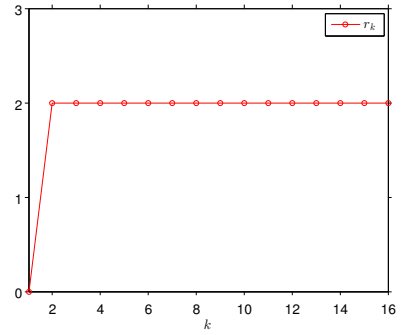
(c) Preconditioned spectrum, $\tau = 10^{-3}$



(d) Adaptive semiseparable order, $\tau = 10^{-3}$



(e) Preconditioned spectrum, $\tau = 10^{-1}$



(f) Adaptive semiseparable order, $\tau = 10^{-1}$

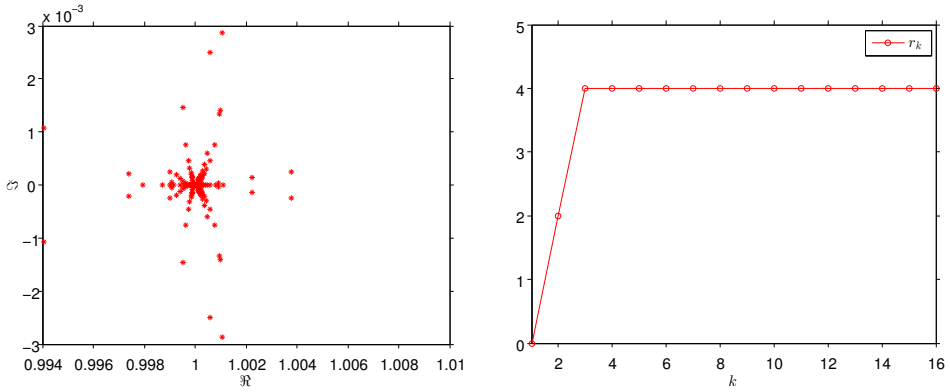
Figure 4.26: Preconditioned spectrum and adaptive semiseparable order for $h = 2^{-4}$, $\beta = 10^{-1}$

We can even set a bigger τ to compute an MSSS preconditioner, the computational results for $\tau = 10^{-1}$ are plotted in Figure 4.26(e) - 4.26(f). Figure 4.26(e) illustrates that a bigger τ gives a bigger radius of the circle that contains the preconditioned

spectrum. But the circle is still small and the IDR(4) solver computes the solution of the preconditioned system in only 5 iterations. Moreover, this settings of τ gives a smaller semiseparable order, which is shown in Figure 4.26(f).

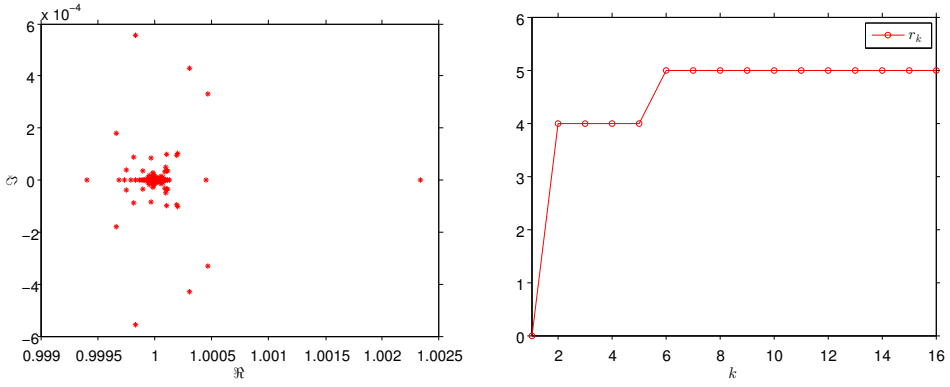
The smallest singular value of the saddle-point system (4.14) scales with β. A smaller β in turn gives a smaller smallest singular value. This makes the saddle-point system (4.14) even more ill-conditioned and difficult to solve. Next, we test the case for a moderate $\beta = 10^{-2}$, and a much smaller $\beta = 10^{-5}$, the computational results for $\tau = 10^{-2}$, and $\tau = 10^{-3}$ are reported in Figure 4.27 - 4.30.

The computational results in Figure 4.27 - 4.30 show that a moderate settings of τ gives satisfactory clustering of the preconditioned spectrum. The IDR(4) solver computes the solution in 2 or 3 iterations for all the settings of τ and β that corresponds to the test cases in Figure 4.27 - 4.30.



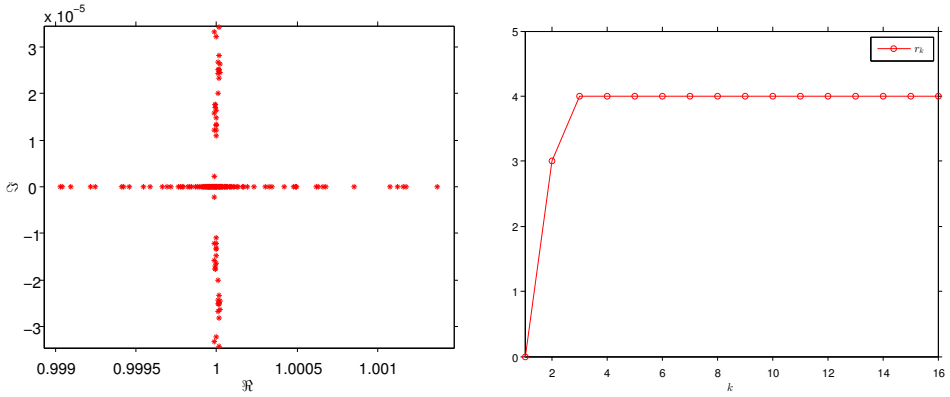
(a) Preconditioned spectrum (b) Adaptive semiseparable order

Figure 4.27: Preconditioned spectrum and adaptive semiseparable order for $h = 2^{-4}$, $\tau = 10^{-2}$, $\beta = 10^{-2}$



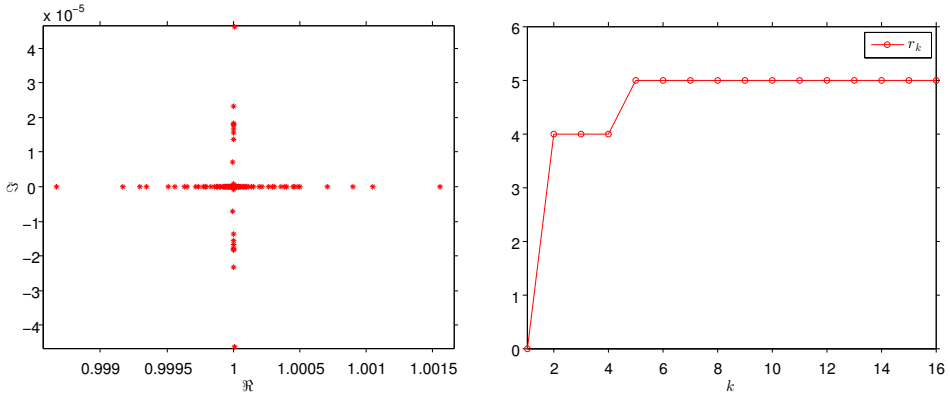
(a) Preconditioned spectrum (b) Adaptive semiseparable order

Figure 4.28: Preconditioned spectrum and adaptive semiseparable order for $h = 2^{-4}$, $\tau = 10^{-3}$, $\beta = 10^{-2}$



(a) Preconditioned spectrum (b) Adaptive semiseparable order

Figure 4.29: Preconditioned spectrum and adaptive semiseparable order for $h = 2^{-4}$, $\tau = 10^{-2}$, $\beta = 10^{-5}$



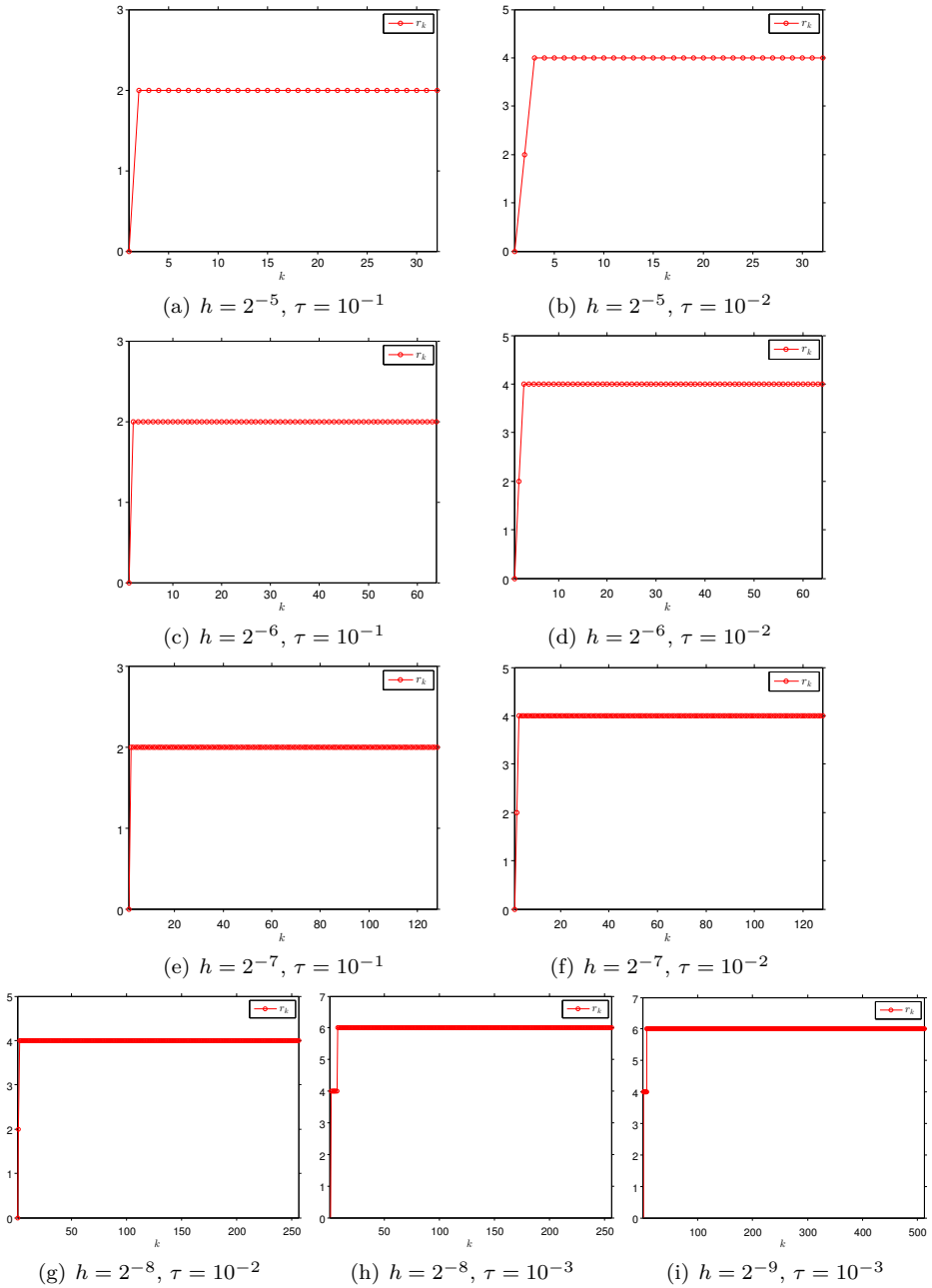
(a) Preconditioned spectrum (b) Adaptive semiseparable order

Figure 4.30: Preconditioned spectrum and adaptive semiseparable order for $h = 2^{-4}$, $\tau = 10^{-3}$, $\beta = 10^{-5}$

Table 4.5: Performance of MSSS preconditioner for the PDE-constrained optimization problem with different β

h	N^2	$\beta = 10^{-1}$		$\beta = 10^{-2}$		$\beta = 10^{-5}$	
		τ	# iter.	τ	# iter.	τ	# iter.
2^{-5}	$3.07e+03$	10^{-1}	6	10^{-1}	6	10^{-1}	4
		10^{-2}	3	10^{-2}	4	10^{-2}	3
2^{-6}	$1.23e+04$	10^{-1}	9	10^{-1}	8	10^{-2}	16
		10^{-2}	4	10^{-2}	4	10^{-3}	3
2^{-7}	$4.92e+04$	10^{-1}	13	10^{-1}	16	10^{-3}	6
		10^{-2}	5	10^{-2}	5	10^{-4}	2
2^{-8}	$1.97e+05$	10^{-2}	10	10^{-2}	10	10^{-4}	3
		10^{-3}	4	10^{-3}	4		
2^{-9}	$7.86e+05$	10^{-3}	6	10^{-3}	6	10^{-4}	19
						10^{-5}	2

The performance of the MSSS preconditioner for different settings of τ, β, and the mesh size h are given in Table 4.5. The corresponding semiseparable order are plotted in Figure 4.31 - 4.33.



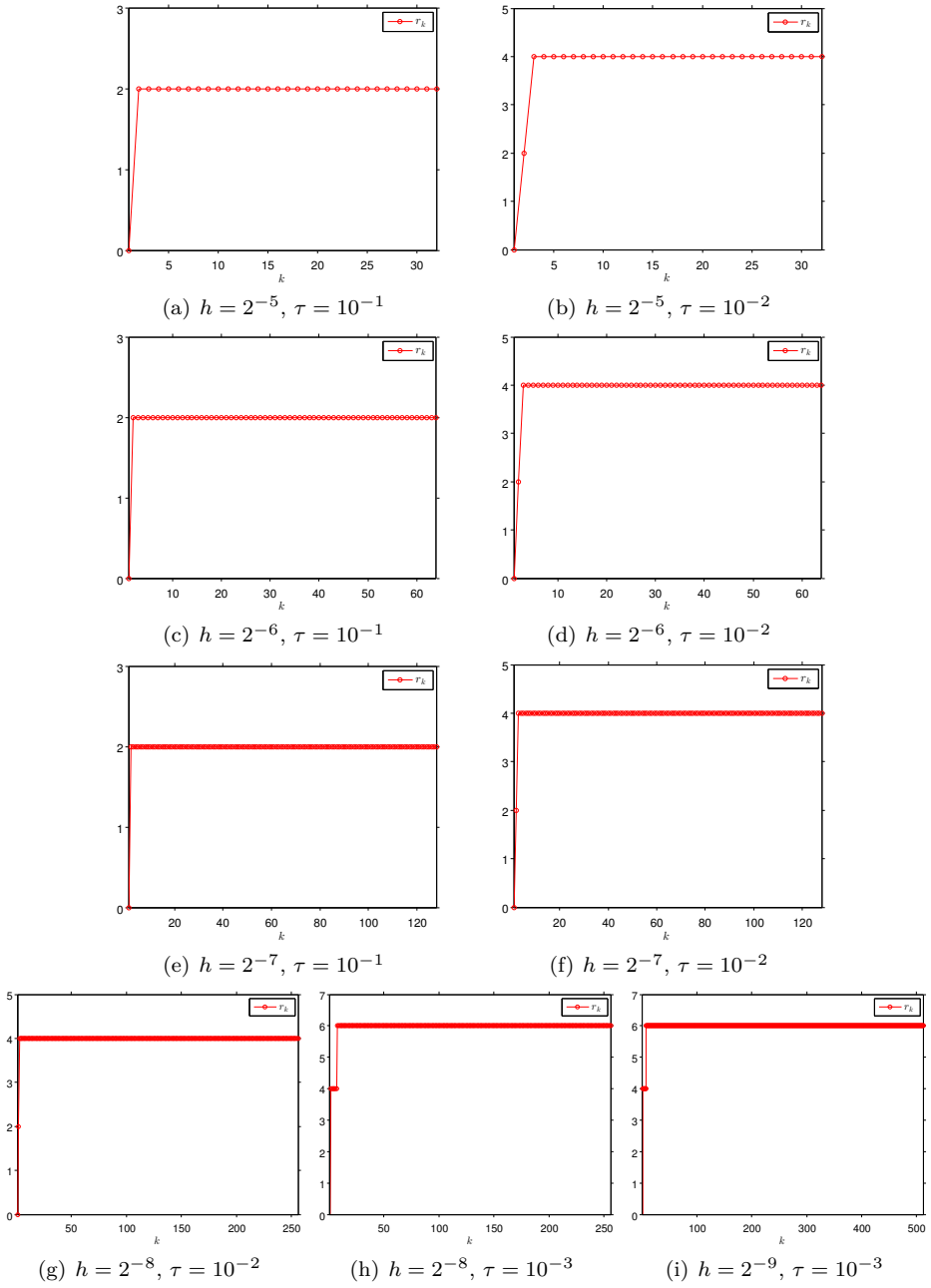
(a) $h = 2^{-5}$, $\tau = 10^{-1}$ (b) $h = 2^{-5}$, $\tau = 10^{-2}$

(c) $h = 2^{-6}$, $\tau = 10^{-1}$ (d) $h = 2^{-6}$, $\tau = 10^{-2}$

(e) $h = 2^{-7}$, $\tau = 10^{-1}$ (f) $h = 2^{-7}$, $\tau = 10^{-2}$

(g) $h = 2^{-8}$, $\tau = 10^{-2}$ (h) $h = 2^{-8}$, $\tau = 10^{-3}$ (i) $h = 2^{-9}$, $\tau = 10^{-3}$

Figure 4.31: Adaptive semiseparable order for $\beta = 10^{-1}$



(a) $h = 2^{-5}$, $\tau = 10^{-1}$ (b) $h = 2^{-5}$, $\tau = 10^{-2}$

(c) $h = 2^{-6}$, $\tau = 10^{-1}$ (d) $h = 2^{-6}$, $\tau = 10^{-2}$

(e) $h = 2^{-7}$, $\tau = 10^{-1}$ (f) $h = 2^{-7}$, $\tau = 10^{-2}$

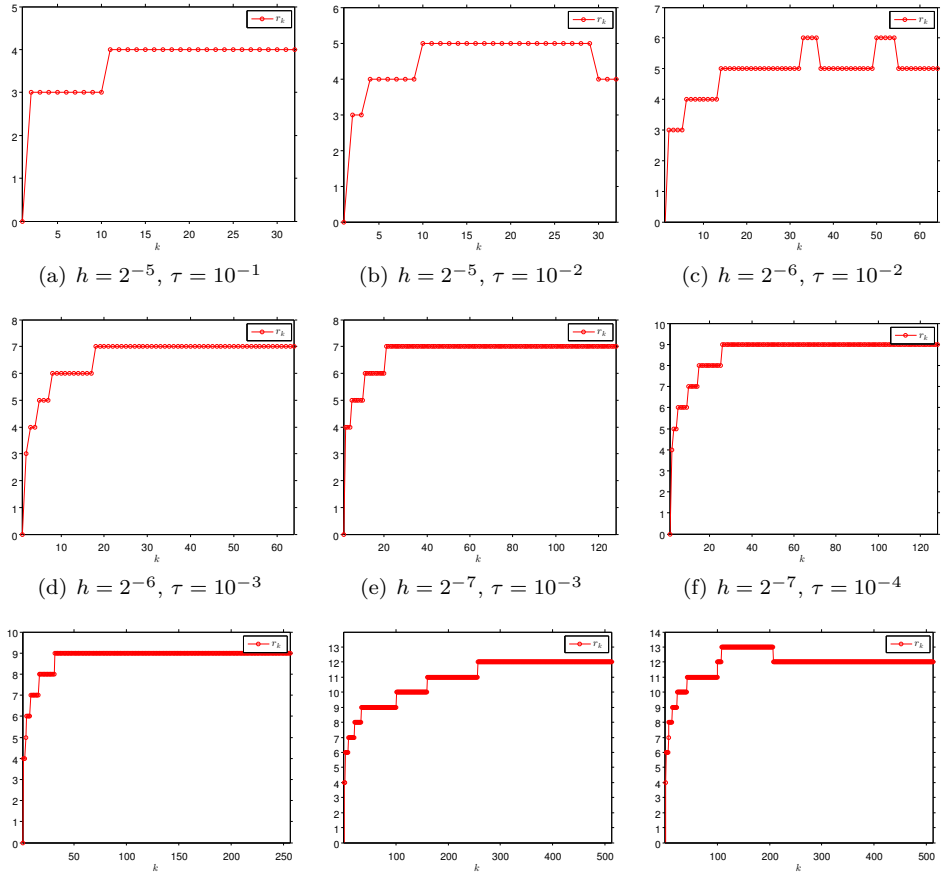
(g) $h = 2^{-8}$, $\tau = 10^{-2}$ (h) $h = 2^{-8}$, $\tau = 10^{-3}$ (i) $h = 2^{-9}$, $\tau = 10^{-3}$

Figure 4.32: Adaptive semiseparable order for $\beta = 10^{-2}$

The computational results in Figure 4.31 - 4.32 for the test case with $\beta = 10^{-1}$ and $\beta = 10^{-2}$ show that for a constant setting of τ, the semiseparable order is bounded by a constant 4 for the mesh size h ranges from 2^{-7} to 2^{-5}. The semiseparable order is independent of the mesh size h and β. Since the smallest singular value for all

the principle leading sub-matrices of the saddle-point systems also scales with mesh size h, for a bigger test example with mesh size $h = 2^{-9}$ and $h = 2^{-8}$, a smaller τ is needed to get a satisfactory radius of the circle that contains the preconditioned spectrum according to Proposition 4.8. This is verified by the computational results in Figure 4.31 - 4.33 and Table 4.5. The setting of a smaller τ yields a slightly increase of the semiseparable order from 4 to 6, which is still quite small.

For much smaller $\beta = 10^{-5}$, the saddle-point system is even more ill-conditioned. The smallest singular value for all the principle leading sub-matrices is even smaller. To solve such an ill-conditioned system, a smaller τ is necessary to get a satisfactory radius of the circle that contains the preconditioned spectrum, compared with the case for moderate β. This yields a slightly increase of the semiseparable order and the semiseparable orders for all the test cases are still bounded by a small constant, which is illustrated by Figure 4.33.



(a) $h = 2^{-5}$, $\tau = 10^{-1}$ (b) $h = 2^{-5}$, $\tau = 10^{-2}$ (c) $h = 2^{-6}$, $\tau = 10^{-2}$

(d) $h = 2^{-6}$, $\tau = 10^{-3}$ (e) $h = 2^{-7}$, $\tau = 10^{-3}$ (f) $h = 2^{-7}$, $\tau = 10^{-4}$

(g) $h = 2^{-8}$, $\tau = 10^{-4}$ (h) $h = 2^{-9}$, $\tau = 10^{-4}$ (i) $h = 2^{-9}$, $\tau = 10^{-5}$

Figure 4.33: Adaptive semiseparable order for $\beta = 10^{-5}$

Remark 4.26 According to the computational results for different regularization parameter β and mesh size h in Figure 4.31 - 4.33 and Table 4.5, we show that by a

proper setting of the parameter τ for the MSSS preconditioner, we can compute an efficient preconditioner that gives mesh size and regularization parameter independent convergence. The computational complexity for the MSSS preconditioning technique can be kept linear with the problem sizes.

4.7 Conclusions

In this chapter, we made a convergence analysis of the multilevel sequentially semiseparable (MSSS) preconditioners for a wide class of linear systems. This includes unsymmetric systems, symmetric indefinite systems from discretization of scalar PDEs and saddle-point systems. We showed that the spectrum of the preconditioned system is contained in a circle centered at $(1, 0)$ and we gave an analytic bound for the radius. Our analysis shows that the radius of the circle can be made arbitrarily small by properly setting a parameter in the MSSS preconditioner. We also demonstrated how to select the parameter. We validate our analysis by performing numerical experiments.

4.8 Appendix: Proof of Lemma 4.16

Before the proof, we give the following lemmas and corollaries that are necessary.

Lemma 4.27 ([120]) *Let $A \in \mathbb{C}^{m\times n}$ be partitioned in the form*

$$A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}.$$

Let the singular values of A be $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$ and those of A_1 be $\tau_1 \geq \tau_2 \geq \cdots \geq \tau_n$. Then

$$\sigma_i \geq \tau_i, \quad i = 1,\ 2,\ \ldots,\ n.$$

From Lemma 4.27, we can also get the inequality between singular values of A and A_2, which is stated by Proposition 4.28.

Proposition 4.28 *Let the singular values of A_2 in Lemma 4.27 be $\nu_1 \geq \nu_2 \geq \cdots \geq \nu_n$. Then*

$$\sigma_i \geq \nu_i, \quad i = 1,\ 2,\ \ldots,\ n.$$

Proof: It is easy to obtain

$$\begin{bmatrix} A_2 \\ A_1 \end{bmatrix} = \begin{bmatrix} 0 & I_p \\ I_q & 0 \end{bmatrix} \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} = \begin{bmatrix} 0 & I_p \\ I_q & 0 \end{bmatrix} A,$$

where I_p and I_q are identity matrices with proper sizes. Let $\bar{A} = \begin{bmatrix} 0 & I_p \\ I_q & 0 \end{bmatrix} A$, then according to Lemma 4.27, we have

$$\sigma_i \geq \nu_i, \quad i = 1,\ 2,\ \ldots,\ n.$$

This is because that $\bar{A}$ and A have the same singular values. □

According to Lemma 4.27 and Proposition 4.28, we have the following corollary.

Corollary 4.29 *If all the factors $\mathcal{C}_i$ are transformed to the form with orthonormal rows, then we have*

$$\|R_i\|_2 \leq 1, \text{ and } \|Q_i\|_2 \leq 1. \tag{4.15}$$

Proof. According to the procedure to transform $\mathcal{C}_i$ to the form with orthonormal rows introduced in Section 4.5, at step $i+1$, we perform an SVD that gives

$$\begin{bmatrix} R_i & Q_i \end{bmatrix} = U_i \Sigma_i V_i^T,$$

and let $\begin{bmatrix} R_i & Q_i \end{bmatrix} = V_i^T$. This gives

$$V_i = \begin{bmatrix} R_i^T \\ Q_i^T \end{bmatrix}.$$

According to Lemma 4.27 and Proposition 4.28, we have

$$\sigma_k(V_i) \geq \sigma_k(R_i^T), \quad \sigma_k(V_i) \geq \sigma_k(Q_i^T).$$

Since $\sigma_k(R_i) = \sigma_k(R_i^T)$, $\sigma_k(Q_i) = \sigma_k(Q_i^T)$ and $\sigma_k(V_i) = 1$, we have

$$\|R_i\|_2 \leq 1, \quad \|Q_i\|_2 \leq 1. \qquad \square$$

With these lemmas and corollaries, we now give the proof of Lemma 4.16 in the following part.

Proof: Since these inequalities in the lemma start from different steps, we first give the proof at step N and then start the proof by induction from step $N-1$.

For step N, perform an SVD on $\mathcal{O}_N$ gives

$$\mathcal{O}_N = P_N = \begin{bmatrix} U_N & \Delta U_N \end{bmatrix} \begin{bmatrix} \Sigma_N & \\ & \Delta\Sigma_N \end{bmatrix} \begin{bmatrix} V_N^T \\ \Delta V_N^T \end{bmatrix},$$

where Σ_N and $\Delta\Sigma_N$ are diagonal matrices with diagonal entries $\sigma_1, \sigma_2, \cdots, \sigma_{\tilde{r}_N}$, and $\sigma_{\tilde{r}_{N+1}}, \cdots, \sigma_{r_N}$ with

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\tilde{r}_N} > \tau \geq \sigma_{\tilde{r}_{N+1}} \geq \cdots \geq \sigma_{r_N}.$$

Let $\tilde{\mathcal{O}}_N = U_N$ and $\tilde{\mathcal{C}}_N = \Sigma_N V_N^T \mathcal{C}_N$, we have

$$\begin{aligned} \left\| \mathcal{O}_N \mathcal{C}_N - \tilde{O}_N \tilde{\mathcal{C}}_N \right\|_2 &= \left\| \Delta U_N \Delta\Sigma_N \Delta V_N^T \mathcal{C}_N \right\|_2 \\ &= \left\| \Delta U_N \Delta\Sigma_N \Delta V_N^T \right\|_2 \quad (\mathcal{C}_N \text{ has orthonormal rows}) \\ &\leq \tau. \end{aligned}$$

This is exactly the inequality in (4.7) for $i = N$, i.e.,

$$\left\|\tilde{H}_N - \mathcal{H}_N\right\|_2 \leq \tau.$$

After this step, the factor $\tilde{\mathcal{O}}_N$ has orthonormal columns.

According to $\mathcal{C}_N = \begin{bmatrix} R_{N-1}\mathcal{C}_{N-1} & Q_{N-1} \end{bmatrix}$ and $\tilde{\mathcal{C}}_N = \Sigma_N V_N^T \mathcal{C}_N$, we have

$$\tilde{R}_{N-1}^1 = \Sigma_N V_N^T R_{N-1}, \quad \tilde{Q}_{N-1} = \Sigma_N V_N^T Q_{N-1}.$$

Then

$$\begin{aligned}
\left\|\tilde{\mathcal{O}}_N \tilde{Q}_{N-1} - \mathcal{O}_N Q_{N-1}\right\|_2 &= \left\|\Delta U_N \Delta\Sigma_N \Delta V_N^T Q_{N-1}\right\|_2 \\
&\leq \left\|\Delta U_N \Delta\Sigma_N \Delta V_N^T\right\|_2 \|Q_{N-1}\|_2 \\
&\leq \left\|\Delta U_N \Delta\Sigma_N \Delta V_N^T\right\|_2 \quad \text{(Corollary 4.29)} \\
&\leq \tau,
\end{aligned}$$

which gives the inequality (4.9) for $i = N$.

Now, we start our proof from the step $i = N - 1$ by induction. Because of the approximation of $\tilde{\mathcal{O}}_N$, we have

$$\tilde{\mathcal{O}}_{N-1}^1 = \begin{bmatrix} P_{N-1} \\ \tilde{\mathcal{O}}_N \tilde{R}_{N-1}^1 \end{bmatrix}$$

and we have the following inequality hold

$$\left\|\mathcal{O}_{N-1} - \tilde{\mathcal{O}}_{N-1}^1\right\|_2 \leq \tau. \tag{4.16}$$

This is because

$$\begin{aligned}
\left\|\mathcal{O}_{N-1} - \tilde{\mathcal{O}}_{N-1}^1\right\|_2 &= \left\| \begin{bmatrix} P_{N-1} \\ \mathcal{O}_N R_{N-1} \end{bmatrix} - \begin{bmatrix} P_{N-1} \\ \tilde{\mathcal{O}}_N \tilde{R}_{N-1}^1 \end{bmatrix} \right\|_2 \\
&= \left\|\mathcal{O}_N R_{N-1} - \tilde{\mathcal{O}}_N \tilde{R}_{N-1}^1\right\|_2 \\
&= \left\|\Delta U_N \Delta\Sigma_N \Delta V_N^T R_{N-1}\right\|_2 \\
&\leq \left\|\Delta U_N \Delta\Sigma_N \Delta V_N^T\right\|_2 \|R_{N-1}\|_2 \\
&\leq \left\|\Delta U_N \Delta\Sigma_N \Delta V_N^T\right\|_2 \quad \text{(Corollary 4.29)} \\
&\leq \tau.
\end{aligned}$$

This proves the inequality (4.8) for $i = N - 1$.

According to

$$\tilde{\mathcal{O}}_{N-1}^1 = \begin{bmatrix} P_{N-1} \\ \tilde{\mathcal{O}}_N \tilde{R}_{N-1}^1 \end{bmatrix} = \begin{bmatrix} I & \\ & \tilde{\mathcal{O}}_N \end{bmatrix} \begin{bmatrix} P_{N-1} \\ \tilde{R}_{N-1}^1 \end{bmatrix},$$

and $\tilde{\mathcal{O}}_N$ has orthonormal columns, we perform an SVD on $\begin{bmatrix} P_{N-1} \\ \tilde{R}^1_{N-1} \end{bmatrix}$ and get

$$\begin{bmatrix} P_{N-1} \\ \tilde{R}^1_{N-1} \end{bmatrix} = \begin{bmatrix} U_{N-1} & \Delta U_{N-1} \end{bmatrix} \begin{bmatrix} \Sigma_{N-1} & \\ & \Delta\Sigma_{N-1} \end{bmatrix} \begin{bmatrix} V^T_{N-1} \\ \Delta V^T_{N-1} \end{bmatrix}.$$

Let $\tilde{\mathcal{O}}^2_{N-1} = \begin{bmatrix} I & \\ & \tilde{\mathcal{O}}_N \end{bmatrix} U_{N-1}$ and $\tilde{\mathcal{C}}_{N-1} = \Sigma_{N-1} V^T_{N-1} \mathcal{C}_{N-2}$, i.e.,

$$\tilde{\mathcal{C}}_{N-1} = \Sigma_{N-1} V^T_{N-1} \begin{bmatrix} R_{N-2}\mathcal{C}_{N-2} & Q_{N-2} \end{bmatrix},$$

which yields $\tilde{R}^1_{N-2} = \Sigma_{N-1} V^T_{N-1} R_{N-2}$ and $\tilde{Q}_{N-2} = \Sigma_{N-1} V^T_{N-1} Q_{N-2}$.

Then, we obtain

$$\begin{aligned} \left\| \tilde{\mathcal{O}}^2_{N-1}\tilde{\mathcal{C}}_{N-1} - \tilde{\mathcal{O}}^1_{N-1}\mathcal{C}_{N-1} \right\|_2 &= \left\| \Delta U_{N-1}\Delta\Sigma_{N-1}\Delta V^T_{N-1}\mathcal{C}_{N-1} \right\|_2 \\ &= \left\| \Delta U_{N-1}\Delta\Sigma_{N-1}\Delta V^T_{N-1} \right\|_2 \\ &\leq \tau \quad (\mathcal{C}_{N-1} \text{ has orthonormal rows}). \end{aligned} \tag{4.17}$$

This in turn gives

$$\begin{aligned} \left\| \mathcal{O}_{N-1}\mathcal{C}_{N-1} - \tilde{\mathcal{O}}^2_{N-1}\tilde{\mathcal{C}}_{N-1} \right\|_2 &\leq \left\| \mathcal{O}_{N-1}\mathcal{C}_{N-1} - \tilde{\mathcal{O}}^1_{N-1}\mathcal{C}_{N-1} \right\|_2 \\ &\quad + \left\| \tilde{\mathcal{O}}^1_{N-1}\mathcal{C}_{N-1} - \tilde{\mathcal{O}}^2_{N-1}\tilde{\mathcal{C}}_{N-1} \right\|_2 \\ &= \left\| \mathcal{O}_{N-1} - \tilde{\mathcal{O}}^1_{N-1} \right\|_2 + \left\| \tilde{\mathcal{O}}^1_{N-1}\mathcal{C}_{N-1} - \tilde{\mathcal{O}}^2_{N-1}\tilde{\mathcal{C}}_{N-1} \right\|_2 \\ &\leq 2\tau \\ &\quad (\mathcal{C}_{N-1} \text{ has orthonormal rows and (4.16) (4.17)}), \end{aligned}$$

i.e.,

$$\left\| \mathcal{H}_{N-1} - \tilde{\mathcal{H}}_{N-1} \right\|_2 \leq 2\tau,$$

which proves inequality (4.7) for $i = N-1$.

Additionally,

$$\begin{aligned} \left\| \tilde{\mathcal{O}}^2_{N-1}\tilde{Q}_{N-2} - \tilde{\mathcal{O}}^1_{N-1}Q_{N-2} \right\|_2 &= \left\| \begin{bmatrix} I & \\ & \tilde{\mathcal{O}}_N \end{bmatrix} \Delta U_{N-1}\Delta\Sigma_{N-1}\Delta V^T_{N-1}Q_{N-2} \right\|_2 \\ &= \left\| \Delta U_{N-1}\Delta\Sigma_{N-1}\Delta V^T_{N-1}Q_{N-2} \right\|_2 \\ &\quad (\tilde{\mathcal{O}}_N \text{ is orthonormal}) \\ &\leq \left\| \Delta U_{N-1}\Delta\Sigma_{N-1}\Delta V^T_{N-1} \right\|_2, \quad \text{(Corollary 4.29)} \\ &\leq \tau. \end{aligned}$$

And,

$$\begin{aligned}\left\|\tilde{\mathcal{O}}^1_{N-1}Q_{N-2}-\mathcal{O}_{N-1}Q_{N-2}\right\|_2 &\le \left\|\tilde{\mathcal{O}}^1_{N-1}-\mathcal{O}_{N-1}\right\|_2\|Q_{N-2}\|_2\\ &\le \left\|\tilde{\mathcal{O}}^1_{N-1}-\mathcal{O}_{N-1}\right\|_2 \quad \text{(Corollary 4.29)}\\ &\le \tau. \quad ((4.16))\end{aligned}$$

This in turn yields

$$\begin{aligned}\left\|\tilde{\mathcal{O}}^2_{N-1}\tilde{Q}_{N-2}-\mathcal{O}_{N-1}Q_{N-2}\right\|_2 &\le \left\|\tilde{\mathcal{O}}^2_{N-1}\tilde{Q}_{N-2}-\tilde{\mathcal{O}}^1_{N-1}Q_{N-2}\right\|_2\\ &\quad+\left\|\tilde{\mathcal{O}}^1_{N-1}Q_{N-2}-\mathcal{O}_{N-1}Q_{N-2}\right\|_2\\ &\le 2\tau,\end{aligned} \tag{4.18}$$

which is exactly inequality (4.9) for $i = N-1$.

Till now, we have proven that all the inequalities (4.7) (4.8) (4.9) hold for $i = N-1$.

Next, we suppose that at step $(k+1)$, $2 \le k \le N-2$, the following inequalities hold,

$$\begin{aligned}\left\|\mathcal{O}^1_{k+1}-\tilde{\mathcal{O}}^1_{k+1}\right\|_2 &\le (N-k-1)\tau,\\ \left\|\tilde{\mathcal{O}}^2_{k+1}\tilde{Q}_k-\mathcal{O}_{k+1}Q_k\right\|_2 &\le (N-k)\tau.\end{aligned}$$

Therefore, at step k, we have

$$\begin{aligned}\left\|\tilde{\mathcal{O}}^1_k-\mathcal{O}_k\right\|_2 &= \left\|\begin{bmatrix}P_k\\ \tilde{\mathcal{O}}^2_{k+1}\tilde{R}^1_k\end{bmatrix}-\begin{bmatrix}P_k\\ \mathcal{O}_{k+1}R_k\end{bmatrix}\right\|_2\\ &= \left\|\tilde{\mathcal{O}}^2_{k+1}\tilde{R}^1_k-\mathcal{O}_{k+1}R_k\right\|_2\\ &\le \left\|\tilde{\mathcal{O}}^2_{k+1}\tilde{R}^1_k-\tilde{\mathcal{O}}^1_{k+1}R_k\right\|_2+\left\|\tilde{\mathcal{O}}^1_{k+1}R_k-\mathcal{O}_{k+1}R_k\right\|_2.\end{aligned}$$

And it is easy to obtain

$$\begin{aligned}\left\|\tilde{\mathcal{O}}^2_{k+1}\tilde{R}^1_k-\tilde{\mathcal{O}}^1_{k+1}R_k\right\|_2 &= \left\|\begin{bmatrix}I & \\ & \tilde{\mathcal{O}}^2_{k+1}\end{bmatrix}\Delta U_{k+1}\Delta\Sigma_{k+1}\Delta V^T_{k+1}R_k\right\|_2\\ &= \left\|\Delta U_{k+1}\Delta\Sigma_{k+1}\Delta V^T_{k+1}R_k\right\|_2\\ &\quad(\tilde{\mathcal{O}}^2_{k+1}\text{ has orthonormal columns})\\ &\le \left\|\Delta U_{k+1}\Delta\Sigma_{k+1}\Delta V^T_{k+1}\right\|_2 \quad \text{(Corollary 4.29)}\\ &\le \tau.\end{aligned}$$

Besides

$$\begin{aligned}\left\|\tilde{\mathcal{O}}_{k+1}^1 R_k - \mathcal{O}_{k+1} R_k\right\|_2 &\leq \left\|\tilde{\mathcal{O}}_{k+1}^1 - \mathcal{O}_{k+1}\right\|_2 \|R_k\|_2 \\ &\leq \left\|\tilde{\mathcal{O}}_{k+1}^1 - \mathcal{O}_{k+1}\right\|_2 \quad \text{(Corollary 4.29)} \\ &\leq (N-k-1)\tau.\end{aligned}$$

This gives

$$\left\|\tilde{\mathcal{O}}_k^1 - \mathcal{O}_k\right\|_2 \leq \tau + (N-k-1)\tau = (N-k)\tau,$$

which is exactly the inequality (4.8) for step k.

Next, we start to approximate $\tilde{\mathcal{O}}_k^1$. Since

$$\tilde{\mathcal{O}}_k^1 = \begin{bmatrix} P_k \\ \tilde{\mathcal{O}}_{k+1}^2 \tilde{R}_k^1 \end{bmatrix} = \begin{bmatrix} I & \\ & \tilde{\mathcal{O}}_{k+1}^2 \end{bmatrix} \begin{bmatrix} P_k \\ \tilde{R}_k^1 \end{bmatrix},$$

and $\tilde{\mathcal{O}}_{k+1}^2$ has orthonormal columns, we first perform an SVD on $\begin{bmatrix} P_k \\ \tilde{R}_k^1 \end{bmatrix}$ that gives

$$\begin{bmatrix} P_k \\ \tilde{R}_k^1 \end{bmatrix} = \begin{bmatrix} U_k & \Delta U_k \end{bmatrix} \begin{bmatrix} \Sigma_k & \\ & \Delta\Sigma_k \end{bmatrix} \begin{bmatrix} V_k^T \\ \Delta V_k^T \end{bmatrix}.$$

Let

$$\begin{bmatrix} \tilde{P}_k \\ \tilde{R}_k \end{bmatrix} = U_k, \text{ and } \tilde{\mathcal{C}}_k = \Sigma_k V_k^T \mathcal{C}_k.$$

This yields,

$$\begin{aligned}\left\|\tilde{\mathcal{O}}_k^2 \tilde{\mathcal{C}}_k - \tilde{\mathcal{O}}_k^1 \mathcal{C}_k\right\|_2 &= \left\|\Delta U_k \Delta\Sigma_k \Delta V_k^T \mathcal{C}_k\right\|_2 \\ &= \left\|\Delta U_k \Delta\Sigma_k \Delta V_k^T\right\|_2 \quad (\mathcal{C}_k \text{ has orthonormal rows}) \\ &\leq \tau.\end{aligned}$$

Therefore,

$$\begin{aligned}\left\|\tilde{\mathcal{O}}_k^2 \tilde{\mathcal{C}}_k - \mathcal{O}_k \mathcal{C}_k\right\|_2 &\leq \left\|\tilde{\mathcal{O}}_k^2 \tilde{\mathcal{C}}_k - \tilde{\mathcal{O}}_k^1 \mathcal{C}_k\right\|_2 + \left\|\tilde{\mathcal{O}}_k^1 \mathcal{C}_k - \mathcal{O}_k \mathcal{C}_k\right\|_2 \\ &\leq \tau + \left\|\tilde{\mathcal{O}}_k^1 \mathcal{C}_k - \mathcal{O}_k \mathcal{C}_k\right\|_2 \\ &= \tau + \left\|\tilde{\mathcal{O}}_k^1 - \mathcal{O}_k\right\|_2 \quad (\mathcal{C}_k \text{ has orthonormal rows}) \\ &\leq (N-k+1)\tau,\end{aligned}$$

i.e.,

$$\left\|\tilde{\mathcal{H}}_k - \mathcal{H}_k\right\|_2 \leq (N-k+1)\tau,$$

which proves inequality (4.7) for step k.

Besides,

$$\begin{aligned}\left\|\tilde{\mathcal{O}}_k^2\tilde{Q}_{k-1}-\mathcal{O}_kQ_{k-1}\right\|_2 &\le \left\|\tilde{\mathcal{O}}_k^2\tilde{Q}_{k-1}-\tilde{\mathcal{O}}_k^1Q_{k-1}\right\|_2+\left\|\tilde{\mathcal{O}}_k^1Q_{k-1}-\mathcal{O}_kQ_{k-1}\right\|_2\\ &\le \left\|\tilde{\mathcal{O}}_k^2\tilde{Q}_{k-1}-\tilde{\mathcal{O}}_k^1Q_{k-1}\right\|_2+\left\|\tilde{\mathcal{O}}_k^1-\mathcal{O}_k\right\|_2. \quad \text{(Corollary 4.29)}\end{aligned}$$

It is easy to obtain that

$$\begin{aligned}\left\|\tilde{\mathcal{O}}_k^2\tilde{Q}_{k-1}-\tilde{\mathcal{O}}_k^1Q_{k-1}\right\|_2 &= \left\|\begin{bmatrix}I & \\ & \tilde{\mathcal{O}}_{k+1}^2\end{bmatrix}\Delta U_k\Delta\Sigma_k\Delta V_k^TQ_{k-1}\right\|_2\\ &= \left\|\Delta U_k\Delta\Sigma_k\Delta V_k^TQ_{k-1}\right\|_2 \quad (\tilde{\mathcal{O}}_{k+1}^2 \text{ has orthonormal columns})\\ &\le \left\|\Delta U_k\Delta\Sigma_k\Delta V_k^T\right\|_2\left\|Q_{k-1}\right\|_2\\ &\le \left\|\Delta U_k\Delta\Sigma_k\Delta V_k^T\right\|_2 \quad \text{(Corollary 4.29)}\\ &\le \tau.\end{aligned}$$

Therefore,

$$\left\|\tilde{\mathcal{O}}_k^2\tilde{Q}_{k-1}-\mathcal{O}_kQ_{k-1}\right\|_2 \le \tau+(N-k)\tau=(N-k+1)\tau.$$

This gives the proof of inequality (4.9) for step k. □

5 Chapter

MSSS Preconditioning Technique for Optimal In-Domain Control of PDEs: a Wind Farm Example

In this chapter, we study the multilevel sequentially semiseparable (MSSS) preconditioning techniques for the optimal in-domain control of partial differential equations (PDEs). In contrast to the PDE-constrained optimization problem studied in Chapter 2, where the control inputs are distributed throughout the domain, the control input here only acts on a few grid points in the domain where actuators are placed. This in turn makes the resulting generalized saddle-point system even more difficult to solve since its Schur complement is very difficult or even impossible to approximate. Standard block preconditioners do not give satisfactory performance. We evaluate the performance of the MSSS preconditioning techniques for this type of problems by using a simplified wind farm control example. We will show that by exploiting the multilevel sequentially semiseparable structure, we can compute a robust preconditioner in linear computational complexity. We also study the performance of the state-of-the-art preconditioning techniques and our results show the superiority of the MSSS preconditioning techniques over standard preconditioning techniques for this type of in-domain control problems.

5.1 Introduction

Nowadays, optimal control problems in practice are mostly solved with nonlinear programming (NLP) methods based on some discretization strategies of the original continuous problems in the functional space [79]. Once the optimization problem is discretized, the optimization variable is reduced to a finite-dimensional space. This results in a parameter optimization problem [25]. Simultaneous strategies, explicitly perform a discretization of the PDEs that prescribe the dynamics of the

system as well as the cost function, i.e., PDE simulation and the optimization procedure proceed simultaneously, cf. [21, 58, 113] and Chapter 2 of this dissertation. Sequential strategies, on the other hand, just parameterize the control input and employ numerical schemes in a black-box manner by utilizing the implicit function theorem (IFT) [9, 25, 71]. This approach turns out to be very efficient when the dimension of the control variable is much smaller than the system states that are described by the partial differential equations [25, 71], where optimal shape design [77], boundary control of fluid dynamics [18, 56] are applications for this type of optimization approaches. For the simultaneous approach, its high potential in terms of efficiency and robustness turns out to be very difficult to be realized when the sizes of the states and inputs are very large. This yields a very large optimization problem, where the equality constraints by PDEs are appended to the augmented cost function. The Lagrangian multipliers, the number of which is the same as the state variables of PDEs, make the size of the optimization problem even bigger.

As in many problems in science and engineering, the most time-consuming part for the optimal control of PDEs is to solve a series of linear systems arising from the simulation of PDEs [21]. With increasing improvement of computational resources and the advancement of numerical techniques, very-large problems can be taken into consideration [59]. An important building block for the optimal control of PDEs is the preconditioning techniques to accelerate the simulation of PDEs. In the last decades, many efforts have been dedicated to the development of efficient and robust preconditioning techniques for these types of problems [4, 95, 99, 107, 121]. These research projects are devoted to preconditioning control problems of the tracking type where the control is distributed throughout the domain. For the case of the in-domain control where the control only acts on a few grid points where the actuators are placed, or the boundary control case that only the boundary condition can be controlled, preconditioning techniques for this in-domain control problems do not give satisfactory performance. This is because the developed preconditioning techniques highly depend on an efficient approximation of the Schur complement of the linear system arising from the discretized optimality condition, cf. conclusion parts of [94, 107] for a discussion. Some research for optimal in-domain control problems focuses on developing novel computational techniques for the specific objective [10, 79, 84] without considering efficient preconditioning techniques.

In this chapter, we focus on designing efficient and robust preconditioning techniques for optimal in-domain control of the Navier-Stokes equation and use a simplified wind farm control example to illustrate the performance of our preconditioning technique. Our contributions include: (1) By formulating the optimal in-domain control of the Navier-Stokes problem as a generalized saddle-point problem using the implicit function theorem (IFT), we can reuse the preconditioners for the linearized Navier-Stokes equation to solve the adjoint equations for the computations of the gradient and Hessian matrix. This reduces the computational cost significantly. (2) We study the multilevel sequentially semiseparable (MSSS) preconditioner for the generalized saddle-point system. In contrast to the standard block preconditioners that require to approximate the Schur complement of the block linear system, the MSSS preconditioner computes an approximate factoriza-

tion of the global system matrix up to a prescribed accuracy in linear computational complexity. This is a big advantage over the standard block preconditioners; (3) We evaluate the performance of the MSSS preconditioner using incompressible flow and fast iterative solver software (IFISS)[1] [117]. (4) Our analysis shows that the computational cost can be further reduced by applying block-Krylov methods to solve a linear system with multiple left-hand sides and multiple right-hand sides for the computations of the gradient and Hessian matrix.

The structure of this chapter is organized as follows. In Section 5.2, we use a simplified wind farm control example to formulate an in-domain control problem that is governed by the stationary Navier-Stokes equation. We introduce the optimization techniques for this type of problems as well as the preconditioning techniques based on MSSS matrix computations in Section 5.3. In Section 5.4, we perform numerical experiments to illustrate the performance of the MSSS preconditioning techniques for this type of problems. We also apply the state-of-the-art preconditioning techniques in Section 5.4 as a comparison for evaluation of the MSSS preconditioning techniques. Conclusions are drawn in Section 5.5.

5.2 Problem Formulation

In this section, we use wind farm control as an example to formulate the in-domain control problem. Due to the wake interactions in a wind farm, a reduced velocity profile is obtained after the wind passes through turbines. This in turn reduces the energy extracted by the turbines downstream because the energy captured by the turbines roughly scales with the cubic power of the velocity at the turbine location [68].

Currently, wind turbines are usually controlled by the individual controllers to maximize their own performance. Many research activities illustrate that operating all the turbines in a wind farm at their own optimal state gives sub-optimal performance of the overall wind farm [5, 82]. Wind farm control aims to optimize the total power captured from the wind by taking coordinating control strategies to the turbines in the wind farm. By appropriately choosing the computational domain for the flow (wind farm), the wind farm control can be formulated as an optimal flow control problem that aims to maximize the total power captured by all the turbines in the wind farm. For the wind farm control, the control only acts on a few parts of the domain where the turbines are located. This in turn gives an in-domain control problem. In the next part, we aim to build a simplified wind farm model and use this model to formulate the optimization problem.

5.2.1 Fluid Dynamics

Some efforts have been devoted to develop a suitable model to simulate the wake effect in the wind farm, cf. [38, 112] for a general survey and [5, 57] for recent

[1] IFISS is a computational laboratory for experimenting with state-of-the-art preconditioned iterative solvers for the discrete linear equation systems that arise in incompressible flow modeling. http://www.cs.umd.edu/∼elman/ifiss/index.html

developments. In general there exist two approaches for modeling of the wake. The one is the heuristic approach that does not solve a flow equation but uses some rules of thumb [5, 105], a typical example is the Park model [5]. The second approach is solving an incompressible Navier-Stokes or Euler equation, cf. [59, 122]. In this chapter, we use the second approach to model the flow in the wind farm. Moreover, some recent research tries to take the boundary layer and some physical behavior into account. This in turn requires a more complicated turbulence model for the wind farm simulation study [3, 59, 112]. In this chapter, we focus on preconditioning techniques, and we evaluate the performance of preconditioning techniques by the Incompressible Flow & Iterative Solver Software (IFISS). We only focus on flow problems that can be addressed within the framework of IFISS. Numerical methods that deal with turbulent flow are out of the scope of this dissertation.

Consider the stationary incompressible Navier-Stokes equation in $\Omega \in \mathbb{R}^2$ that is given by

$$\begin{aligned} -\nu\Delta\overrightarrow{u} + (\overrightarrow{u}\cdot\nabla)\overrightarrow{u} + \nabla p &= \overrightarrow{f} \\ \nabla\cdot\overrightarrow{u} &= 0 \end{aligned} \tag{5.1}$$

where ν is the kinematic viscosity, $\overrightarrow{u}$ is the velocity field, p denotes the pressure, $\overrightarrow{f}$ is a source term. Here Ω is a bounded domain with its boundary given by $\Gamma = \partial\Omega = \partial\Omega_D \cup \partial\Omega_N$, where $\partial\Omega_D$ denotes the boundary where Dirichlet boundary conditions are prescribed and $\partial\Omega_N$ represents the boundary where Neumann boundary conditions are imposed. The Reynolds number $Re \in \mathbb{R}^+$ describes the ratio of the inertial and viscous forces within the fluid [131] and is defined by

$$Re \triangleq \frac{u_r L_r}{\nu}, \tag{5.2}$$

where $u_r \in \mathbb{R}^+$ is the reference velocity, $L_r \in \mathbb{R}^+$ is the reference distance that the flow travels.

In this chapter, we focus on designing efficient preconditioning techniques for the control of laminar flow using IFISS. The Reynolds number plays an important role in the flow equation that describes whether the flow under consideration is laminar or turbulent. In many engineering problems, turbulent flow happens when the Reynolds number $Re > 2000$ [131]. In the case of flow through a straight pipe with a circular cross-section, at a Reynolds number below a critical value of approximately 2040, fluid motion will ultimately be laminar, whereas at larger Reynolds numbers, the flow can be turbulent [8]. Since we focus on efficient preconditioning techniques for optimal in-domain flow control using IFISS in this chapter and no turbulent flow model is included in IFISS, we consider a flow with Reynolds number $Re = 2000$, although this does not correspond to practical flow for wind farm control.

To study the aerodynamics of the wind farm, we cannot set an infinite dimensional computational domain. We can prescribe suitable boundary conditions for the flow that in turn gives a finite domain. We set a standard reference domain $\Omega = [-1, 1] \times [-1, 1]$ for the wind farm simulation study in Figure 5.1. The reference velocity u_r is set to 1.

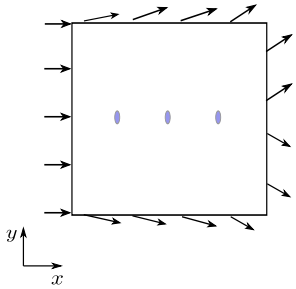


Figure 5.1: Computational domain for wind farm simulation

The turbines in the wind farm are located in a line that follows the stream direction, and the leftmost turbine is placed at the center of the domain. Such configurations are widely used in the wind farm simulation studies [5, 59, 122]. Here the diameter of the turbine is set to be $^1/_{64}$ of the reference domain in Figure 5.1. The distance between turbines is set to be 5 times of the diameter of the turbines. The constant inflow is imposed on the left boundary and is given by

$$u_x = u_c, \quad u_y = 0, \tag{5.3}$$

or equivalently

$$\vec{u} = -u_c \vec{n}. \tag{5.4}$$

Here, $\vec{n}$ is the unit normal vector of the boundary that points outwards. For top, right, and bottom boundary that are far away from the turbines, the flow is considered to be free stream and zero stress boundary condition for outflow given by (5.5) is prescribed,

$$\nu \frac{\partial \vec{u}}{\partial \vec{n}} - p\vec{n} = \mathbf{0}. \tag{5.5}$$

Here, $\frac{\partial \vec{u}}{\partial \vec{n}}$ is the Gâteaux derivative at $\partial\Omega_N$ in the direction of $\vec{n}$. By setting the right hand side of (5.5) to $\mathbf{0}$, the average pressure on the outflow boundary is also set to be 0. This boundary condition states that the flow can move freely across the boundary by resolving the Navier-Stokes equation (5.1). Associated with the prescribed boundary condition (5.4) (5.5), the resolved flow field without outer source is shown in Figure 5.2.

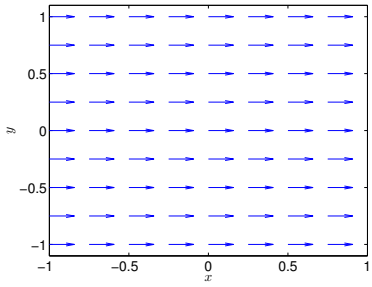


Figure 5.2: Resolved flow field without outer source

5.2.2 Wind Turbines

Turbines in a wind farm capture kinetic energy from the wind and generate electrical power. From the wind farm control perspective, it can be modeled as an outer source that interacts with the flow. There are two models that are widely used to study the turbine interaction with the flow field, one is the actuator disk method (ADM) [85] while the other is the actuator line method [123]. The ADM method is widely used because it is simple and can give satisfactory results in many applications [85]. The ADM models the wind turbine as a disk that can have constant, radial, or variable force that acts on the flow. It does not consider the model of the blades of the turbine, which reduces the computational complexity dramatically. In contrast, the actuator line model takes finite sections of the turbine blades and compute the lift and drag force that act on the flow [123] while the blades geometry and the flow conditions should also be considered. This in turn increases the computational complexity. In this chapter, we take the widely used actuator disk method as the wind turbines model.

According to the classic actuator disk method [76], the thrust on the disk is given by

$$f = 2\rho A a(1-a)u_\infty^2. \tag{5.6}$$

Here ρ is the density of the air, A is the area that is swept by the blades, a is the axial induction factor at rotor plane and is defined by,

$$a = \frac{u_\infty - u_d}{u_\infty} = \frac{1}{2}\frac{u_\infty - u_w}{u_\infty}. \tag{5.7}$$

u_∞ is the velocity in the free stream region ($u_\infty = u_c$) that is shown in Figure 5.3.

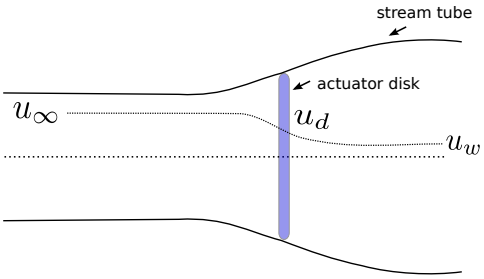


Figure 5.3: Actuator-disk model

For the idealized case, all reduced kinetic energy of the wind is captured by the wind turbine, which can be given as

$$P = \frac{1}{2}\rho A u_\infty^3 a(1-a)^2. \tag{5.8}$$

The optimal power extraction is obtained when $a = {}^1/_3$, which is called the Betz

limit [27]. This in turn gives the optimal thrust force f_m in (5.6) by

$$f_m = \frac{4}{9}\rho A u_\infty^2. \tag{5.9}$$

According to (5.7), we have

$$u_d = u_\infty(1-a). \tag{5.10}$$

For the optimal power extraction, $a = {}^1/_3$, therefore, we have $u_d = {}^2/_3 u_\infty$. Then we obtain,

$$f_m = \rho A u_d^2. \tag{5.11}$$

For modern turbines, the thrust force can be denoted by,

$$f = -C_T \rho A \hat{u}_d^2, \tag{5.12}$$

where $\hat{u}_d$ is the average axial flow velocity at the turbine disk, C_T is the disk based thrust coefficient that depends on the geometry and design of the turbine blades. Nowadays, turbine blades are designed to make C_T maximum around 1.

5.2.3 Objective Function

The operation status for wind turbines can be illustrated by Figure 5.4. In region I, the wind speed is below the cut-in speed that triggers the start of the wind turbine, while in region II, the wind speed is below the rated speed and the wind turbine is set to operate in the maximum power tracking mode. This is performed by the generator torque control to keep a constant tip-speed ratio that is independent of the wind speed while the pitch angle is kept constant at the optimal value of C_T, i.e, $C_T = 1$. In region III, the wind speed is above the rated speed and below the cut out speed, the turbine is operated with rated power output by active yaw control (AYC) and pitch control. This corresponds to reducing C_T to keep the power constant with the increase of the wind speed. The wind turbine is cut out in region IV for protection.

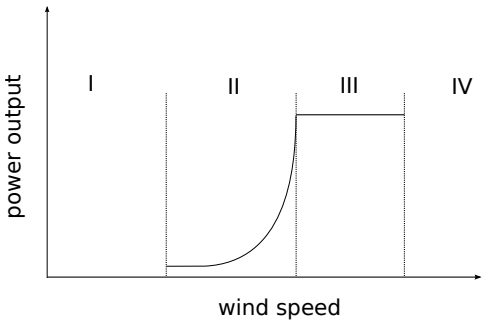


Figure 5.4: Operation region of wind turbine

For wind turbines, the disk based thrust coefficient C_T can be changed from 0 to 1 by active yaw control and pitch control. For conventional wind farm control, all the turbines are operated in their own optimal status in region II, i.e., $C_T = 1$ for all the turbines. Due to the wake interactions, the velocity downstream is reduced, this in turn gives a sub-optimal performance for the wind farm [5]. Here we consider the ideal case that $C_P = C_T$ where C_P is the power coefficient and usually $C_P \leq C_T$. The wind farm control aims to maximize the total power captured by all the wind turbines in the wind farm, which can be represented as

$$P_t = \sum_{j=1}^{N_t} -f_j \hat{u}_j = \rho A \sum_{j=1}^{N_t} C_{T_j} \hat{u}_j^3, \tag{5.13}$$

where N_t is the number of turbines in the wind farm, ρ is the density of air, A is the area swept by the wind turbine blades, and $\hat{u}_j$ is the uniform disk averaged axial flow speed of the j-th turbine.

Therefore, the wind farm control problem can be formulated as the following optimization problem,

$$\begin{aligned} \min_{C_T, \overrightarrow{u}} \quad & -\sum_{j=1}^{N_t} C_{T_j} \hat{u}_j^3 \\ \text{s.t.} \quad & -\nu \Delta \overrightarrow{u} + (\overrightarrow{u} \cdot \nabla) \overrightarrow{u} + \nabla p = \overrightarrow{f}(C_T, \overrightarrow{u}), \\ & \nabla \cdot \overrightarrow{u} = 0, \\ & 0 \leq C_{T_j} \leq 1, \ (j = 1, \ldots, N_t). \end{aligned} \tag{5.14}$$

Here $\overrightarrow{f}(C_T, \overrightarrow{u})$ is a nonlinear function and it is of value (5.12) at the position where turbines are placed and 0 elsewhere, and $C_T = \begin{bmatrix} C_{T_1} & C_{T_2} & \cdots & C_{T_{N_t}} \end{bmatrix}^T$.

5.3 Reduced Nonlinear Programming

In the previous section, we formulated an in-domain control problem (5.14) by using a wind farm control example. The size of the control variable N_t, which is the number of turbines, is much smaller than the size of the state variables (number of velocity and pressure grid points). Therefore, we use the sequential approach that is based on the implicit function theorem to solve a reduced optimization problem.

5.3.1 Reduced Optimization Problem

Denote the equality constraints in (5.14) for the flow equation by $h(C_T, \phi) = 0$ where $\phi = (\overrightarrow{u}, p)$, and the objective function by $g(C_T, \phi)$, then the optimization problem (5.14) can be written as

$$\begin{aligned} \min_{C_T,\phi} \quad & g(C_T,\phi) \\ \text{s.t.} \quad & h(C_T,\phi)=0, \\ & 0 \le C_T \le 1. \end{aligned} \tag{5.15}$$

The equality constraint in (5.14) implies that the velocity and pressure ϕ is a function of $\overrightarrow{f}$. For the existence of this function, cf. [71] for a proof. Since $\overrightarrow{f}$ is an explicit function of C_T, ϕ is a function of C_T and denote this function by $\phi = s(C_T)$. The function $s(C_T)$ is not explicitly computed but is obtained implicitly by solving the flow equation (5.1) using given C_T.

By using the implicit function theorem, we can re-write the optimization problem in a reduced form,

$$\begin{aligned} \min_{C_T} \quad & g(C_T, s(C_T)) \\ \text{s.t.} \quad & 0 \le C_T \le 1. \end{aligned} \tag{5.16}$$

Newton-type methods, which are second order methods, are well-suited to solve this type of nonlinear programming (NLP) problems. An alternative approach to solve this type of problem is the (reduced) sequentially quadratic programming ((R)SQP) [88]. For the reduced optimization problem (5.16), these two types of methods are quite similar and we refer to [71] for a detailed discussion.

In this section, we apply Newton's method to solve the reduced NLP problem (5.16). The reason to choose the Newton's method is that the Hessian matrix for this problem is of small size and can be computed explicitly. Moreover, we can reuse the information from the last Newton step of solving the nonlinear flow equation to compute the gradient and the Hessian matrix, which makes Newton's method computationally competitive for this optimization problem. This will be explained in the following part. The Newton's method for this problem is described by Algorithm 5.1.

In Algorithm 5.1, we denote the optimization step as the outer iteration, and at each outer iteration, we need to solve a Navier-Stokes equation with nonlinear right-hand side. This step is denoted by the inner iteration in line 4 of the algorithm. From the description of Algorithm 5.1, it is clear that the most time-consuming part for this optimization problem is the solution of the nonlinear flow equation and the computations of the gradient and Hessian matrix. Therefore, efficient numerical methods need to be deployed.

Algorithm 5.1 Reduced Newton's algorithm for (5.15)

1: **procedure** OPT(Wind) ▷ Optimization procedure for wind farm control
2: **Input:** Initial guess $C_T^{(0)}$, maximal optimization steps $\mathtt{it}_{\max}$, stop tolerance ε_0
3: **while** $\|\nabla g_k\| > \varepsilon_0$ && $k \le \mathtt{it}_{\max}$ **do** ▷ outer iteration
4: solve $h(C_T^{(k)}, \phi) = 0$ to compute $\phi^{(k)}$ ▷ inner iteration
5: evaluate the gradient ∇g_k at $(C_T^{(k)}, \phi^{(k)})$
6: evaluate the Hessian matrix H_k at $(C_T^{(k)}, \phi^{(k)})$
7: compute the update $\Delta C_T^{(k)} = \arg\min \Delta C_T^T H_k \Delta C_T + \nabla g_k^T \Delta C_T$
8: $C_T^{(k+1)} \leftarrow C_T^{(k)} + \Delta C_T^{(k)}$
9: Check inequality constraints by projection
10: **if** $C_{T_j} > 1$ **then**
11: $C_{T_j} = 1$ ▷ project on boundary
12: **else if** $C_{T_j} < 0$ **then**
13: $C_{T_j} = 0$ ▷ project on boundary
14: **end if**
15: $k \leftarrow k + 1$
16: **end while**
17: **Output:** Optimal control variable C_T and corresponding solution of u
18: **end procedure**

5.3.2 Preconditioning Flow Equation

In this part, we introduce efficient numerical methods for Algorithm 5.1. At each outer iteration, we need to solve a nonlinear flow equation that has a nonlinear right-hand side at step 4 of Algorithm 5.1, we explicitly write this equation in decomposed form as

$$
\begin{aligned}
-\nu(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2})u_x + \overbrace{(u_x\frac{\partial}{\partial x} + u_y\frac{\partial}{\partial y})u_x}^{\text{velocity convection}} + \frac{\partial}{\partial x}p &= f_x(C_T, u_x, u_y),\\
-\nu(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2})u_y + \underbrace{(u_x\frac{\partial}{\partial x} + u_y\frac{\partial}{\partial y})u_y}_{\text{velocity convection}} + \frac{\partial}{\partial y}p &= f_y(C_T, u_x, u_y),\\
\frac{\partial}{\partial x}u_x + \frac{\partial}{\partial y}u_y &= 0.
\end{aligned}
\tag{5.17}
$$

Equation (5.17) is a nonlinear equation where the nonlinearity is caused by both the velocity convection operator and the nonlinear right-hand side. To solve such a nonlinear equation (5.17), we apply Newton's method. At each Newton iteration of step 4 in Algorithm 5.1, we need to solve a linear system of the following type,

$$
\begin{bmatrix}
\nu K + N + J_{xx}^n + J_{xx}^f & J_{xy}^n + J_{xy}^f & B_x^T \\
J_{yx}^n + J_{yx}^f & \nu K + N + J_{yy}^n + J_{yy}^f & B_y^T \\
B_x & B_y & 0
\end{bmatrix}
\begin{bmatrix} \Delta u_x \\ \Delta u_y \\ \Delta p \end{bmatrix}
=
\begin{bmatrix} a \\ b \\ c \end{bmatrix},
\tag{5.18}
$$

after discretizing the nonlinear partial differential equation (5.17) using mixed finite element method. Here N is the convection matrix, $J^n_{(\cdot)}$ denote the Jacobian matrices from the nonlinear velocity convection term, and $J^f_{(\cdot)}$ represent the Jacobian matrices from the nonlinear right-hand side. Since only a small part of the domain is controlled, f_x and f_y are zero almost everywhere in the domain except in the vicinity where the turbines are placed. This in turn gives singular Jacobian matrices $J^f_{(\cdot)}$.

Comparing system (5.18) with the standard linearized Navier-Stokes equation by the Newton's method given by

$$\begin{bmatrix} \nu K + N + J^n_{xx} & J^n_{xy} & B_x^T \\ J^n_{yx} & \nu K + N + J^n_{yy} & B_y^T \\ B_x & B_y & 0 \end{bmatrix} \begin{bmatrix} \Delta u_x \\ \Delta u_y \\ \Delta p \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}, \tag{5.19}$$

we see that the linearized flow equation (5.18) is a perturbed linearized Navier-Stokes equation with singular perturbation in the matrix blocks that correspond to the velocity. Re-write equation (5.19) in a compact form as

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \Delta u \\ \Delta p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}, \tag{5.20}$$

and the equation (5.18) is given by a perturbed form

$$\begin{bmatrix} A + J^f & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \Delta u \\ \Delta p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}. \tag{5.21}$$

The linear system (5.21) is large, sparse, and highly indefinite. Efficient preconditioning techniques are needed to solve such systems using Krylov solvers. Standard preconditioning techniques for such 2-by-2 system highly depend on efficient approximation of the Schur complement, or computing a spectrally equivalent Schur complement as introduced in Chapter 3. For the linearized Navier-Stokes problem (5.20), there exits some well-established preconditioning techniques such as the SIMPLE method [80], augmented Lagrangian preconditioner [17], pressure convection-diffusion (PCD) preconditioner [74], et al. However, the aforementioned preconditioners do not perform well to solve the perturbed linear system (5.21) because the Schur complement for the perturbed linear system is even more difficult to approximate.

As introduced in Chapter 3, all the blocks of the matrix in (5.18) have an MSSS structure, it is therefore natural to permute the matrix with MSSS blocks into a global MSSS matrix by applying Lemma 2.4. With this permutation, we can compute an approximate LU factorization using Lemma 2.1. This factorization gives a global MSSS preconditioner for the system (5.18). We have already shown the superiority of the MSSS preconditioning technique over state-of-the-art preconditioning techniques for the computational fluid dynamics (CFD) problems in Chapter 3. We will show the performance of the MSSS preconditioning techniques for the flow problem described by (5.18) and compare its performance with state-of-the-art preconditioning techniques in Section 5.4.

5.3.3 Computations of Partial Derivatives

Next, we introduce efficient numerical methods to compute the reduced gradient and the reduced Hessian matrix. Denote the reduced gradient of the cost function by

$$\nabla g = \left[\frac{\partial}{\partial C_{T_1}} g \quad \frac{\partial}{\partial C_{T_2}} g \quad \cdots \quad \frac{\partial}{\partial C_{T_{N_t}}} g \right]^T. \tag{5.22}$$

The gradient can be computed using

$$\frac{\partial}{\partial C_{T_j}} g = \frac{\partial g}{\partial C_{T_j}} + \frac{\partial g}{\partial u_x}\frac{\partial u_x}{\partial C_{T_j}} + \frac{\partial g}{\partial u_y}\frac{\partial u_y}{\partial C_{T_j}}, \quad (j = 1, 2, \ldots, N_t). \tag{5.23}$$

Since the cost function g is an analytic function of C_T, u_x and u_y, the partial derivatives $\frac{\partial g}{\partial C_{T_j}}$, $\frac{\partial g}{\partial u_x}$, and $\frac{\partial g}{\partial u_y}$ are trivial to compute. Next, we show how to compute the partial derivatives $\frac{\partial u_x}{\partial C_{T_j}}$, and $\frac{\partial u_y}{\partial C_{T_j}}$.

Assume that u_x, u_y, and p are twice differentiable with respect to $C_{T_j} (j = 1, 2, \ldots, N_t)$, and that the first and second order derivatives have continuous second order derivative in Ω, i.e.,

$$\frac{\partial u_x}{\partial C_{T_j}}, \quad \frac{\partial u_y}{\partial C_{T_j}}, \quad \frac{\partial p}{\partial C_{T_j}} \in C^2(\Omega),$$

and

$$\frac{\partial^2 u_x}{\partial C_{T_i} \partial C_{T_j}}, \quad \frac{\partial^2 u_y}{\partial C_{T_i} \partial C_{T_j}}, \quad \frac{\partial^2 p}{\partial C_{T_i} \partial C_{T_j}} \in C^2(\Omega),$$

for $(i,\ j = 1, 2, \ldots, N_t)$.

According to the flow equation (5.17), we have the first order derivative given by

$$\begin{aligned}
&-\nu \overbrace{\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right)}^{\text{diffusion operator}} \left(\frac{\partial u_x}{\partial C_{T_j}} \right) + \overbrace{\left(u_x \frac{\partial}{\partial x} + u_y \frac{\partial}{\partial y} \right)}^{\text{convection operator}} \left(\frac{\partial u_x}{\partial C_{T_j}} \right) + \overbrace{\left(\frac{\partial u_x}{\partial x}\frac{\partial u_x}{\partial C_{T_j}} + \frac{\partial u_x}{\partial y}\frac{\partial u_y}{\partial C_{T_j}} \right)}^{\text{linear term}} \\
&\qquad + \frac{\partial}{\partial x} \left(\frac{\partial p}{\partial C_{T_j}} \right) = \frac{\partial}{\partial C_{T_j}} f_x(C_T, u_x, u_y), \\
&-\nu \underbrace{\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right)}_{\text{diffusion operator}} \left(\frac{\partial u_y}{\partial C_{T_j}} \right) + \underbrace{\left(u_x \frac{\partial}{\partial x} + u_y \frac{\partial}{\partial y} \right)}_{\text{convection operator}} \left(\frac{\partial u_y}{\partial C_{T_j}} \right) + \underbrace{\left(\frac{\partial u_y}{\partial x}\frac{\partial u_x}{\partial C_{T_j}} + \frac{\partial u_y}{\partial y}\frac{\partial u_y}{\partial C_{T_j}} \right)}_{\text{linear term}} \\
&\qquad + \frac{\partial}{\partial y} \left(\frac{\partial p}{\partial C_{T_j}} \right) = \frac{\partial}{\partial C_{T_j}} f_y(C_T, u_x, u_y), \\
&\qquad \frac{\partial}{\partial x} \left(\frac{\partial u_x}{\partial C_{T_j}} \right) + \frac{\partial}{\partial y} \left(\frac{\partial u_y}{\partial C_{T_j}} \right) = 0.
\end{aligned} \tag{5.24}$$

Here the partial differential equation (5.24) is obtained by computing the first order

derivative with respect to $C_{T_j}(j = 1, 2, \ldots, N_t)$ for the flow equation (5.17) and making use of the linearity of the diffusion operator and the divergence operator.

Note that equation (5.24) is a linear partial differential equation. It is often called the adjoint equation of (5.17). Homogeneous Dirichlet boundary conditions are imposed on the constant inflow boundary and natural boundary conditions are prescribed for the outflow boundary for the adjoint equation (5.24). We use a mixed-finite element method to discretize (5.24) and use the following notations

$$\chi_j = \begin{bmatrix} \left(\frac{\partial u_x}{\partial C_{T_j}}\right)^T & \left(\frac{\partial u_y}{\partial C_{T_j}}\right)^T & \left(\frac{\partial p}{\partial C_{T_j}}\right)^T \end{bmatrix}^T.$$

We reuse the same notation to denote its discrete analog to get a linear system given by

$$\underbrace{\begin{bmatrix} \nu K + N_l + J_{xx}^{n_l} + J_{xx}^f & J_{xy}^{n_l} + J_{xy}^f & B_x^T \\ J_{yx}^{n_l} + J_{yx}^f & \nu K + N_l + J_{yy}^{n_l} + J_{yy}^f & B_y^T \\ B_x & B_y & 0 \end{bmatrix}}_{\mathcal{A}} \chi_j = \zeta_j, \tag{5.25}$$

where N_l is the convection matrix, $J_{xx}^{n_l}$, $J_{xy}^{n_l}$, $J_{yx}^{n_l}$, and $J_{yy}^{n_l}$ are the corresponding Jacobian matrices by linearizing the nonlinear velocity convection operator, respectively. J_{xx}^f, J_{xy}^f, J_{yx}^f, and J_{yy}^f are the associating Jacobian matrices that linearize the nonlinear right-hand side f with respect to C_{T_j}. They are all from the last Newton step of the linearized flow equation (5.18). The right-hand side vector ζ_j is obtained by discretizing known variables in (5.24). Therefore, the matrix $\mathcal{A}$ is nothing but the linear system matrix from the last Newton step for solving the flow equation (5.18). The computed preconditioner for the last Newton step to solve the flow equation can be reused. Therefore, the computational work is much reduced.

By stacking χ_j, and ζ_j $(j = 1,\ 2,\ \ldots,\ N_t)$ as

$$\chi = \begin{bmatrix} \chi_1 & \chi_2 & \cdots & \chi_{N_t} \end{bmatrix}, \text{ and } \zeta = \begin{bmatrix} \zeta_1 & \zeta_2 & \cdots & \zeta_{N_t} \end{bmatrix},$$

we obtain the following linear equations with multiple left-hand sides and multiple right-hand sides

$$\mathcal{A}\chi = \zeta. \tag{5.26}$$

Here the size of unknowns N_t, is much smaller than the problem size. Block Krylov subspace methods, such as block IDR(s) [2, 43], are well-suited to solve all the unknowns simultaneously for this type of problems.

Next, we use the same idea as above to compute the Hessian matrix H, which is given by

$$H = \begin{bmatrix} \frac{\partial^2}{\partial C_{T_1}^2} g & \cdots & \frac{\partial^2}{\partial C_{T_1} \partial C_{T_{N_t}}} g \\ \vdots & \ddots & \vdots \\ \frac{\partial^2}{\partial C_{T_{N_t}} \partial C_{T_1}} g & \cdots & \frac{\partial^2}{\partial C_{T_{N_t}}^2} g \end{bmatrix}.$$

According to (5.23), we have

$$\begin{aligned}\frac{\partial^2}{\partial C_{T_j}\partial C_{T_k}}g = &\frac{\partial^2 g}{\partial C_{T_j}\partial C_{T_k}} + \frac{\partial^2 g}{\partial u_x^2}\frac{\partial u_x}{\partial C_{T_k}}\frac{\partial u_x}{\partial C_{T_j}} + \frac{\partial g}{\partial u_x}\frac{\partial^2 u_x}{\partial C_{T_j}\partial C_{T_k}} \\ &+ \frac{\partial^2 g}{\partial u_y^2}\frac{\partial u_y}{\partial C_{T_k}}\frac{\partial u_y}{\partial C_{T_j}} + \frac{\partial g}{\partial u_y}\frac{\partial^2 u_y}{\partial C_{T_j}\partial C_{T_k}}.\end{aligned} \tag{5.27}$$

Since g is an analytic function of C_T, u_x, u_y and we have already computed the first order derivative $\frac{\partial u_x}{\partial C_{T_i}}$, and $\frac{\partial u_y}{\partial C_{T_i}}$ in the previous step, we just need to compute the second order derivatives $\frac{\partial^2 u_x}{\partial C_{T_j}\partial C_{T_k}}$, and $\frac{\partial^2 u_y}{\partial C_{T_j}\partial C_{T_k}}$ for the computations of the Hessian matrix. Here we use the same idea as we used in the previous part to compute the first order derivatives.

Compute the partial derivative with respect to C_{T_k} using the adjoint equation (5.24), we get

$$\begin{aligned}
&-\nu\overbrace{\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right)}^{\text{diffusion operator}}\left(\frac{\partial^2 u_x}{\partial C_{T_j}\partial C_{T_k}}\right)+\overbrace{\left(u_x\frac{\partial}{\partial x}+u_y\frac{\partial}{\partial y}\right)}^{\text{convection operator}}\left(\frac{\partial^2 u_x}{\partial C_{T_j}\partial C_{T_k}}\right)\\
&+\overbrace{\left(\frac{\partial u_x}{\partial x}\frac{\partial^2 u_x}{\partial C_{T_j}\partial C_{T_k}}+\frac{\partial u_x}{\partial y}\frac{\partial^2 u_y}{\partial C_{T_j}\partial C_{T_k}}\right)}^{\text{linear term}}+\frac{\partial}{\partial x}\left(\frac{\partial^2 p}{\partial C_{T_j}\partial C_{T_k}}\right)\\
&+\overbrace{\left(\frac{\partial u_x}{\partial C_{T_k}}\frac{\partial}{\partial x}\left(\frac{\partial u_x}{\partial C_{T_j}}\right)+\frac{\partial u_y}{\partial C_{T_k}}\frac{\partial}{\partial y}\left(\frac{\partial u_x}{\partial C_{T_j}}\right)\right)}^{\text{known}}\\
&+\overbrace{\left(\frac{\partial u_x}{\partial C_{T_j}}\frac{\partial}{\partial x}\left(\frac{\partial u_x}{\partial C_{T_k}}\right)+\frac{\partial u_y}{\partial C_{T_j}}\frac{\partial}{\partial y}\left(\frac{\partial u_x}{\partial C_{T_k}}\right)\right)}^{\text{known}}=\frac{\partial^2}{\partial C_{T_j}\partial C_{T_k}}f_x(C_T,u_x,u_y),\\
&-\underbrace{\nu\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right)}_{\text{diffusion operator}}\left(\frac{\partial^2 u_y}{\partial C_{T_j}\partial C_{T_k}}\right)+\underbrace{\left(u_x\frac{\partial}{\partial x}+u_y\frac{\partial}{\partial y}\right)}_{\text{convection operator}}\left(\frac{\partial^2 u_y}{\partial C_{T_j}\partial C_{T_k}}\right)\\
&+\underbrace{\left(\frac{\partial u_y}{\partial x}\frac{\partial^2 u_x}{\partial C_{T_j}\partial C_{T_k}}+\frac{\partial u_y}{\partial y}\frac{\partial^2 u_y}{\partial C_{T_j}\partial C_{T_k}}\right)}_{\text{linear term}}+\frac{\partial}{\partial x}\left(\frac{\partial^2 p}{\partial C_{T_j}\partial C_{T_k}}\right)\\
&+\underbrace{\left(\frac{\partial u_x}{\partial C_{T_k}}\frac{\partial}{\partial x}\left(\frac{\partial u_y}{\partial C_{T_j}}\right)+\frac{\partial u_y}{\partial C_{T_k}}\frac{\partial}{\partial y}\left(\frac{\partial u_y}{\partial C_{T_j}}\right)\right)}_{\text{known}}\\
&+\underbrace{\left(\frac{\partial u_x}{\partial C_{T_j}}\frac{\partial}{\partial x}\left(\frac{\partial u_y}{\partial C_{T_k}}\right)+\frac{\partial u_y}{\partial C_{T_j}}\frac{\partial}{\partial y}\left(\frac{\partial u_y}{\partial C_{T_k}}\right)\right)}_{\text{known}}=\frac{\partial^2}{\partial C_{T_j}\partial C_{T_k}}f_y(C_T,u_x,u_y),\\
&\frac{\partial}{\partial x}\left(\frac{\partial^2 u_x}{\partial C_{T_j}\partial C_{T_k}}\right)+\frac{\partial}{\partial y}\left(\frac{\partial^2 u_y}{\partial C_{T_j}\partial C_{T_k}}\right)=0.
\end{aligned} \tag{5.28}$$

We see that (5.28) is a linear partial differential equation and it is also an adjoint

equation of the flow equation (5.17). Boundary conditions for the adjoint equation (5.28) are set by homogeneous Dirichlet boundary conditions on the constant inflow boundary and natural boundary conditions for the outflow boundary. By using mixed finite element method to discretize the equation (5.28), we have

$$\underbrace{\begin{bmatrix} \nu K + N_l + J_{xx}^{n_l} + J_{xx}^{f'} & J_{xy}^{n_l} + J_{xy}^{f'} & B_x^T \\ J_{yx}^{n_l} + J_{yx}^{f'} & \nu K + N_l + J_{yy}^{n_l} + J_{yy}^{f'} & B_y^T \\ B_x & B_y & 0 \end{bmatrix}}_{\mathcal{A}'} \gamma_{jk} = \xi_{jk}, \tag{5.29}$$

Here we reuse $\frac{\partial^2}{\partial C_{T_j} \partial C_{T_k}} u_x$, $\frac{\partial^2}{\partial C_{T_j} \partial C_{T_k}} u_y$, $\frac{\partial^2}{\partial C_{T_j} \partial C_{T_k}} p$ to represent their discrete analog, respectively, and $\gamma_{jk} = \begin{bmatrix} \frac{\partial^2}{\partial C_{T_j} \partial C_{T_k}} u_x^T & \frac{\partial^2}{\partial C_{T_j} \partial C_{T_k}} u_y^T & \frac{\partial^2}{\partial C_{T_j} \partial C_{T_k}} p^T \end{bmatrix}^T$.

For the linear system (5.29), $J_{xx}^{n_l}$, $J_{xy}^{n_l}$, $J_{yx}^{n_l}$, and $J_{yy}^{n_l}$ are the corresponding Jacobian matrices as introduced in (5.25). The Jacobian matrices $J_{(\cdot)}^{f'}$ is obtained by computing the second order derivatives of f_x, f_y with respect to C_T and using partial differentiation rules. It is not difficult to see that $J_{(\cdot)}^{f'}$ are identical to $J_{(\cdot)}^{f}$ in (5.25). Therefore, the linear system (5.29) has the same system matrix as the linear system (5.25), i.e., $\mathcal{A}' = \mathcal{A}$. Moreover, $\mathcal{A}$ is just the system matrix from the last Newton step to solve the nonlinear flow equation (5.17). The preconditioners computed for the last Newton step can also be reused here.

The right-hand side vector ξ_{jk} is obtained by discretizing known variables in (5.28). By stacking the unknowns in the following way,

$$\gamma = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \dots & \gamma_{N_t N_t} \end{bmatrix} \quad \text{and} \quad \xi = \begin{bmatrix} \xi_{11} & \xi_{12} & \dots & \xi_{N_t N_t} \end{bmatrix},$$

We get the following linear system with multiple left-hand and right-hand sides,

$$\mathcal{A}\gamma = \xi. \tag{5.30}$$

Here the size of unknowns N_t^2 is also much smaller than the problem size, block Krylov methods are still well-suited to this type of problems.

5.4 Numerical Experiments

In the previous sections, we use the wind farm control example to formulate an in-domain control problem. It is shown that this in-domain control problem can be solved by the reduced Newton method described by Algorithm 5.1. The biggest computational issue is to solve a nonlinear flow equation (5.17) by using the Newton's method and two adjoint equations (5.25) (5.29) to compute the gradient and the Hessian matrix. At each outer iteration step, we solve the nonlinear flow equation (5.17) with inner iterations. A preconditioned Krylov solver is performed at each inner iteration. Efficient and robust preconditioning techniques are necessary.

In Section 5.3, we showed that the linearized flow equation (5.18) is a perturbed linearized Navier-Stokes equation (5.19) with singular perturbation on the $(1,1)$

block of the linearized Navier-Stokes system matrix. Standard preconditioning techniques for the Navier-Stokes equation, which highly depend on efficient approximation of the Schur complement, fail to give satisfactory performance for this problem due to this perturbation. This will be illustrated by numerical results in the next section. Just as introduced in Chapter 2 and 3, all the blocks of (5.18) have an MSSS structure, it is natural to permute the matrix with MSSS blocks into a global MSSS matrix. With this permutation, we can compute an approximate factorization with prescribed accuracy based on MSSS matrix computations which is introduced in Chapter 4. This in turn gives a global MSSS preconditioner.

In this section, we evaluate the performance of the MSSS preconditioning techniques for the in-domain control problem and compare with the pressure convection diffusion (PCD) preconditioner [53] that is implemented in IFISS. All the numerical experiments are implemented in MATLAB 2015a on a desktop of Intel Core i5 CPU of 3.10 GHz and 16 Gb memory with the Debian GNU/Linux 8.0 system.

5.4.1 Preconditioning Techniques

In this part, we report the performance of the MSSS preconditioner and the PCD preconditioner for the second inner iteration of the first outer iteration. We use the `IDR(s)` method [127] to solve the preconditioned system. The preconditioned `IDR(s)` solver is stopped if the the 2-norm of the residual at step k, which is denoted by $\|r_k\|_2$, satisfies $\|r_k\|_2 \leq 10^{-4} \|r_0\|_2$.

The PCD preconditioner $\mathcal{P}_p$ for the linear system (5.21) is chosen as,

$$\mathcal{P}_p = \begin{bmatrix} A + J^f & B^T \\ & -S_p \end{bmatrix}, \tag{5.31}$$

where $S_p = L_p A_p^{-1} M_p$ is the approximation of the Schur complement $BA^{-1}B^T$. Here, A_p and L_p are the convection-diffusion operator and Laplace operator in the finite dimensional solution space of the pressure with prescribed boundary conditions, M_p is the pressure mass matrix. For this PCD preconditioner, both $A + J^f$ and S_p are approximated by the algebraic multigrid (AMG) method that is implemented in IFISS.

We set the Reynolds number $Re = 2000$ as discussed in the previous section. We report the performance of both preconditioners in Table 5.1-5.2. Here the column "precon." represents the time to compute the MSSS preconditioner or the time for the setup of the AMG method, and the column "IDR(4)" denotes the time of the preconditioned `IDR(4)` solver to compute the solution up to prescribed accuracy. Both time are measured in seconds.

It is shown in Table 5.1 the time to compute the MSSS preconditioner scales linearly with the problem size. The number of iterations remains virtually constant with the refinement of the mesh. The time of the preconditioned `IDR(4)` solver also scales linearly with the problem size. For PCD preconditioner, the preconditioned `IDR(4)` solver fails to converge to the solution of prescribed accuracy within 400

iterations for relatively bigger mesh size. For a very fine mesh, the preconditioned `IDR(4)` solver computes the solution up to the prescribed accuracy within 300 iterations. This is because the entries of perturbation by the matrix J^f in (5.31), which is introduced by the nonlinear right-hand side, is of $\mathcal{O}(h^2)$. As $h \to 0$, this perturbation by the Jacobian matrix becomes smaller compared with A in the $(1,1)$ block of (5.31). Therefore, S_p approximates the perturbed Schur complement well.

Table 5.1: Computational results of the MSSS preconditioner for $Re = 2000$

grid	problem size	# iter.	precon. (sec.)	IDR(4) (sec.)	total (sec.)
64×64	$1.25e+04$	2	4.36	0.64	5.00
128×128	$4.97e+04$	3	18.43	2.53	20.96
256×256	$1.98e+05$	5	65.09	9.25	74.34
512×512	$7.88e+05$	3	272.63	24.62	297.25

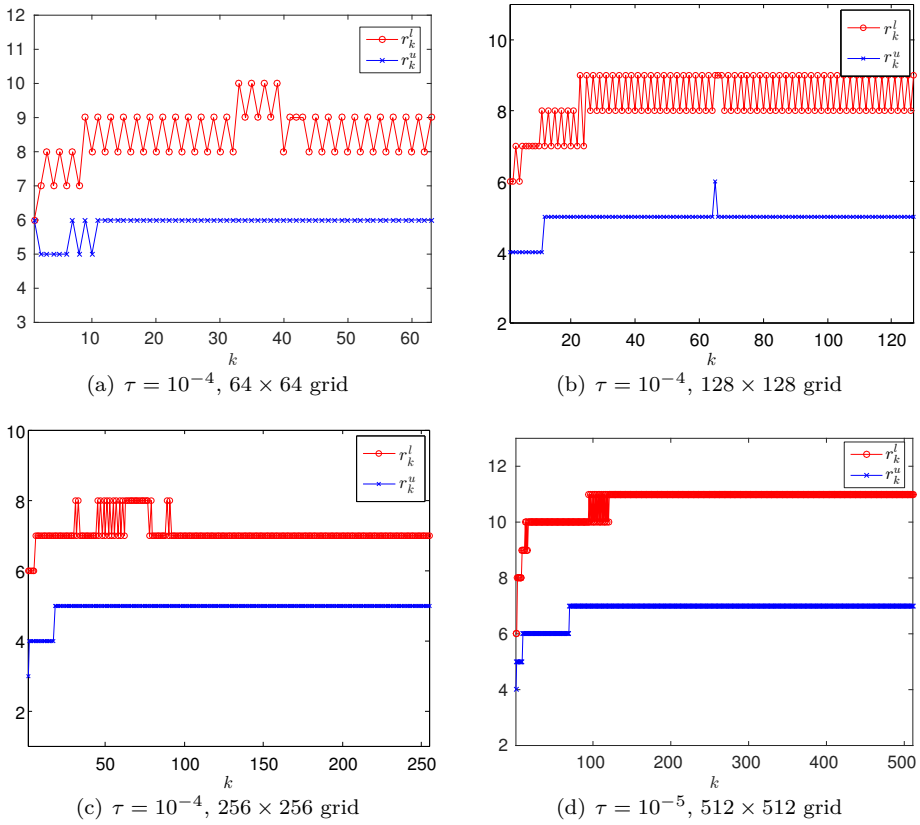
Table 5.2: Computational results of the PCD preconditioner for $Re = 2000$

grid	problem size	# iter.	precon. (sec.)	IDR(4) (sec.)	total (sec.)
64×64	$1.25e+04$	400	8.56	no convergence	-
128×128	$4.97e+04$	400	70.74	no convergence	-
256×256	$1.98e+05$	266	237.68	42.93	280.61
512×512	$7.88e+05$	203	1386.98	101.72	1488.70

The computational results by the MSSS preconditioner in Table 5.2 show that the time for the setup of AMG does not scale linearly with the problem size. The reason may be that the AMG implemented in IFISS is not optimized, or the AMG algorithm in IFISS does not have linear computational complexity.

To compute the MSSS preconditioner, we set $\tau = 10^{-5}$ for the 512×512 grid and 10^{-4} for the rest grids. Here τ is the upper bound of the discarded singular values for the model order reduction that is explained in Chapter 4. The adaptive upper semiseparable order r_k^u and the lower semiseparable order r_k^l for this set up is plotted in Figure 5.5.

The semiseparable order in Figure 5.5 show that the upper and lower semiseparable order is bounded by a small constant around 10 for all the computations of the MSSS preconditioners. This in turn gives the linear computational complexity of the MSSS preconditioning techniques, which is illustrated by Table 5.5.



(a) $\tau = 10^{-4}$, 64×64 grid

(b) $\tau = 10^{-4}$, 128×128 grid

(c) $\tau = 10^{-4}$, 256×256 grid

(d) $\tau = 10^{-5}$, 512×512 grid

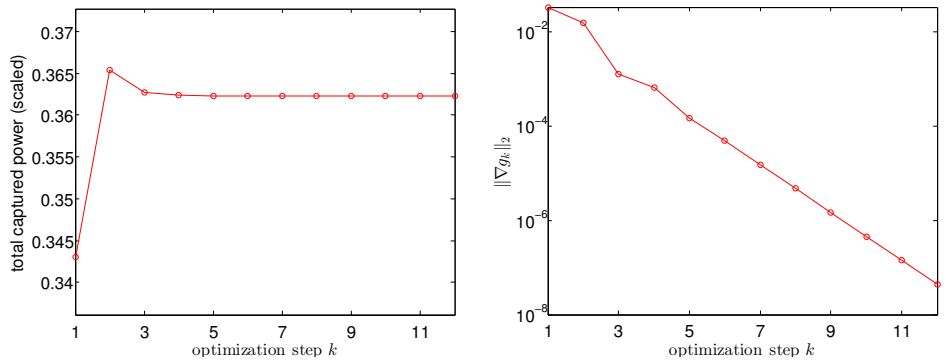
Figure 5.5: Adaptive semiseparable order to compute the MSSS preconditioner for $\nu = 10^{-3}$

5.4.2 Optimization Algorithm

We test Algorithm 5.1 for the optimization problem (5.15) by using a 256×256 grid. At each outer iteration, we need to solve a series of linear equations of size 197634×197634 to compute the solution of the nonlinear flow equation (5.17) by the Newton's method. We also need to solve two adjoint equations (5.25) and (5.29) of the same size to compute the gradient and the Hessian matrix.

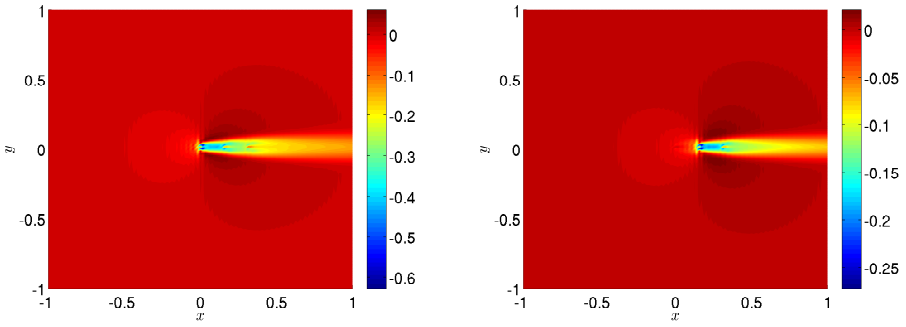
We use the wind farm configuration as introduced in Section 5.2.1. With this problem settings, the rightmost turbine just operates in the maximum power tracking mode, i.e., $C_{T_3} = 1$. Therefore, we just need to optimize C_{T_1} and C_{T_2} for the first two turbines. Without optimization, the turbines are operated in the mode that corresponds to the maximal power extraction for a single turbine, i.e., $C_T = 1$. We start with $C_{T_1}^{(0)} = C_{T_2}^{(0)} = 1$ as an initial guess for this optimization problem. Then the (scaled) total extracted power by the wind farm at each optimization step is given in Figure 5.6(a), while the 2-norm of the gradient at each optimization step is shown by Figure 5.6(b). Results in Figure 5.6(a) show that the total power is increased by around 5.5% when applying the optimal control scheme.

Note that there is an overshoot after the second optimization step as illustrated by Figure 5.6(a), which makes the optimization problem to converge to a local optimum. This is primarily because of the non-convexity and highly nonlinearity of this optimization problem. The convexity of this optimization problem is still an open problem. Another reason that contributes to this behavior is the sensitivity of this optimization problem. Here we measure the sensitivity of the change of the velocity in the flow direction with respect to C_{T_j} by $\frac{\partial u_x}{\partial C_{T_j}}$. We plot the magnitude of $\frac{\partial u_x}{\partial C_{T_1}}$ and $\frac{\partial u_x}{\partial C_{T_2}}$ at the local optimum of C_{T_1} and C_{T_2} in Figure 5.7.



(a) Total power (b) 2-norm of the gradient

Figure 5.6: Total power and 2-norm of the gradient



(a) $\frac{\partial u_x}{\partial C_{T_1}}$ (b) $\frac{\partial u_x}{\partial C_{T_2}}$

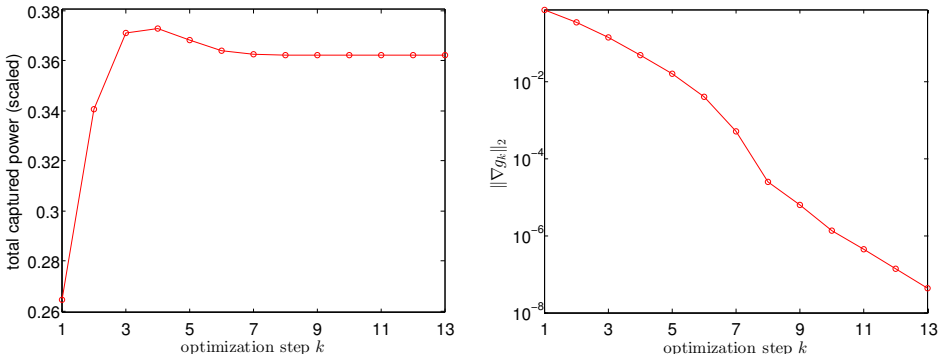
Figure 5.7: $\frac{\partial u_x}{\partial C_{T_1}}$ and $\frac{\partial u_x}{\partial C_{T_2}}$ in magnitude

Figure 5.7 show that there is a big gradient in the vicinity of the places where the turbines are located. This in turn tells us that the velocity in the vicinity of the turbines is very sensitive to the changes of C_{T_j}. This makes this optimization problem very sensitive.

Another reason may be the robustness and the efficiency of the optimization method around the optimum. Since we focus on preconditioning, we leave this for the

discussions and recommendations in the next chapter.

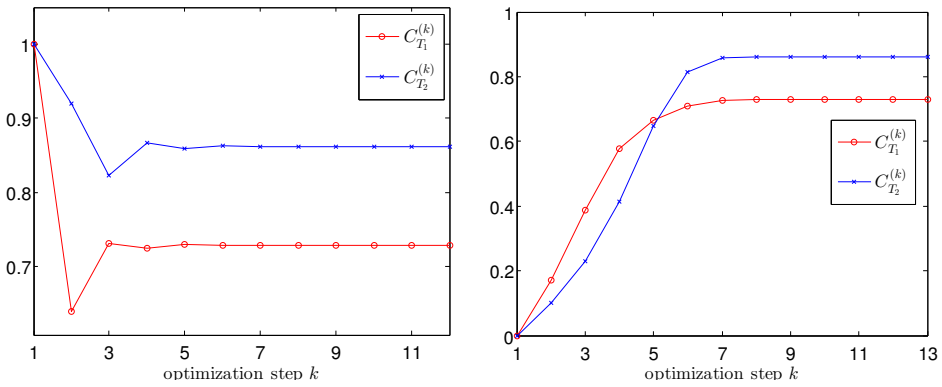
We also solve the optimization problem with an initial guess $C_{T_1}^{(0)} = C_{T_2}^{(0)} = 0$, although this corresponds to an impractical operation status. The scaled total power and the gradient at each optimization step are given in Figure 5.8.



(a) Total power (b) 2-norm of the gradient

Figure 5.8: Total power and 2-norm of the gradient

For those two cases with different initial guesses, the corresponding optimized variables C_{T_1} and C_{T_2} at each optimization step are given in Figure 5.9.



(a) $C_{T_1}^{(0)} = C_{T_2}^{(0)} = 1$ (b) $C_{T_1}^{(0)} = C_{T_2}^{(0)} = 0$

Figure 5.9: Optimized C_{T_1}, C_{T_2} with different initial guesses

Figure 5.9(a) and 5.9(b) show that with different initial guesses, the optimized variables (C_{T_1}, C_{T_2}) converge to the same point (0.729, 0.862), which corresponds to a local optimum of the optimization problem.

5.5 Conclusions

In this chapter, we formulate an optimal in-domain control of PDEs problem by using a simplified wind farm example. The control input only acts on part of the domain, which results a problem even more difficult to solve. The optimization problem is solved by a reduced Newton's method while the most time consuming part for this problem is solving a nonlinear flow equation and computations of the gradient and the Hessian matrix. We show that the gradient and the Hessian matrix can be computed by solving an adjoint equation such that the information from the solution of the flow equation can be reused. This in turn reduces the computational complexity dramatically. We evaluate the performance of the MSSS preconditioning technique for solving the flow equation by using IFISS. Numerical results show the superiority of the MSSS preconditioning technique over the standard preconditioning techniques.

Since we focus on preconditioning techniques for the optimal in-domain control problems, we use a simplified wind farm model in IFISS to study the performance of the MSSS preconditioning technique. The next step to extend this research shall focus on applying the turbulent flow model for the real-world wind farm control applications.

6 Chapter

Conclusions and Recommendations

6.1 Conclusions

This dissertation focused on developing efficient preconditioning techniques for optimal flow control problems using multilevel sequentially semiseparable (MSSS) matrix computations. Optimal flow control problems give a generalized saddle-point system to be solved. MSSS preconditioners compute an approximate factorization of the global system matrix up to a prescribed accuracy in linear computational complexity. This in turn avoids approximation of the Schur complement of the block linear system matrix, while efficient approximation of the Schur complement is the key and also a big challenge to block preconditioners. This is a big advantage over standard block preconditioners.

In Chapter 2, we studied the PDE-constrained optimization problem and develop a novel preconditioner for the saddle-point system arising from such problem using MSSS matrix computations. This novel preconditioner exploits the global MSSS structure of the saddle-point matrix and computes a global factorization of the saddle-point matrix using MSSS matrix computations in linear computational complexity. We also studied the block MSSS preconditioners for such problems. Model order reduction plays a key role in keeping the computational complexity low for such preconditioning techniques. We proposed a new model order reduction algorithm for 1-level MSSS matrix computations, which is computationally cheaper than standard algorithms. Numerical results illustrate that the global MSSS preconditioner gives mesh size and regularization parameter independent convergence.

We extended MSSS preconditioning techniques to computational fluid dynamics (CFD) problems in Chapter 3 and evaluated their performance using IFISS. For convection-diffusion problems, the MSSS preconditioning technique is much more robust than both the algebraic multigrid (AMG) method and geometric multigrid (GMG) method and considerably faster than the AMG method. Block preconditioners computed by the GMG method outperform for Stokes problems and global MSSS preconditioner is competitive with block preconditioners computed by the AMG method. All these methods give mesh size independent convergence. For linearized Navier-Stokes equation, global MSSS preconditioner gives mesh size and

Reynolds number independent convergence while block preconditioner computed by the AMG method only gives mesh size independent convergence. Meanwhile, global MSSS preconditioning technique is relative faster than the block preconditioner computed by the AMG method.

To give insights into the performance of MSSS preconditioning techniques, we analyzed the convergence property of MSSS preconditioners in Chapter 4. Our analysis showed that the preconditioned spectrum is contained in a circle centered at $(1, 0)$ and the radius of the circle that contains the preconditioned spectrum can be controlled arbitrarily by a proper chosen parameter in computing the MSSS preconditioner. The analysis in this chapter applies to general linear systems that can be either symmetric or unsymmetric, definite or indefinite. Related convergence analysis only suit symmetric positive definite linear systems from the discretization of scalar PDEs while our analysis also applies to block linear systems from the discretization of coupled PDEs. We also studied the MSSS preconditioner for the Helmholtz equation. Numerical results illustrate that the MSSS preconditioner is comparable with state-of-the-art methods for such problems.

In Chapter 5, we studied the problem of optimal in-domain control of the Navier-Stokes equation and used a simplified model of wind farm control to formulate an optimal in-domain control problem. We applied reduced Newton's method according to the implicit function theorem to solve such optimization problem. This in turn yields a series of linear systems of the generalized saddle-point type to be solved. Standard block preconditioners did not give satisfactory performance for such generalized saddle-point system because the Schur complement is even more difficult or even impossible to approximate. We applied global MSSS preconditioner to solve such systems and compared its performance with the standard block preconditioner. Numerical results state the superior performance of MSSS preconditioning techniques again.

6.2 Recommendations and Open Questions

- Model order reduction for higher level MSSS matrices.

  As introduced in Chapter 2, the model order reduction for 2-level MSSS matrices is still an open problem. The biggest challenge is that the model order reduction should preserve the lower level MSSS structure. Standard model order reduction algorithms for 1-level MSSS matrices rely on the singular value decomposition (SVD) or rank revealing QR (RRQR) factorization of the Hankel blocks. Extending standard model order reduction algorithms to higher level fails because neither SVD nor RRQR preserves the lower level MSSS structure. Structure preserving model order reduction algorithms should be developed. The model order reduction algorithm for higher level MSSS matrix computations plays a vital role in distributed control and identification of multidimensional dynamical systems, cf. [108] for further reference.

- MSSS preconditioners for PDEs discretized using topologically unstructured grids

The MSSS preconditioning technique is efficient and robust for linear system from discretization of PDEs on topologically structured grid. For PDEs with a complicated domain, unstructured grids are necessary. As introduced in Chapter 3, it is not direct to infer the MSSS structure from an unstructured gird while it is natural to infer an HSS structure from an unstructured grid. Moreover, as stated in [115], the HSS structure and 1-level MSSS structured can be transferred from one to the other. This makes it possible for MSSS matrices to represent discretized PDEs on unstructured grids. Extending MSSS preconditioning techniques to discretized PDEs with complicated geometries is of great practical importance.

- Algebraic preconditioning techniques using structured matrix computations

  To compute an MSSS preconditioner, one needs the topology of the grid that is used to discretize the PDEs. This prohibits the MSSS preconditioner to be used as a purely algebraic method. No grid information is needed for HSS matrix computations and $\mathcal{H}$-matrix computations. Therefore, $\mathcal{H}$-matrix and HSS matrix computations can be used to compute an algebraic preconditioner. However, applying such algebraic methods to block linear systems is neither natural nor direct. This is because these algebraic methods require to permute block matrices with low rank structured blocks into global matrices with low rank structure.

- Wind farm control

  We use a simplified model of the wind farm control to formulate the problem of optimal in-domain control of PDEs using IFISS in Chapter 5. Since no turbulent flow model is contained in IFISS, we use a laminar flow model for the wind farm control problem, which is not realistic. Future research could focus on applying the MSSS preconditioning technique to the simulation of turbulent flow in the wind farm. Moreover, the reduced Newton's method that is studied in Chapter 5 needs to compute an $N_t^2 \times N_t^2$ Hessian matrix, where N_t is the number of wind turbines in a wind farm. If the number of turbines in a wind farm is big, this optimization method is not computationally feasible. Numerical optimization algorithms of the line search type or interior point type could be employed.

- Preconditioning time-dependent PDEs using structured matrix computations

  All the problems under consideration in this dissertation are stationary. Some recent development for time-dependent PDEs that applies low-rank approximation approaches in [121] can be combined with MSSS preconditioning techniques to develop numerical algorithms to compute the solution in time and space all at once.

A Appendix

MSSS Lower-Triangular Preconditioner for Optimal Convection-Diffusion Control

This part gives the performance of the block lower-triangular preconditioner for optimal control of the convection-diffusion equation in Example 2.2. Take the block lower-triangular preconditioner $\mathcal{P}_2$ in (2.5) by the approximate balanced truncation Algorithm 2.2 - 2.3 and the Hankel blocks approximation Algorithm 2.4, solve the unsymmetric preconditioned system with IDR(s) method. The computational results are shown in Table A.1 - A.10.

Table A.1: By approximate balanced truncation for $\beta = 10^{-1}$, $\nu = 10^{-1}$

problem size	iterations	preconditioning	IDR(16)	total
3.07e+03 (3)	12	0.34	1.09	1.43
1.23e+04 (6)	12	0.99	2.61	3.60
4.92e+04 (6)	11	4.07	7.02	11.09
1.97e+05 (10)	12	18.05	24.09	42.14

Table A.2: By Hankel blocks approximation for $\beta = 10^{-1}$, $\nu = 10^{-1}$

problem size	iterations	preconditioning	IDR(16)	total
3.07e+03 (3)	13	0.56	1.29	1.85
1.23e+04 (6)	9	1.77	2.01	3.78
4.92e+04 (6)	16	9.02	9.89	18.91
1.97e+05 (10)	10	28.28	19.76	48.04

Table A.3: By approximate balanced truncation for $\beta = 10^{-1}$, $\nu = 10^{-2}$

problem size	iterations	preconditioning	IDR(32)	total
3.07e+03 (3)	15	0.26	1.20	1.46
1.23e+04 (3)	13	0.70	2.74	3.14
4.92e+04 (4)	13	2.43	7.76	10.19
1.97e+05 (10)	13	25.06	30.67	55.73

Table A.4: By Hankel blocks approximation for $\beta = 10^{-1}$, $\nu = 10^{-2}$

problem size	iterations	preconditioning	IDR(32)	total
3.07e+03 (3)	15	0.45	1.23	1.68
1.23e+04 (3)	17	1.29	3.39	4.68
4.92e+04 (4)	17	4.77	9.97	14.74
1.97e+05 (10)	14	48.20	32.40	80.60

Table A.5: By approximate balanced truncation for $\beta = 10^{-1}$, $\nu = 10^{-2}$

problem size	iterations	preconditioning	IDR(16)	total
3.07e+03 (3)	18	0.37	1.51	1.43
1.23e+04 (3)	16	0.68	3.17	3.85
4.92e+04 (4)	15	2.38	7.95	10.33
1.97e+05 (8)	18	13.61	35.46	49.07

Table A.6: By Hankel blocks approximation for $\beta = 10^{-1}$, $\nu = 10^{-2}$

problem size	iterations	preconditioning	IDR(16)	total
3.07e+03 (4)	20	0.51	1.62	2.13
1.23e+04 (3)	27	1.24	5.44	6.68
4.92e+04 (4)	16	4.77	8.19	12.96
1.97e+05 (8)	19	24.70	36.75	59.45

Table A.7: By approximate balanced truncation for $\beta = 10^{-2}$, $\nu = 10^{-1}$

problem size	iterations	preconditioning	IDR(32)	total
3.07e+03 (6)	16	0.42	1.41	1.83
1.23e+04 (6)	17	1.17	3.65	4.82
4.92e+04 (7)	19	4.41	11.80	16.21
1.97e+05 (10)	18	25.33	41.86	67.19

Table A.8: By Hankel blocks approximation for $\beta = 10^{-2}$, $\nu = 10^{-1}$

problem size	iterations	preconditioning	IDR(32)	total
3.07e+03 (6)	17	0.66	1.49	2.15
1.23e+04 (6)	19	2.22	4.03	6.25
4.92e+04 (7)	21	9.81	12.81	22.62
1.97e+05 (10)	16	49.78	36.75	86.53

Table A.9: By approximate balanced truncation for $\beta = 10^{-2}$, $\nu = 10^{-2}$

problem size	iterations	preconditioning	IDR(32)	total
3.07e+03 (6)	30	0.39	2.65	3.04
1.23e+04 (6)	32	1.12	6.85	7.97
4.92e+04 (7)	32	4.32	20.65	24.97
1.97e+05 (10)	31	25.08	71.03	96.11

Table A.10: By Hankel blocks approximation for $\beta = 10^{-2}$, $\nu = 10^{-2}$

problem size	iterations	preconditioning	IDR(32)	total
3.07e+03 (6)	30	0.68	2.59	3.27
1.23e+04 (6)	36	2.37	7.75	10.12
4.92e+04 (7)	31	9.55	19.55	39.10
1.97e+05 (10)	32	48.78	72.58	121.36

Bibliography

[1] Gassan S. Abdoulaev, Kui Ren, and Andreas H. Hielscher, *Optical tomography as a PDE-constrained optimization problem*, Inverse Problems, 21 (2005), pp. 1507–1530.

[2] Kuniyoshi Abe and Gerard L.G. Sleijpen, *Hybrid Bi-CG methods with a Bi-CG formulation closer to the IDR approach*, Applied Mathematics and Computation, 218 (2012), pp. 10889 – 10899.

[3] Mahdi Abkar and Fernando Porté-Agel, *The effect of free-atmosphere stratification on boundary-layer flow and power output from very large wind farms*, Energies, 6 (2013), pp. 2338–2361.

[4] Santi S. Adavani and George Biros, *Multigrid algorithms for inverse problems with linear parabolic PDE constraints*, SIAM Journal on Scientific Computing, 31 (2008), pp. 369–397.

[5] Jennifer Annoni, Patrick Seiler, Kenneth Johnson, Paul Fleming, and Pieter Gebraad, *Evaluating wake models for wind farm control*, in Proceedings of American Control Conference, 2014, pp. 2517–2523.

[6] Athanasios C. Antoulas, Danny C. Sorensen, and Yunkai Zhou, *On the decay rate of Hankel singular values and related issues*, Systems & Control Letters, 46 (2002), pp. 323–342.

[7] Jasbir S. Arora and Qian Wang, *Review of formulations for structural and mechanical system optimization*, Structural and Multidisciplinary Optimization, 30 (2005), pp. 251–272.

[8] Kerstin Avila, David Moxey, Alberto de Lozar, Marc Avila, Dwight Barkley, and Björn Hof, *The onset of turbulence in pipe flow*, Science, 333 (2011), pp. 192–196.

[9] Hassan Badreddine, Stefan Vandewalle, and Johan Meyers, *Sequential quadratic programming (SQP) for optimal control in direct numerical simulation of turbulent flow*, Journal of Computational Physics, 256 (2014), pp. 1–16.

[10] Eberhard Bänsch, Peter Benner, Jens Saak, and Heiko K. Weichelt, *Riccati-based boundary feedback stabilization of incompressible Navier-Stokes flows*, SIAM Journal on Scientific Computing, 37 (2015), pp. A832–A858.

[11] Hakan Bağci, Joseph E. Pasciak, and Kostyantyn Y. Sirenko, *A convergence analysis for a sweeping preconditioner for block tridiagonal systems of linear equations*, Numerical Linear Algebra with Applications, 22 (2015), pp. 371–392.

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

[13] Mario Bebendorf, *Hierarchical matrices*, in Lecture Notes in Computational Science and Engineering, vol. 63, Springer Berlin-Heidelberg, 2008.

[14] Peter Benner, *Solving large-scale control problems*, IEEE Control Systems Magazine, 24 (2004), pp. 44–59.

[15] Michele Benzi, Gene H. Golub, and Jörg Liesen, *Numerical solution of saddle point problems*, Acta Numerica, 14 (2005), pp. 1–137.

[16] Michele Benzi and Maxim Olshanskii, *An augmented Lagrangian-based approach to the Oseen problem*, SIAM Journal on Scientific Computing, 28 (2006), pp. 2095–2113.

[17] Michele Benzi, Maxim A. Olshanskii, and Zhen Wang, *Modified augmented Lagrangian preconditioners for the incompressible Navier-Stokes equations*, International Journal for Numerical Methods in Fluids, 66 (2011), pp. 486–508.

[18] Martin Berggren, *Numerical solution of a flow-control problem: vorticity reduction by dynamic boundary action*, SIAM Journal on Scientific Computing, 19 (1998), pp. 829–860.

[19] Dennis S. Bernstein, *Matrix Mathematics: Theory, Facts, and Formulas*, Princeton University Press, New Jersey, second ed., 2009.

[20] Lorenz T. Biegler, Omar Ghattas, Matthias Heinkenschloss, David Keyes, and Bart van Bloemen Waanders, *Real-time PDE-constrained Optimization*, vol. 3, Society for Industrial and Applied Mathematics, Philadelphia, 2007.

[21] Lorenz T. Biegler, Omar Ghattas, Matthias Heinkenschloss, and Bart van Bloemen Waanders, *Large-Scale PDE-Constrained Optimization*, vol. 30, Springer Science & Business Media, 2012.

[22] George Biros and Omar Ghattas, *Parallel Lagrange-Newton-Krylov-Schur methods for PDE-constrained optimization. Part I: the Krylov-Schur solver*, SIAM Journal on Scientific Computing, 27 (2005), pp. 687–713.

[23] ——, *Parallel Lagrange-Newton-Krylov-Schur methods for PDE-constrained optimization. Part II: the Lagrange-Newton solver and its application to optimal control of steady viscous flows*, SIAM Journal on Scientific Computing, 27 (2005), pp. 714–739.

[24] 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 (2012), pp. 83–98.

[25] Alfio Borzì and Volker Schulz, *Multigrid methods for PDE optimization*, SIAM Review, 51 (2009), pp. 361–395.

[26] Martin Burger and Wolfram Mühlhuber, *Iterative regularization of parameter identification problems by sequential quadratic programming methods*, Inverse Problems, 18 (2002), p. 943.

[27] Tony Burton, David Sharpe, Nick Jenkins, and Ervin Bossanyi, *Wind Energy Handbook*, John Wiley & Sons, West Sussex, 2001.

[28] Younes Chahlaoui, *Low-rank Approximation and Model Reduction*, PhD thesis, Université catholique de Louvain, December 2003.

[29] ——, *Two efficient SVD/Krylov algorithms for model order reduction of large scale systems*, Electronic Transactions on Numerical Analysis, 38 (2011), pp. 113–145.

[30] Younes Chahlaoui and Paul Van Dooren, *Estimating gramians of large-scale time-varying systems*, in IFAC World Congress, vol. 15, 2002, pp. 540–545.

[31] ——, *Model reduction of time-varying systems*, in Dimension Reduction of Large-Scale Systems, Peter Benner, Danny C. Sorensen, and Volker Mehrmann, eds., vol. 45 of Lecture Notes in Computational Science and Engineering, Springer Berlin-Heidelberg, 2005, pp. 131–148.

[32] Shivkumar Chandrasekaran, Patrick Dewilde, Ming Gu, William Lyons, and Timothy Pals, *A fast solver for HSS representations via sparse matrices*, SIAM Journal on Matrix Analysis and Applications, 29 (2006), pp. 67–81.

[33] Shivkumar Chandrasekaran, Patrick Dewilde, Ming Gu, Timothy P. Pals, Xiaorui Sun, Alle-Jan van der Veen, and Daniel White, *Some fast algorithms for sequentially semiseparable representations*, SIAM Journal on Matrix Analysis and Applications, 27 (2005), pp. 341–364.

[34] Shivkumar Chandrasekaran, Patrick Dewilde, Ming Gu, Timothy P. Pals, and Alle-Jan van der Veen, *Fast stable solvers for sequentially semi-separable linear systems of equations*, tech. report, Lawrence Livermore National Laboratory, 2003.

[35] Shivkumar Chandrasekaran, Patrick Dewilde, Ming Gu, and Naveen 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 (2010), pp. 2261–2290.

[36] Souma Chowdhury, Jie Zhang, Achille Messac, and Luciano Castillo, *Unrestricted wind farm layout optimization (UWFLO): Investigating key factors influencing the maximum power generation*, Renewable Energy, 38 (2012), pp. 16–30.

[37] Paul Concus, Gene Golub, and Gérard Meurant, *Block preconditioning for the conjugate gradient method*, SIAM Journal on Scientific and Statistical Computing, 6 (1985), pp. 220–252.

[38] Antonio Crespo, Francisco J. R. Hernández, and Sten Frandsen, *Survey of modelling methods for wind turbine wakes and wind farms*, Wind energy, 2 (1999), pp. 1–24.

[39] Sara Delport, Martine Baelmans, and Johan Meyers, *Constrained optimization of turbulent mixing-layer evolution*, Journal of Turbulence, 10 (2009), pp. 1–26.

[40] Patrick Dewilde, Haiyan Jiao, and Shivkumar 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, vol. 197 of Operator Theory: Advances and Applications, Birkhäser Basel, 2010, pp. 99–132.

[41] Patrick Dewilde and Alle-Jan van der Veen, *Time-Varying Systems and Computations*, Kluwer Academic Publishers, Boston, 1998.

[42] Valérie Dos Santos and Christophe Prieur, *Boundary control of open channels with numerical and experimental validations*, IEEE Transactions on Control Systems Technology, 16 (2008), pp. 1252–1264.

[43] Lei Du, Tomohiro Sogabe, Bo Yu, Yusaku Yamamoto, and Shaoliang Zhang, *A block IDR(s) method for nonsymmetric linear systems with multiple right-hand sides*, Journal of Computational and Applied Mathematics, 235 (2011), pp. 4095 – 4106.

[44] Yuli Eidelman and Israel Gohberg, *On a new class of structured matrices*, Integral Equations and Operator Theory, 34 (1999), pp. 293–324.

[45] ——, *A modification of the Dewilde-van der Veen method for inversion of finite structured matrices*, Linear Algebra and Its Applications, 343-344 (2002), pp. 419–450.

[46] ——, *On generators of quasiseparable finite block matrices*, Calcolo, 42 (2005), pp. 187–214.

[47] Yuli Eidelman, Israel Gohberg, and Vadim Olshevsky, *The QR iteration method for Hermitian quasiseparable matrices of an arbitrary order*, Linear Algebra and Its Applications, 404 (2005), pp. 305–324.

[48] Howard Elman, Victoria E. Howle, John Shadid, Robert Shuttleworth, and Ray Tuminaro, *Block preconditioners based on approximate commutators*, SIAM Journal on Scientific Computing, 27 (2006), pp. 1651–1668.

[49] Howard Elman, Victoria E. Howle, John Shadid, David Silvester, and Ray Tuminaro, *Least squares preconditioners for stabilized discretizations of the Navier-Stokes equations*, SIAM Journal on Scientific Computing, 30 (2008), pp. 290–311.

[50] Howard Elman and David Silvester, *Fast nonsymmetric iterations and preconditioning for Navier-Stokes equations*, SIAM Journal on Scientific Computing, 17 (1996), pp. 33–46.

[51] Howard C. Elman, *Preconditioning for the steady-state Navier-Stokes equations with low viscosity*, SIAM Journal on Scientific Computing, 20 (1999), pp. 1299–1316.

[52] Howard C. Elman and Gene H. Golub, *Inexact and preconditioned Uzawa algorithms for saddle point problems*, SIAM Journal on Numerical Analysis, 31 (1994), pp. 1645–1661.

[53] Howard C. Elman, David J. Silvester, and Andrew J. Wathen, *Finite Elements and Fast Iterative Solvers: with Applications in Incompressible Fluid Dynamics*, Oxford University Press, New York, 2005.

[54] Björn Engquist and Lexing Ying, *Sweeping preconditioner for the Helmholtz equation: hierarchical matrix representation*, Communications on pure and applied mathematics, 64 (2011), pp. 697–735.

[55] Yogi A. Erlangga, Cornelis W. Oosterlee, and Cornelis Vuik, *A novel multigrid based preconditioner for heterogeneous Helmholtz problems*, SIAM Journal on Scientific Computing, 27 (2006), pp. 1471–1492.

[56] Andrei V. Fursikov, Max D. Gunzburger, and L. Steven Hou, *Boundary value problems and optimal boundary control for the Navier-Stokes system: the two-dimensional case*, SIAM Journal on Control and Optimization, 36 (1998), pp. 852–894.

[57] Pieter M. O. Gebraad, *Data-Driven Wind Plant Control*, PhD thesis, Delft University of Technology, 2014.

[58] Philip E. Gill, Walter Murray, Dulce B. Ponceleón, and Michael A. Saunders, *Preconditioners for indefinite systems arising in optimization*, SIAM Journal on Matrix Analysis and Applications, 13 (1992), pp. 292–311.

[59] Jay P. Goit and Johan Meyers, *Optimal control of energy extraction in wind-farm boundary layers*, Journal of Fluid Mechanics, 768 (2015), pp. 5–50.

[60] Gene H. Golub and Charles F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, third ed., 1996.

[61] Jacek Gondzio and Pavel Zhlobich, *Multilevel quasiseparable matrices in PDE-constrained optimization*, arXiv preprint arXiv:1112.6018, (2011).

[62] Reo A. Gonzales, Jens Eisert, Israel Koltracht, Michael Neumann, and George H. Rawitscher, *Integral equation method for the continuous spectrum radial Schrodinger equation*, Journal of Computational Physics, 134 (1997), pp. 134–149.

[63] Leslie F. Greengard and Vladimir Rokhlin, *On the numerical solution of two-point boundary value problems*, Communications on Pure and Applied Mathematics, 44 (1991), pp. 419–452.

[64] Ming Gu, Xiaoye S. Li, and Panayot S. Vassilevski, *Direction-preserving and Schur-monotonic semiseparable approximations of symmetric positive definite matrices*, SIAM Journal on Matrix Analysis and Applications, 31 (2010), pp. 2650–2664.

[65] Eldad Haber, Uri M. Ascher, and Doug Oldenburg, *On optimization techniques for solving nonlinear inverse problems*, Inverse problems, 16 (2000), p. 1263.

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

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

[68] Martin O. L. Hansen, *Aerodynamics of Wind Turbines*, Routledge, New York, 2015.

[69] Subhendu B. Hazra, Volker H. Schulz, Joël Brezillon, and Nicolas R. Gauger, *Aerodynamic shape optimization using simultaneous pseudo-timestepping*, Journal of Computational Physics, 204 (2005), pp. 46 – 64.

[70] Lars Henning, Dmitri Kuzmin, Volker Mehrmann, Michael Schmidt, Andriy Sokolov, and Stefan Turek, *Flow control on the basis of a Featflow-Matlab coupling*, in Active Flow Control, Rudibert King, ed., vol. 95 of Notes on Numerical Fluid Mechanics and Multidisciplinary Design (NNFM), Springer Berlin-Heidelberg, 2007, pp. 325–338.

[71] Michael Hinze and Karl Kunisch, *Second order methods for optimal control of time-dependent fluid flow*, SIAM Journal on Control and Optimization, 40 (2001), pp. 925–946.

[72] Ilse CF Ipsen, *A note on preconditioning nonsymmetric matrices*, SIAM Journal on Scientific Computing, 23 (2001), pp. 1050–1051.

[73] Alcksandar Kavčić and José M.F. Moura, *Matrices with banded inverses: inversion algorithms and factorization of Gauss-Markov processes*, IEEE Transactions on Information Theory, 46 (2000), pp. 1495–1509.

[74] David Kay, Daniel Loghin, and Andrew Wathen, *A preconditioner for the steady-state Navier-Stokes equations*, SIAM Journal on Scientific Computing, 24 (2002), pp. 237–256.

[75] Miroslav Krstic and Andrey Smyshlyaev, *Boundary Control of PDEs: a Course on Backstepping Designs*, Society for Industrial and Applied Mathematics, Philadelphia, 2008.

[76] Emrah Kulunk, *Aerodynamics of Wind Turbines*, INTECH Open Access Publisher, 2011.

[77] Manfred Laumen, *Newton's method for a class of optimal shape design problems*, SIAM Journal on Optimization, 10 (2000), pp. 503–533.

[78] Sabine Le Borne and Lars Grasedyck, *$\mathcal{H}$-matrix preconditioners in convection-dominated problems*, SIAM Journal on Matrix Analysis and Applications, 27 (2006), pp. 1172–1183.

[79] Daniel B. Leineweber, Irene Bauer, Hans Georg Bock, and Johannes P. Schlöder, *An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization. Part I: theoretical aspects*, Computers & Chemical Engineering, 27 (2003), pp. 157 – 166.

[80] Chunguang Li and Cornelis Vuik, *Eigenvalue analysis of the SIMPLE preconditioning for incompressible flow*, Numerical Linear Algebra with Applications, 11 (2004), pp. 511–523.

[81] Jacques Louis Lions, *Optimal Control of Systems Governed by Partial Differential Equations*, vol. 170, Springer Verlag, 1971.

[82] Jason R. Marden, Shalom D. Ruben, and Lucy Y. Pao, *A model-free approach to wind farm control using game theoretic methods*, IEEE Transactions on Control Systems Technology, 21 (2013), pp. 1207–1214.

[83] Gérard Meurant, *A review on the inverse of symmetric tridiagonal and block tridiagonal matrices*, SIAM Journal on Matrix Analysis and Applications, 13 (1992), pp. 707–728.

[84] Johan Meyers and Charles Meneveau, *Optimal turbine spacing in fully developed wind farm boundary layers*, Wind Energy, 15 (2012), pp. 305–317.

[85] Robert Mikkelsen, *Actuator Disc Methods Applied to Wind Turbines*, PhD thesis, Technical University of Denmark, 2003.

[86] Malcolm F. Murphy, Gene H. Golub, and Andrew J. Wathen, *A note on preconditioning for indefinite linear systems*, SIAM Journal on Scientific Computing, 21 (2000), pp. 1969–1972.

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

[88] Jorge Nocedal and Stephen Wright, *Numerical Optimization*, Springer Science & Business Media, New York, 2006.

[89] Yvan Notay, *A new analysis of block preconditioners for saddle point problems*, SIAM Journal on Matrix Analysis and Applications, 35 (2014), pp. 143–173.

[90] Maxim A. Olshanskii and Yuri Vassilevski, *Pressure Schur complement preconditioners for the discrete Oseen problem*, SIAM Journal on Scientific Computing, 29 (2007), pp. 2686–2704.

[91] Cornelis W. Oosterlee, Cornelis Vuik, Wim A. Mulder, and Rene-Edouard Plessix, *Shifted-Laplacian preconditioners for heterogeneous Helmholtz problems*, in Advanced Computational Methods in Science and Engineering, Barry Koren and Kees Vuik, eds., vol. 71 of Lecture Notes in Computational Science and Engineering, Springer-Verlag Berlin-Heidelberg, 2010, pp. 21–46.

[92] Chris Paige and Michael Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM Journal on Numerical Analysis, 12 (1975), pp. 617–629.

[93] John W. Pearson, *On the development of parameter-robust preconditioners and commutator arguments for solving Stokes control problems*, Electronic Transactions on Numerical Analysis, 44 (2015), pp. 53–72.

[94] ——, *Preconditioned iterative methods for Navier-Stokes control problems*, Journal of Computational Physics, 292 (2015), pp. 194 – 207.

[95] John W. Pearson and Martin Stoll, *Fast iterative solution of reaction-diffusion control problems arising from chemical processes*, SIAM Journal on Scientific Computing, 35 (2013), pp. B987–B1009.

[96] John W. Pearson and Andrew J. Wathen, *A new approximation of the Schur complement in preconditioners for PDE-constrained optimization*, Numerical Linear Algebra with Applications, 19 (2012), pp. 816–829.

[97] ——, *Fast iterative solvers for convection-diffusion control problems*, Electronic Transactions on Numerical Analysis, 40 (2013), pp. 294–310.

[98] Jennifer Pestana and Andrew J. Wathen, *Natural preconditioning and iterative methods for saddle point systems*, SIAM Review, 57 (2015), pp. 71–91.

[99] Andreas Potschka, Mario S. Mommer, Johannes P. Schlöder, and Hans Georg Bock, *Newton-Picard-based preconditioning for linear-quadratic optimization problems with time-periodic parabolic PDE constraints*, SIAM Journal on Scientific Computing, 34 (2012), pp. A1214–A1239.

[100] Yue Qiu, Martin B. van Gijzen, Jan-Willem van Wingerden, and Michel Verhaegen, *A class of efficient preconditioners with multilevel sequentially semiseparable matrix structure*, AIP Conference Proceedings, 1558 (2013), pp. 2253–2256.

[101] ——, *On the application of a novel model order reduction algorithm for sequentially semi-separable matrices to the identification of one-dimensional distributed systems*, in Proceedings of the 13th European Control Conference, June 2014, pp. 2750–2755.

[102] Yue Qiu, Martin B. van Gijzen, Jan-Willem van Wingerden, Michel Verhaegen, and Cornelis Vuik, *Efficient preconditioners for PDE-constrained optimization problems with a multilevel sequentially semiseparable matrix structure*, Tech. Report 13-04, Delft Institution of Applied Mathematics, Delft University of Technology, 2013. Available at `http://ta.twi.tudelft.nl/nw/users/yueqiu/publications.html`.

[103] ——, *Efficient preconditioners for PDE-constrained optimization problems with a multilevel sequentially semiseparable matrix structure*, Electronic Transactions on Numerical Analysis, 44 (2015), pp. 367–400.

[104] ——, *Evaluation of multilevel sequentially semiseparable preconditioners on CFD benchmark problems using incompressible flow and iterative solver software*, Mathematical Methods in the Applied Sciences, 38 (2015), pp. n/a–n/a.

[105] Ole Rathmann, Sten T. Frandsen, and Rebecca J. Barthelmie, *Wake modelling for intermediate and large wind farms*, in 2007 European Wind Energy Conference and Exhibition.

[106] Tyrone Rees, *Preconditioning Iterative Methods for PDE-Constrained Optimization*, PhD thesis, University of Oxford, 2010.

[107] Tyrone Rees and Andrew J. Wathen, *Preconditioning iterative methods for the optimal control of the Stokes equations*, SIAM Journal on Scientific Computing, 33 (2011), pp. 2903–2926.

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

[109] Justin K. Rice and Michel Verhaegen, *Distributed control: a sequentially semi-separable approach for spatially heterogeneous linear systems*, IEEE Transactions on Automatic Control, 54 (2009), pp. 1270–1283.

[110] Yousef Saad, *Iterative Methods for Sparse Linear Systems*, Society for Industrial and Applied Mathematics, Philadelphia, 2003.

[111] Henrik Sandberg and Anders Rantzer, *Balanced truncation of linear time-varying systems*, IEEE Transactions on Automatic Control, 49 (2004), pp. 217–229.

[112] Benjamin Sanderse, Sander P. van der Pijl, and Barry Koren, *Review of computational fluid dynamics for wind turbine wake aerodynamics*, Wind Energy, 14 (2011), pp. 799–819.

[113] Joachim Schöberl and Walter Zulehner, *Symmetric indefinite preconditioners for saddle point problems with applications to PDE-constrained optimization problems*, SIAM Journal on Matrix Analysis and Applications, 29 (2007), pp. 752–773.

[114] Abdul H. Sheikh, Domenico Lahaye, and Cornelis Vuik, *On the convergence of shifted Laplace preconditioner combined with multilevel deflation*, Numerical Linear Algebra with Applications, 20 (2013), pp. 645–662.

[115] Zhifeng Sheng, Patrick Dewilde, and Shivkumar Chandrasekaran, *Algorithms to solve hierarchically semi-separable systems*, in Operator Theory: Advances and Applications, Daniel Alpay and Victor Vinnikov, eds., vol. 176, Birkhäuser Basel, 2007, pp. 255–294.

[116] David Silvester, Howard Elman, David Kay, and Andrew Wathen, *Efficient preconditioning of the linearized Navier-Stokes equations for incompressible flow*, Journal of Computational and Applied Mathematics, 128 (2001), pp. 261–279.

[117] David J. Silvester, Howard C. Elman, and Alison Ramage, *Incompressible Flow and Iterative Solver Software (IFISS) version 3.2*, May 2012. `http://www.manchester.ac.uk/ifiss/`.

[118] David J. Silvester and Andrew Wathen, *Fast iterative solution of stabilised Stokes systems. Part II: using general block preconditioners*, SIAM Journal on Numerical Analysis, 31 (1994), pp. 1352–1367.

[119] Peter Sonneveld and Martin 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 (2008), pp. 1035–1062.

[120] Gilbert W. Stewart, Jiguang Sun, and Harcourt B. Jovanovich, *Matrix Perturbation Theory*, Academic Press, New York, 1990.

[121] Martin Stoll and Tobias Breiten, *A low-rank in time approach to PDE-constrained optimization*, SIAM Journal on Scientific Computing, 37 (2015), pp. B1–B29.

[122] Patricio Torres, Jan-Willem Van Wingerden, and Michel Verhaegen, *Modeling of the flow in wind farms for total power optimization*, in Proceedings of the 9th IEEE International Conference on Control and Automation, 2011, pp. 963–968.

[123] Niels Troldborg, *Actuator Line Modeling of Wind Turbine Wakes*, PhD thesis, Technical University of Denmar, 2008.

[124] Marc Van Barel, Dario Fasino, Luca Gemignani, and Nicola Mastronardi, *Orthogonal rational functions and diagonal-plus-semiseparable matrices*, in International Symposium on Optical Science and Technology, International Society for Optics and Photonics, 2002, pp. 162–170.

[125] Alle-Jan van der Veen, *Time-Varying System Theory and Computational Modeling*, PhD thesis, Delft University of Technology, 1993.

[126] Martin B. van Gijzen, Yogi A. Erlangga, and Cornelis Vuik, *Spectral analysis of the discrete Helmholtz operator preconditioned with a shifted Laplacian*, SIAM Journal on Scientific Computing, 29 (2007), pp. 1942–1958.

[127] 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 (2011), pp. 5:1–5:19.

[128] Raf Vandebril, Marc Van Barel, and Nicola Mastronardi, *Matrix Computations and Semiseparable Matrices: Linear Systems*, Johns Hopkins University Press, Baltimore, 2007.

[129] Dragan Žigić and Layne T. Watson, *Contragredient transformations applied to the optimal projection equations*, Linear Algebra and its Applications, 188-189 (1993), pp. 665–676.

[130] Andrew J. Wathen and David Silvester, *Fast iterative solution of stabilised Stokes systems. Part I: using simple diagonal preconditioners*, SIAM Journal on Numerical Analysis, 30 (1993), pp. 630–649.

[131] Frank M. White and Isla Corfield, *Viscous Fluid Flow*, vol. 3, McGraw-Hill, New York, 2006.

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

[133] ——, *Efficient structured multifrontal factorization for general large sparse matrices*, SIAM Journal on Scientific Computing, 35 (2013), pp. 832–860.

[134] Jianlin Xia, Shivkumar Chandrasekaran, Ming Gu, and Xiaoye Li, *Superfast multifrontal method for large structured linear systems of equations*, SIAM Journal on Matrix Analysis and Applications, 31 (2010), pp. 1382–1411.

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

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

List of Abbreviations

ABT	approximate balanced truncation
ADM	actuator disk method
AL	augmented Lagrangian
AMG	algebraic multigrid
AYC	active yaw control
CFD	computational fluid dynamics
CG	conjugate gradient
FDM	finite difference method
FEM	finite element method
FLOPS	floating point operations
GMG	geometric multigrid
GMRES	generalized minimal residual
HSS	hierarchically semiseparable
IDR	induced dimension reduction
IFISS	incompress flow & iterative solver software
ILU	imcomplete LU factorization
LMI	linear matrix inequality
LTI	linear time-invariant
LTV	linear time-varying
MINRES	minimal residual
MOR	model order reduction
MSSS	multilevel sequentially semiseparable
NLP	nonlinear programming
PCD	pressure convection-diffusion
PDE	partial differential equation

RRQR	rank revealing QR
RSQP	reduced sequential quadratic programming
SIMPLE	semi-implicit method for pressure linked equations
SPD	symmetric positive definite
SQP	sequential quadratic programming
SSS	sequentially semiseparable
SVD	singular value decomposition
1D	one dimensional
2D	two dimensional
3D	three dimensional

List of Academic Activities

纸上得来终觉浅, 绝知此事要躬行

– 宋·陆游

Tell me, and I will remember for a day.
Show me, and I will remember for a week.
Involve me, and I will remember forever.

– Lu Yu (1125 – 1209)

Journal Articles & Preprints

1. Yue Qiu, Martin B. van Gijzen, Jan-Willem van Wingerden, Michel Verhaegen, Cornelis Vuik. *Preconditioning Optimal In-Domain Control of Navier-Stokes Equation Using Multilevel Sequentially Semiseparable Matrix Computations.* Submitted to a journal, available as Technical Report 15-06, Delft Institute of Applied Mathematics.

2. Yue Qiu, Martin B. van Gijzen, Jan-Willem van Wingerden, Michel Verhaegen, Cornelis Vuik. *Convergence Analysis of Multilevel Sequentially Semiseparable Preconditioners.* Submitted to a journal, available as Technical Report 15-01, Delft Institute of Applied Mathematics.

3. Yue Qiu, Martin B. van Gijzen, Jan-Willem van Wingerden, Michel Verhaegen, Cornelis Vuik. *Efficient Preconditioners for PDE-Constrained Optimization Problems with a Multilevel Sequentially Semiseparable Matrix Structure.* **Electronic Transactions on Numerical Analysis**, Vol. 44, 2015.

4. Yue Qiu, Martin B. van Gijzen, Jan-Willem van Wingerden, Michel Verhaegen, Cornelis Vuik. *Evaluation of Multilevel Sequentially Semiseparable Preconditioners on CFD Benchmark Problems Using In-*

compressible Flow and Iterative Solver Software. **Mathematical Methods in the Applied Sciences**, Vol. 38, 2015.

5. Qiuye Sun, Yaguang Wang, Jun Yang, Yue Qiu, Huaguang Zhang. *Chaotic Dynamics in Smart Grid and Suppression Scheme via Generalized Fuzzy Hyperbolic Model.* **Mathematical Problems in Engineering**, Vol. 2014, 2014.

Articles in Peer-Referred Conference Proceedings

6. Yue Qiu, Martin B. van Gijzen, Jan-Willem van Wingerden, Michel Verhaegen. *On the Application of a Novel Model Order Reduction Algorithm for Sequentially Semi-Separable Matrices to the Identification of One-dimensional Distributed Systems.* **Proceedings of the 13th European Control Conference**, 2014.

7. Yue Qiu, Martin B. van Gijzen, Jan-Willem van Wingerden, Michel Verhaegen. *A Class of Efficient Preconditioners with Multilevel Sequentially Semiseparable Matrix Structure.* **AIP Conference Proceedings**, Vol. 1558, 2013.

Contributed Talks at Conferences & Workshops

1. *Multilevel Sequentially Semiseparable Preconditioners for Optimal In-Domain Control of PDEs: A Wind Farm Example.* **GAMM Workshop on Applied and Numerical Linear Algebra**, Magdeburg, Germany, July 9-10, 2015.

2. *Convergence Analysis of Multilevel Sequentially Semiseparable Preconditioners.* **International Conference on Preconditioning Techniques for Scientific and Industrial Applications**, Eindhoven, the Netherlands, June 17-19, 2015.

3. *Preconditioners for Saddle Point Problems with Multilevel Sequentially Semiseparable Matrix Structure.* **Structured Numerical Linear and Multilinear Algebra: Analysis, Algorithms and Applications**, Kalamata, Greece, September 8-12, 2014.

4. *A New Model Reduction Algorithm for Sequentially Semiseparable Matrices and its Applications to System Identification.* **13th European Control Conference (ECC)**, Strassburg, France, June 27-30, 2014.

5. *Efficient Preconditioners with Multilevel Sequentially Semiseparable Structure.* **11th International Conference on Numerical Analysis and Applied Mathematics (ICNAAM)**, Rhodos, Greece, September 21-27, 2013.

6. *Fast Computation Algorithm for Identification and Distributed Control of Large-Scale Systems.* **31th Benelux Meeting on Systems and Control**, Heijden, the Netherlands, March 27-29, 2012.

Poster Presenting

7. *Efficient Preconditioners for PDE-Constrained Optimization Problems with Multilevel Sequentially Semiseparable Structure.* **Recent Advances on Optimization**, Toulouse, France, July 24-26, 2013.

8. *Identification and Distributed Control of Large Scale Systems: an Application to Wind Farm.* **8th PhD Seminar on Wind Energy in Europe**, ETH Zürich, Switzerland, September 12-13, 2012.

9. *Fast Algorithms for SSS Matrix Problems and their Applications to Wind Farm Control.* **Dutch-Japanese Symposium on Numerical Linear Algebra, Algorithms, Applications, and Training**, Delft, the Netherlands, April 10-13, 2012.

Miscellany

10. **Fortieth Numerical Analysis Conference Woudschoten Past, Present and Future of Scientific Computing**. Zeist, the Netherlands, October 7-9, 2015.

11. **Het 50ste Nederlands Mathematisch Congres (NMC 2014)**. Delft, the Netherlands, April 16-17, 2014.

12. **Winter School on Hierarchical Matrices**. Max Planck Institute for Mathematics in the Sciences, Leipzig, Germany, February 24-27, 2014.

13. **Dutch Institute for Systems and Control Summer School on Optimization in Control Theory**. Noordwijk, the Netherlands, June 11-14, 2012.

14. **European Embedded Control Institute Graduate School on Decentralized and Distributed Control**. Supélec, Gif-sur-Yvette Cedex, France, February 20-24, 2012.

15. **7th PhD Seminar on Wind Energy in Europe**. Delft, the Netherlands, October 27-28, 2011.

Curriculum Vitae

物格而后知至, 知至而后意诚;
意诚而后心正, 心正而后身修.

– «大学»

When things are investigated, knowledge is extended.
When knowledge is extended, the will becomes sincere.
When the will is sincere, the mind is correct.
When the mind is correct, the self is cultivated.

– «The Great Learning»

– translated by A. Charles Muller

Yue Qiu (邱越 yuè qiū), was born and raised in Huainan (淮南 huái nán), Anhui (安徽 ān huī), China on September 26, 1989. After finishing high school in 2005, he joined the College of Information Science and Engineering, Northeastern University in Shenyang. During the bachelor study, he was funded by the National Undergraduate Students Innovation Program of China. He finished his bachelor thesis at the Institute of Electrical Engineering in July 2009 with *cum laude*. After that he was recommended to continue his study at the same institute to work on power grid control. His master thesis was entitled "Grid-Integration Wind Turbines Modeling and Their Networked Control", with which he obtained his master degree and being awarded Excellent Graduate Student of Northeastern University in July 2011.

From September 19, 2011, he started working as a PhD researcher at Delft University of Technology, where he did a joint research project carried out between Delft Institute of Applied Mathematics and Delft Center for Systems and Control. He is daily supervised by Dr. Ir. Martin B. van Gijzen. His PhD research aims at designing fast and robust numerical algorithms for optimal flow control problems using structured matrix computations. His research interest focuses on numerical

linear algebra and systems & control in general, specifically on preconditioning technique, structured matrix computations, numerical optimization, model order reduction and their applications to control of large-scale dynamical systems. In addition to research, he also conducts teaching assistance of numerical analysis for graduate courses of petroleum engineering.

From December 15, 2015, he will start working as a postdoctoral researcher in the Computational Methods in Systems and Control Theory group headed by Prof. Dr. Peter Benner at Max Planck Institute for Dynamics of Complex Technical Systems. He will perform his research on efficient numerical methods for control and optimization of large-scale dynamical systems.

Postface

虽有至知, 万人谋之

– 庄子

Although you are extremely smart,
you still need to take united wisdom to work things out.

– Chuang-Tze (369 - 286 BC)

Stepping into the mathematical world is pretty challenging for engineers and of great fun, of course. So many people, in one way or the other, have contributed to my four-year PhD research and life in the Netherlands. Right, it is you, who are reading this part right now most probably because reading a PhD dissertation out of one's specialization is quite tough. Without those people, the accomplishment of this dissertation will not be possible.

First of all, I would like to thank my promotors Michel and Kees, who give me the opportunity and freedom to perform colorful interdisciplinary research between systems & control and applied mathematics. Their support and motivation encourage me to step into new areas and reach a state that I have never imagined. I also want to thank Michel for the brainstorming during our meetings and his interest in every step I made in the research. Thanks to Kees for his insights, open-mindedness, kindness, and willingness, which benefit to not only my research, but also the development of my career and personality. I also would like to thank Kees for extending my contract to make me have a luxury moment to perform research only for my own interest after I finished my dissertation while waiting for the PhD defense. Special thanks to all the Vuik family members for the kind invitation and nice hosting of the visit.

Second and foremost, I would like to express my biggest gratitude to my PhD father Martin, who is always a benchmark for not only the way of thinking, doing research, and supervising my research, but also his great personality. I do think that I can never ask for a better tutor. I feel so lucky that I am being taken care of so well throughout my PhD life. Starting with engineering level of mathematical background, every step I make towards finishing this dissertation is attributed to his

effort. He is always positive to every (baby) step I made and supportive for all the ups and downs. His endless enthusiasm and constructive criticism made working with him a real pleasure. His thoughtfulness motivates me in both my personality and career development. He also founds a great PhD family that makes it enjoyable for all the family members.

I also want to thank Jan-Willem for bringing MSSS matrices and wind farm control to the research. Thanks for the input and comments during our regular meetings, also for the suggestions on my research development and career. Special thanks to Dr. Marielba Rojas for teaching me numerical optimization in the first year of my PhD and sharing interesting topics with me during her visit of Martin from the 5th floor to the 3rd floor. Also thanks to Marielba and Martin for the kind invitation to visit their home and sharing their books with me. Thanks to the nice gift from them for my marriage, and especially memorable birthday celebration. Thanks to making me really feel at home.

I would like to thank all my committee members for sparing time to reviewing my dissertation, sitting in a traditional Dutch PhD Promotion. Thanks to examining my dissertation carefully and giving me valuable comments that improve both the quality and readability. Thanks to Andy for reading my dissertation from A to Z and correct my language mistakes. Also thanks for his detailed remarks not only on the contents but also on the way of presenting research. Thanks to Peter for his articles that motivate me at the early stage of my PhD, his interest in my research and willingness to act as my committee member. I also would like to thank Peter for offering me a position at his research group so that I can continue conducting research on computational mathematics for systems and control. Thanks to Alle-Jan for his insights into my research and suggestive remarks. Thanks to Barry for his positive feedback on my dissertation. Thanks to Arnold for acting as reserved member of my PhD committee.

Special thanks to Dr. Laura Grigori at INRIA Paris for her interest in my research and offering me a postdoctoral position.

Thanks to my colleagues who created an awesome research group that is enjoyable to work at. Thanks to my PhD brothers Manuel and Reinaldo for the time we often spend together for academic activities, nerdy talks, and after work activities. Thanks to Fred, Duncan, Matthias, Kees, Peter, Domenico, Neil, Fons, Guus, Johan, Elisa, Elwin, Daniël, Menel, Kirsten, Tamara, Behrouz, Dennis, Xiaozhou, Jiao, Fei ($\times 2$), Jing, Peiyao, Rohit, Gabriela, Luis, Xin, Jakolien, Baljinnyam, Zaza, and Martijn to make the numerical analysis group attractive and relaxing. Our conversations on various topics are always refreshing and relieving. Thanks to Kees Lemmens for forcing me to get used to the Linux system, which greatly improved my experience on Linux. Thanks to Deborah for addressing all my paper work carefully and efficiently. Thanks to Haixiang for teaching me scientific programming and talking with me on different occasions.

Thanks to Cees and Franca at CICAT for their help in many aspects during the last four years. Special thanks to Cees for his motivation during my PhD.

Also thanks to my colleagues at DCSC for nice hosting of my irregular stay. Thanks to Heleen and Esther for arranging my order of books. Thanks to Saskia, Kitty and

Marieke for answering my questions in time. Thanks to Noortje, Yihui, Chengpu, Mohammad, Yashar, Dang, Zhe, Yu, Alex, Reinier, Elisabeth, Subramanya, Sachin, Anqi, Hai, Edwin, Gijs, Pieter, Yasin, Max, Bart, Le, Shuai (×2), Patricio, Dieky, Marco, Jia, Anna, Ruxandra, Yiming, Renshi, Hans (×2), Hildo, Zhou, Robert, Hans, Gabriel, and Bart for making my stay in DCSC enjoyable.

Thanks to all my officemates Mohammad, Shuai, Hans, Sachin at DCSC and Elwin, Xin, Lisa, Luis, Gabriela, Fei, and Tamara at DIAM. Thank you for making a lovely office that I really enjoy staying at. Also thanks to our regular talks on so many interesting topics ranging from research to life.

Thanks to all my roommates Bin (×2), Zhiyong, Jianbin, Jia, Long, Guangming, and Jelle for sharing apartments with me throughout my stay in the Netherlands. Thanks to good moments we spent together. Also thanks to Niha & Hao, Dongchao, Le, Xin, Carlos, Song, and Reinaldo for helping me move across the whole Delft in the last four years.

Thanks to the big Chinese community in Delft and the Netherlands for the steady support. Thanks to Dan & Zhenpei for the frequent hosting of my visit, for great talks on a wide range of topics, for so many opinions we have in common. Thanks to Niha & Hao for spending so many unforgettable moments together at the early stage of our arrival at the Netherlands, and for the great fun that you made throughout my Dutch life. Thanks to Dongchao for so many moments that we talked like two silly nerds and for the jokes he created that few people can understand. Thanks to Yu for enormous suggestions and advice for my life in the Netherlands. Thanks to Le for his kindness to respond to almost all my requests for help. Thanks to Yao, Yujie, Yongchang, Bin, Kang, Mingxin, Hongkai, Xi, Yannian, Xiaojun, Cong, Xiaojia & Shuanghou, Yongjia, Momo, Zhiyong, Shaoguang, Haoyu, Tianshi, Shuai, Lizuo for the great time early from Professor Street till now. Thanks to Jing & Alex, Peiyao, Xiaoyan, Jiao & Fei, Lu, Xiaozhou, Song, and Xin for interesting hangouts. Thanks to Anqi, Le, Ni & Yuexiang & Chengpu, Yihui & Jian, Shuai (×2), Zhou, Zhe, Yu, Yiming, Jia, Hai and Renshi for the great time together. Thanks to Jitang, Fan, Jianbin, Jie for being my playmates of basketball.

Thanks to all my international friends for cultivating a completely new and colorful land to me and thank you for letting me enjoy a great exotic life. Thanks to Tamara & Manuel, Leonie & Joost, Esther & Guido, Mathilde & Matthias, Chiara & Dave, Laura & Moritz, Lisa, Reinaldo, Fahim, Gabriela & Luis, and Abdul for the hangouts, for sharing different experiences. Special thanks to my best friend Manuel for letting me believe that I can beat Germans using beers and football (in electronic games). Thanks to Joost and Guido for the talks that cover so many topics, also for the time we spend together after work. Special thanks to Guido for translating my summary to Dutch. Thanks to Thea for regular talks, sharing life and bringing me English Bible. Thanks to Fahim for the weekends we spend together either working at the office or hanging out and making nice barbecue. Thanks to Lisa for being my runningmate, officemate and good friend. Thanks to Mohammad, Yashar, and Alfredo for introducing me nice food and drinks from their countries, for the game nights we spent together. Thanks to Ruxandra for the movie nights we organized, and for explaining system filtering techniques to me. Thanks to Noortje for introducing the Netherlands and DCSC to me when I just

started at Delft and answering my questions. Also thanks to showing me around Maastricht and the kind invitation to visit her place. Thanks to Subramanya, Sadegh, Noortje and Mohammad for our regular badminton playing. Special thanks to Mian for hosting my visit to the Max Planck Institute at Magdeburg and sharing his experience.

Thanks to my friends in China, who give consistent support from the first day we met. Thanks to Chong, Gang, Cheng, for being the biggest supportive force at all stages of my life. Thanks to Chao & Yanbo for our friendship for years. Special thanks to my supervisors and teachers at Northeastern University who motivate me for further study and show me the world of knowledge. Thanks to Prof. Dr. Zhitao Han for cultivating my interest in mathematics. Thanks to my bachelor and master supervisors Prof. Dr. Qiuye Sun and Prof. Dr. Huaguang Zhang for teaching me how to conduct research, and supporting me to pursue PhD abroad. Thanks to Prof. Dr. Dongsheng Yang, Prof. Dr. Jun Yang, Prof. Dr. Yanhong Luo, Prof. Dr. Haibin Shi and Dr. Dazhong Ma for continuously useful advice. Thanks to Prof. Dr. Huijun Gao at Harbin Institute of Technology and Prof. Dr. Wei Lin at Case Western Reserve University for their support throughout my master and PhD time.

Traveling over 5000 miles from China to the Netherlands, I am greatly indebted to my family. Thanks to my parents for their endless and unconditional love, which always make me motivated and optimistic towards every challenge in my life. Thanks to their absolute effort for creating me every possible opportunity for higher education, and their open-mindedness for supporting me to study abroad. Every step of my growth is attributed to their great love and endless support. Thanks to my parents in law for their understanding and support for my traveling abroad in the last few years. Finally, I would like to give my biggest gratitude to my beloved wife Rongcui Na. Thanks to her endless love and support to make me stay in peace and joy. Her confidence on me is even bigger than I have for myself. Being separated was really a hard time for both of us, and I think that she suffers more than I do. It is time to end this separation and I do believe we will never be separated.

All stories have an end, but our story has just started. I would like to thank all of you again for bringing me honorable and valuable experience to my life.

Yue Qiu

Delft, November 2015

邱越