DELFT UNIVERSITY OF TECHNOLOGY

REPORT 13-04

Efficient Preconditioners for PDE-Constrained Optimization Problems with a Multi-level Sequentially SemiSeparable Matrix Structure

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

ISSN 1389-6520

Reports of the Delft Institute of Applied Mathematics

Delft 2013

Efficient Preconditioners for PDE-Constrained Optimization Problems with a Multi-level Sequentially SemiSeparable Matrix Structure*

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

Revised on: 10th February 2014

Abstract

In this paper, we consider preconditioning for PDE-constrained optimization problems. The underlying problems yield a linear saddle-point system. We study a class of preconditioners based on multilevel sequentially semiseparable (MSSS) matrix computations. The novel global preconditioner is to make use of the global structure of the saddle-point system, while the block preconditioner makes use of the block structure of the saddle-point system. For the novel global preconditioner, it is independent of the regularization parameter, while for the block preconditioner, this property does not hold. For this MSSS matrix computation approach, 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 algorithm with low-rank approximated Gramians and the other is the standard Hankel blocks approximation algorithm. We test the global preconditioner and the block preconditioner for the problem of optimal control of the Poisson equation and optimal control of the convection-diffusion equation. Numerical experiments illustrate that both preconditioners give linear computational complexity and the global preconditioner yields the fewest number of iterations and computing time. Moreover, the approximate balanced truncation algorithm consumes less floating-point operations (flops) than the Hankel blocks approximation algorithm.

Keywords: PDE-constrained optimization, saddle-point problem, preconditioners, multilevel sequentially semiseparable matrix, model order reduction, low-rank approximation

1 Introduction

Optimal design, optimal control and parameter estimation of systems governed by partial differential equations (PDEs) give rise to a class of problems known as PDE-constrained optimization. PDE-constrained optimization problems have a wide application such as optimal flow control [1] [2], diffuse optical tomography [3], and linear (nonlinear) model predictive control [4]. The solution of these problems is acquired by solving a large-scale linear system of saddle-point type. Much effort has been dedicated to find efficient iterative solution methods for such systems. Some of the most popular techniques are the conjugate gradient (CG) [5], minimal residual (MINRES) [6], generalized minimal residual (GMRES) and induced

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

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

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

dimension reduction (IDR) [7]. The performance of these methods highly depends on the choice of preconditioners. In this paper, 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. the field of integral equations [8], Gauss-Markov processes [9], boundary value problems [10] and rational interpolation [11]. 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 by [12]. Sequentially semiseparable (SSS) matrices of which the off-diagonal blocks are of low-rank, not limited to 1, named by Dewilde et. al. in [13] generalize the semiseparable matrices. Multi-level sequentially semiseparable 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 [14] [15]. 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 still 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 complement. As a result, the LU factorization is not of linear computational complexity. The model order reduction plays a key role in reducing the rank of the off-diagonal blocks. Because of the model order reduction operation being performed, it is possible to compute an inexact LU decomposition of an MSSS matrix that can be used as a preconditioner.

In [14], Gondzio et. al. first introduced the preconditioning of PDE-constrained optimization problems by MSSS matrix computations. They exploited the MSSS matrix structure of the blocks of the saddle-point system and performed an LU factorization method for MSSS matrices to approximate the Schur complement of the saddle-point system. With the approximate Schur complement, conjugate gradient method was performed to solve the preconditioned 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. In [14], Gondzio et. al. used a standard model order reduction algorithm [16] [13] to reduce the computational complexity. In this paper, our work extends [14] in the following ways. 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. The new model order reduction algorithm is motivated by [17]. In [17], the approximate balanced truncation was addressed for the model order reduction of linear time invariant (LTI) systems. In this paper, we extend that method to the linear time varying (LTV) systems. Because of the correspondence between MSSS matrix and LTV systems, it is suitable for model order reduction for MSSS matrix computations. Compared with the conventional model order reduction algorithms in [13] [16], the approximate balanced truncation needs less floating-point operations (flops). 2) With these model order reduction algorithms, we can compute an inexact LU factorization of the MSSS matrix blocks of the saddle-point system in linear computational complexity. This yields block preconditioners for the saddle-point systems of the type that are described in [18] while only single preconditioner for the last block of the saddle-point system is studied in [14]. 3) By permuting the blocks of the saddle-point system, we can also compute an inexact LU factorization of the global system, which gives a novel global preconditioner. 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) We also extend these preconditioning technique to the 3D cases.

Note that the standard block preconditioners depend on the regularization parameter β for the PDE-constrained optimization problem [19]. By permuting the saddle-point system

with MSSS matrix blocks to a single MSSS matrix system, we can compute the inexact LU factorization of the global system in linear computational complexity, which is called the global preconditioner. Numerical experiments for the optimal control of the Poisson equation and the convection-diffusion equation illustrate that the performance of the global preconditioner is independent of the regularization parameter β and is also independent of the mesh size.

The structure of this paper is as follows: we start with formulating a distributed optimal control problem constrained by PDEs. This problem yields a linear system of the saddle point type. Demand for efficient preconditioners to solve this type of system with iterative solution methods motivates this paper. In Section 3, we briefly give an overview of some definitions and the widely used computations of MSSS matrices and then discuss the MSSS preconditioning technique. The novel model order reduction algorithm is also described. Based on the MSSS matrix computations, we propose three preconditioners for this saddle-point problem, they are the novel global preconditioner, the standard block-diagonal preconditioner and the standard block lower-triangular preconditioner. In Section 4, we use the distributed optimal control of the Poisson equation and the convection-diffusion equation as numerical experiments to illustrate the performance of our method. In Section 5, we extend this preconditioning technique to the optimal control of 3D problems. Section 6 draws the conclusion and describes future work.

2 Problem Formulation

2.1 PDE-Constrained Optimization Problem

Consider the PDE-constrained optimization problem described by

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

where $\mathcal{L}$ is an operator, u is the system state, f is the system input, $\hat{u}$ is the desired state of the system, β is the weight of the system input in the cost function or regularization parameter and $\beta > 0$. In this paper, we consider $\mathcal{L} = -\nabla^2$ for optimal control of the Poisson equation and $\mathcal{L} = -\epsilon\nabla^2 + \overrightarrow{w}\cdot\nabla$ for optimal control of the convection-diffusion equation. Here $\overrightarrow{w}$ is a vector in Ω, ∇ is the gradient operator, and ϵ 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 equivalent solutions while different answers are reached for the problem of optimal control of the convection-diffusion equation [19]. Since our focus is on multilevel sequentially semiseparable preconditioners, the *discretize-then-optimize* approach is chosen in this paper.

By introducing the weak formulation and discretizing (1) using the Galerkin method, the discrete analogue of the minimization problem (1) is therefore,

$$\begin{aligned} \min_{u,\ f}\ & \frac{1}{2}u^TMu-u^Tb+c+\beta f^TMf \\ s.t.\ \ & Ku=Mf+d, \end{aligned} \tag{2}$$

where $K=[K_{i,j}]\in\mathbb{R}^{N\times N}$ 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.

Consider the cost function in (2) and associate with the equality constrain, 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

$$\begin{bmatrix} 2\beta M & 0 & -M \\ 0 & M & K^T \\ -M & K & 0 \end{bmatrix} \begin{bmatrix} f \\ u \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ b \\ d \end{bmatrix}. \tag{3}$$

The system (3) is of the saddle-point system type [18], i.e., the system matrix, which is denoted as $\mathcal{A}$, has the following structure

$$\mathcal{A} = \begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix}, \tag{4}$$

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$. For system (3), we have $A = \begin{bmatrix} 2\beta M & 0 \\ 0 & M \end{bmatrix}$ and $B = \begin{bmatrix} -M & K \end{bmatrix}$.

The system matrix of the saddle-point system (3) is large and sparse. Thus it is amenable to solve such systems by preconditioned Krylov solvers, such as MINRES [6] and IDR(s) [7].

2.2 Preconditioning of Saddle-Point Systems

The performance of iterative solution methods highly depends on the quality of the preconditioners [20]. For numerical methods to solve system (3) and construction of preconditioners, we refer to [18] for an extensive survey of numerical methods for this type of systems. In this paper, we study three types of preconditioners. The first two types exploit the MSSS structure of the blocks of the saddle-point system, whereas the second type exploits the MSSS structure of the permuted saddle-point system.

2.2.1 Block Preconditioners

Recall from (4), if A is nonsingular, then $\mathcal{A}$ admits the following LDU 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 of computing the inverse of a large sparse matrix. Meanwhile, solving the system $Sx = b$ is also expensive since S is a large and full matrix. Note that all the matrix blocks

of (3) have a structure called multilevel sequentially semiseparable (MSSS), which will be introduced later. Then the Schur complement S also has the MSSS structure. If we exploit the MSSS structure of S, we can both compute S and solve the preconditioned system in linear complexity.

In this paper, we first study two types of block preconditioners for the saddle-point system. They are the block-diagonal preconditioner $\mathcal{P}_1$ and the block lower-triangular preconditioner $\mathcal{P}_2$, where

$$\mathcal{P}_1 = \begin{bmatrix} 2\beta\hat{M} & & \\ & \hat{M} & \\ & & -\hat{S} \end{bmatrix}, \ \mathcal{P}_2 = \begin{bmatrix} 2\beta\hat{M} & & \\ 0 & \hat{M} & \\ -M & K & \hat{S} \end{bmatrix}, \tag{5}$$

where $\hat{M}$ is an approximation of the mass matrix and $\hat{S}$ is an approximation of the Schur complement. For $\hat{M}$ and $\hat{S}$ without approximation, i.e., $\hat{M} = M$ and $\hat{S} = S$, the preconditioned system $\mathcal{P}_1^{-1}\mathcal{A}$ has three distinct eigenvalues and GMRES applied to the preconditioned system delivers the solution in at most three steps, while the preconditioned system $\mathcal{P}_2^{-1}\mathcal{A}$ has two distinct eigenvalues and GMRES applied to the preconditioned system delivers the solution in at most two steps [18]. For the general properties of $\mathcal{P}_1$ and $\mathcal{P}_2$, we refer to [18] for an extensive study.

As pointed out in [19], to approximate the Schur complement $S = -\left(\frac{1}{2\beta}M + KM^{-1}K^T\right)$, $\hat{S} = -KM^{-1}K^T$ could be used for big to middle range of β while $\hat{S} = -\frac{1}{2\beta}M$ could be chosen for small β. Thus the block-diagonal preconditioner is

$$\mathcal{P}_1 = \begin{bmatrix} 2\beta\hat{M} & & \\ & \hat{M} & \\ & & \hat{K}M^{-1}\hat{K}^T \end{bmatrix}, \tag{6}$$

for big or middle range of β, and

$$\mathcal{P}_1 = \begin{bmatrix} 2\beta\hat{M} & & \\ & \hat{M} & \\ & & \frac{1}{2\beta}\hat{M} \end{bmatrix}, \tag{7}$$

for small β, where $\hat{M}$ and $\hat{K}$ are approximated by MSSS matrix computation. Note that the sub-blocks of $\mathcal{P}_1$ and $\mathcal{P}_2$ all have an MSSS matrix structure such that the linear system $\mathcal{P}_1 y = r$ or $\mathcal{P}_2 y = r$ can be solved with linear computational complexity.

2.2.2 Global Preconditioners

Since the blocks of the saddle-point system (3) keep the MSSS matrix structure, it is possible to permute the saddle-point system (3) with MSSS matrix blocks to a linear system with global MSSS matrix structure, where the details will be introduced in the next section. Thus we have the permuted saddle-point system described by

$$\tilde{\mathcal{A}}\tilde{x} = \tilde{g}, \tag{8}$$

where $\tilde{\mathcal{A}}$, $\tilde{x}$ and $\tilde{g}$ 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 (3) and (4), respectively. Since the global matrix $\tilde{\mathcal{A}}$ of the permuted saddle-point system is an MSSS matrix, we can do an inexact LU factorization of $\tilde{\mathcal{A}}$ in linear computational complexity with MSSS matrix computations, i.e.,

$$\tilde{\mathcal{A}} \approx \tilde{L}\tilde{U}, \tag{9}$$

and use this inexact factorization as a preconditioner. We call this factorization in (9) the global preconditioner. Since no information of β is lost during the permutation and factorization, the global preconditioner is independent of β while for standard block preconditioner $\mathcal{P}_1$ and $\mathcal{P}_2$ in (5) this does not hold. This is a big advantage of the global preconditioner over the standard block preconditioner. Numerical examples in Section 4 verify this statement.

3 Preconditioning Using Multi-level Sequentially Semiseparable Matrix Computations

Matrices in this paper will always be real and their dimensions are compatible for the matrix-matrix operations and the matrix-vector operations when their sizes are not mentioned.

3.1 Multi-level Sequentially Semiseparable Matrices

The generators representation of sequentially semiseparable matrices are defined by Definition 3.1 [21].

Definition 3.1. *Let A be an $N \times N$ matrix with SSS matrix structure and let n positive integers $m_1,\ m_2,\ \cdots\ m_n$ with $N = m_1 + m_2 + \cdots + m_n$ such that A can be written in the following block-partitioned form*

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

where the superscript 'T' denotes the transpose of the matrix.

Table 1: Generator size for the SSS matrix A in Definition 3.1

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

The sequences $\{U_i\}_{i=1}^{n-1}$, $\{W_i\}_{i=2}^{n-1}$, $\{V_i\}_{i=2}^{n}$, $\{D_i\}_{i=1}^{n}$, $\{P_i\}_{i=2}^{n}$, $\{R_i\}_{i=2}^{n-1}$, $\{Q_i\}_{i=1}^{n-1}$ are matrices whose sizes are listed in Table 1 and they are called generators of the SSS matrix A. With the generators representation, the SSS matrix A is denoted as

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

Take $n = 5$ for example, the SSS matrix A is shown by (10),

$$A = \begin{bmatrix} D_1 & U_1V_2^T & U_1W_2V_3^T & U_1W_2W_3V_4^T & U_1W_2W_3W_4V_5^T \\ P_2Q_1^T & D_2 & U_2V_3^T & U_2W_3V_4^T & U_2W_3W_4V_5^T \\ P_3R_2Q_1^T & P_3Q_2^T & D_3 & U_3V_4^T & U_3W_4V_5^T \\ P_4R_3R_2Q_1^T & P_4R_3Q_2^T & P_4Q_3^T & D_4 & U_4V_5^T \\ P_5R_4R_3R_2Q_1^T & P_5R_4R_3Q_2^T & P_5R_4Q_3^T & P_5Q_4^T & D_5 \end{bmatrix}. \tag{10}$$

Remark 3.1. *The generators of a SSS matrix is not unique, there exists a series of non-singular transformations between two different sets of generators of the same SSS matrix A.*

With the generators representation of SSS matrices, basic operations of the underlying matrices such as addition, multiplication and inversion are closed under SSS matrix structure and can be done in linear computational complexity. Moreover, decomposition/factorization such as QR factorization [22] [23], LU/LDU decomposition [24] [14], and ULV decomposition [25] can also be computed in a structure preserving way. Many operations on SSS matrices can be performed with linear computational complexity. Examples are the matrix-matrix multiplication [21], the matrix-vector multiplication [21], the matrix inversion [24], the QR [22], LU [12], and ULV factorization [26]. To keep a clear structure of this paper, Table 2 lists the most widely used operations for SSS matrices and the corresponding references.

Table 2: Commonly used operations for SSS matrices

operations	Ax	$A \pm B$	AB	A^{-1}	LU	$Lx = b$ *
references	[13] [24] [21]	[13] [24] [21]	[13] [24] [21]	[23] [27] [28]	[24] [21]	[21]

* L is a lower-triangular SSS matrix.

Similar to Definition 3.1 for SSS matrices, the generators representation for MSSS matrices, specifically the k-level SSS matrices, are defined by Definition 3.2.

Definition 3.2. *The matrix A is said to be a k-level SSS matrix if all its generators are $(k-1)$-level SSS matrices. The 1-level SSS matrix is the SSS matrix that satisfies Definition 3.1.*

Operations listed in Table 2 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 the discretized partial differential equations (PDEs) [15] [14].

Example 3.1. *For the P_1 finite-element discretization of the 2D Laplacian equation with homogeneous Dirichlet boundary condition, the stiffness matrix K is given by*

$$K = \begin{bmatrix} A & B & & & \\ B & A & B & & \\ & B & \ddots & \ddots & \\ & & \ddots & \ddots & B \\ & & & B & A \end{bmatrix}, \text{ where } A = \begin{bmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & -1 & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 4 \end{bmatrix}, B = -I, \text{ and } I \text{ is}$$

the identity matrix. The matrix A and B are both SSS matrices be denoted by

$$\begin{aligned} A &= \mathcal{SSS}(1,\ 0,\ -1,\ 4,\ 1,\ 0,\ -1), \\ B &= \mathcal{SSS}(0,\ 0,\ 0,\ -1,\ 0,\ 0,\ 0). \end{aligned}$$

The matrix K has the MSSS (2-level SSS) matrix structure and is denoted by

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

Remark 3.2. *Similar with SSS matrices, for MSSS matrix, its generators are not unique. There exists a set of nonsingular transformations between two different sets of generators for a specified MSSS matrix.*

Remark 3.3. *For SSS or MSSS matrices, it is not necessary for their diagonals, sub-diagonal or super-diagonals to be constant like that in Example 3.1. Their sizes can even be different as long as the block-partitioned representation in Definition 3.1 is satisfied.*

Note that for a saddle-point system from the PDE-constrained optimization problem, all its blocks are MSSS matrices, which enables us to compute the LU factorization of all its blocks with MSSS matrix computations in linear computational complexity. However, we fail to compute the LU factorization of the whole saddle-point system matrix because the saddle-point system matrix is not an MSSS matrix but just has MSSS matrix blocks. The following lemma tells us how to permute a matrix with SSS matrix blocks to a single SSS matrix. We can easily extend this lemma to the MSSS matrix cases, which allows us to permute a matrix with MSSS matrix blocks to a single MSSS matrix.

Lemma 3.1. *[29] Let A, B, C and D be SSS matrices with the generators representations*

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

Then the relations

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

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

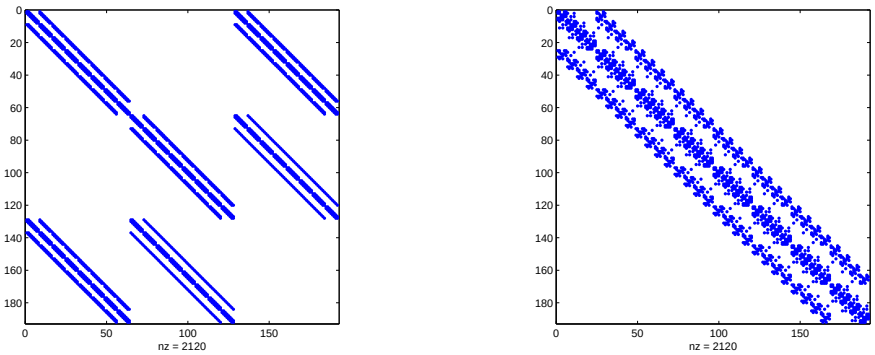
$$\mathcal{T} = \mathcal{SSS}(P_s^t, R_s^t, Q_s^t, D_s^t, U_s^t, W_s^t, V_s^t),$$

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

Remark 3.4. *Lemma 3.1 is for a* 2×2 *block matrix, but it can be extended to matrices with different number of blocks as well.*

Remark 3.5. *Extending Lemma 3.1 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 derived from the permutations of rows and blocks to get a single* $(k-1)$*-level SSS matrix by Lemma 3.1.*

For the saddle-point system (3) derived from the 2D PDE-constrained optimization problem, discretizing using P_1 finite element method yields a saddle-point system that has MSSS (2-level SSS) matrix blocks. The structure of the saddle-point system matrix for mesh size $h = 2^{-3}$ is shown in Figure 1(a). Permuting the saddle-point system using Lemma 3.1 gives system (8). The saddle-point system matrix structure before and after permutation are shown in Figure 1.



(a) Before permutation.

(b) After permutation.

Figure 1: Structure of system matrix of (3) before and after permutation for $h = 2^{-3}$.

3.2 Multi-level Sequentially Semiseparable Preconditioners

The ability to solve a linear system with MSSS matrix structure in linear computational complexity is essential for the purpose of this paper. One way is to compute the *LU* fac-

torization of the system matrix with MSSS matrix computations. In the following part, we first introduce the LU factorization of MSSS matrices and then give a novel model order reduction algorithm for SSS matrices that is required in computing the LU factorization. For comparison, the conventional model order reduction algorithm is also discussed.

3.2.1 LU Factorization of Multilevel Sequentially Semiseparable Matrices

The semiseparable order defined in Definition 3.3 plays an important rule in the MSSS matrix computations. Note that Dewilde et. al. and Golberg et. al. studied this kind of structured matrices separately, SSS matrices named in [21] are called quasiseparable matrices in [24]. Here 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 3.3. *[16] Let*

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

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

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

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

Definition 3.4. *[30] The SSS matrix A with lower and upper semiseparable order r^l and r^u is called block $(r^l,\ r^u)$ semiseparable.*

The definitions in Definition 3.3 and 3.4 of SSS matrices can be directly extended to the MSSS matrices, which leads to Definition 3.5 and 3.6.

Definition 3.5. *Let the matrix A be an $N \times N$ block k-level SSS matrix with its generators be $M \times M$ block $(k-1)$-level SSS matrices. Let*

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

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

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

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

Definition 3.6. *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.*

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

Lemma 3.2. *[12][14] Let A be a strongly regular $N \times N$ block k-level sequentially semiseparable matrix of k-level block $(r^l,\ r^u)$ semiseparable and denoted by its generators representation $A = \mathcal{MSSS}(P_s,\ R_s,\ Q_s,\ D_s,\ U_s,\ W_s,\ V_s)$. Let $A = LU$ be its block LU factorization. Then,*

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

2. *The factors L and U can be denoted by the generators representation*

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

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

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

Initialize: $M_1 \leftarrow 0 \in \mathbb{R}^{r^l \times r^u}$ be a $(k-1)$-level SSS matrix
Compute the LU factorization of the $(k-1)$-level SSS matrix
$D_1 = D_1^L D_1^U$, let $\hat{U}_1 = (D_1^L)^{-1} U_1$ and $\hat{Q}_1 = (D_1^L)^{-T} Q_1$
for $i = 2 : N-1$ **do**
$M_i = \hat{Q}_{i-1}^T \hat{U}_{i-1} + R_i M_{i-1} W_i$,
Compute the LU factorization of the $(k-1)$-level SSS matrix
$(D_i - P_i M_i V_i) = D_i^L D_i^U$,
Let, $\hat{U}_i = (D_i^L)^{-1}(U_i - P_i M_{i-1} W_i)$, $\hat{Q}_i = (D_i^U)^{-T}(Q_i - V_i M_{i-1}^T R_i^T)$.
end for
Compute the LU factorization of the $(k-1)$-level SSS matrix
$\left(D_n - P_n M_{n-1} V_n^T\right) = D_n^L D_n^U$
Output: D_i^L, D_i^U, $\hat{Q}_i$, $\hat{U}_i$

Proof. For the proof of the lemma, we refer to [12] and [14]. □

Remark 3.6. *In Algorithm 1, the LU factorization of a 0-level SSS matrix is just the LU factorization of an ordinary matrix without SSS structure.*

For MSSS matrices, matrix-matrix operations such as addition and multiplication will lead to a growth of the semiseparable order, which can be verified from the matrix-matrix operations introduced in [21] [24]. This results in the growth of the computational complexity. Take the 1-level SSS matrix A for example, the flops needed for computing A^2 is $40n^3N$ where n is the semiseparable order [21] and N is the number of blocks of A. To be specific, the following lemma is introduced.

Lemma 3.3. *[24] Let A_1, A_2 be SSS matrices of sizes $N \times N$ which are lower semiseparable of orders m_1, n_1 respectively. Then the product $A_1 A_2$ is lower semiseparable of order at most $m_1 + n_1$. Let A_1, A_2 be SSS matrices of sizes $N \times N$ which are upper semiseparable of orders m_2, n_2 respectively. Then the product $A_1 A_2$ is upper semiseparable of order at most $m_2 + n_2$.*

Remark 3.7. *For k-level SSS matrices, since semiseparable order varies at different levels, result of Lemma 3.3 holds for the k-level semiseparable order. But we do not know the $(k-1)$-level semiseparable order of the $(k-1)$-level SSS generators exactly, we just know the $(k-1)$-level semiseparable order also increases.*

Lemma 3.3 gives rise to the question whether there exists a minimal semiseparable order for a SSS matrix such that the SSS matrix with a bigger semiseparable order is equivalent to an SSS matrix with minimal semiseparable order. Definition 3.7 and Lemma 3.4 give the answer to the aforementioned question.

Definition 3.7. *[16] We say that the lower generators $P_i (i = 2, \ldots, N)$, $Q_j (j = 1, \ldots, N-1)$, $R_k (k = 2, \ldots, N-1)$ of an SSS matrix A are minimal if all their orders $l_k (k = 1, \ldots, N-1)$ are as small as possible among all lower generators of the same matrix A, i.e., for lower generators of the matrix A with orders $l_k' (k = 1, \ldots, N-1)$, the inequalities*

$$l_k \leq l_k^{'},\ k = 1, \ldots, N-1$$

hold. We also say that the orders $l_k (l = 1, \ldots, N-1)$ *are the minimal orders of the lower generators of* A*.*

Lemma 3.4. *[16] Let* $A = \{A_{ij}\}_{i,j=1}^{N}$ *be a block matrix with lower rank numbers* $r_k (k = 1, \ldots, N-1)$*. Then* A *has lower generators with orders equal to the corresponding rank numbers. Moreover, for any matrices, the rank numbers are the minimal orders of the generators.*

Remark 3.8. *Lemma 3.4 can be extended to the* k*-level SSS matrices directly.*

Remark 3.9. *Lemma 3.4 shows that there exists a minimal semiseparable order for an SSS matrix. Thus, for an SSS matrix of semiseparable order bigger than the minimal separable order, the semiseparable order can be reduced to make the reduced semiseparable order equal to or smaller than the minimal semiseparable order such that the resulting SSS matrix with reduced semiseparable order is equal to or equivalent with the SSS matrices without order reduction up to a small tolerance.*

The aim of model order reduction of 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 size 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., $\|\hat{A} - A\| < \epsilon$.

Remark 3.10. *In Algorithm 1, for computing the LU factorization of a* k*-level SSS matrix, matrix-matrix operations are performed on its generators which are* $(k-1)$*-level SSS matrices. Such operations lead to the growth of semiseparable order of the* $(k-1)$*-level SSS matrices, which induces growth of computational complexity. Model order reduction is necessary to reduce the semiseparable order or keep the semiseparable order under a threshold during the LU factorization, such as computation of the recurrence of* M_i *in Algorithm 1.*

Remark 3.11. *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 3.2 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. Therefore, for discretized one-dimensional PDEs on a regular grid, this factorization could be performed as a direct solver and as an efficient preconditioner for the discretized two- or higher- dimensional PDEs on a regular grid.*

Remark 3.12. *The model order reduction algorithm for SSS matrices has been studied in [13] [16], while for* $2-$*level or even higher-level SSS matrices, it is still a big challenge since model order reduction of* k*-level SSS matrices where* $k \geq 2$ *needs the reduced generators still be* $(k-1)$*-level SSS matrices. The model order reduction algorithms in [13] [16] applied to the* k*-level SSS matrices will not return structure preserving* $(k-1)$*-level SSS matrix generators.*

3.2.2 Approximate Balanced Truncation

In this paper, we design a novel model order reduction algorithm for SSS matrices. With this algorithm, we can construct an efficient preconditioner for two-dimensional PDE-constrained optimization problem, which will be studied in the next section. The correspondence between SSS matrices and the linear time-varying (LTV) systems motivates us to derive this new model order reduction algorithm.

The SSS matrices have a realization of linear time-varying systems, which is studied by Dewilde et. al. in [27]. Consider a mixed-causal system that is described by the following state-space model

$$\begin{aligned} \begin{bmatrix} x_{i+1}^c \\ x_{i-1}^a \end{bmatrix} &= \begin{bmatrix} R_i & \\ & W_i \end{bmatrix} \begin{bmatrix} x_i^c \\ x_i^a \end{bmatrix} + \begin{bmatrix} Q_i \\ V_i \end{bmatrix} u_i \\ y_i &= \begin{bmatrix} P_i & U_i \end{bmatrix} \begin{bmatrix} x_i^c \\ x_i^a \end{bmatrix} + D_i u_i, \end{aligned} \tag{11}$$

where x^c denotes the causal system states, x^a represents the anti-causal system states, 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,

$$\mathcal{H} = \begin{bmatrix} D_i & U_1V_2 & U_1W_2V_3 & U_1W_2W_3V_4 \\ P_2Q_1 & D_2 & U_2V_3 & U_2W_3V_4 \\ P_3R_2Q_1 & P_3Q_2 & D_3 & U_3V_4 \\ P_4R_3R_2Q_1 & P_4R_3Q_2 & P_4Q_3 & D_4 \end{bmatrix}. \tag{12}$$

Remark 3.13. *To reduce the semiseparable order of the SSS matrix $\mathcal{H}$ in (12), the orders of P_s, R_s, Q_s, U_s, W_s and V_s need to be reduced. This corresponds to reduce the order of the mixed-causal LTV system (11). Model reduction for LTV system (11) could be performed to reduce the semiseparable order of $\mathcal{H}$.*

Model order reduction for LTV systems is studied in [31] [32]. In [32], a linear matrix inequality (LMI) was introduced to solve the Lyapunov inequalities for the controllability and observability Gramians. In [31], the low-rank Smith method was presented to approximate the square-root of the controllability and observability Gramians.

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

$$\begin{cases} x_{k+1} = R_k x_k + Q_k u_k \\ y_k = P_k x_k, \end{cases} \tag{13}$$

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 from the following Stein recurrence formulas:

$$\mathcal{G}_c(k+1) = R_k \mathcal{G}_c(k) R_k^T + Q_k Q_k^T, \tag{14}$$

$$\mathcal{G}_o(k) = R_k^T \mathcal{G}_o(k+1) R_k + P_k^T P_k, \tag{15}$$

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 partly controllable and observable, thus their eigenvalues are non-negative. Their eigenvalues often have a large jump at an early stage as pointed out in [17] [33] [34] [35], which suggests to approximate these two Gramians at each step by a low-rank approximation. Below we show how to obtain such approximations. Since the controllability Gramian $\mathcal{G}_c(k)$ and observability Gramian $\mathcal{G}_o(k)$ have similar structure, we will only focus on the controllability Gramian $\mathcal{G}_c(k)$.

The key point of the low-rank approximation is to substitute the Cholesky factorization of the controllability Gramian $\mathcal{G}_c(k)$

$$\mathcal{G}_c(k) = L_k L_k^T, \tag{16}$$

where $L_k \in \mathbb{R}^{N\times N}$ in each step k by its approximate Cholesky factorization,

$$\tilde{\mathcal{G}}_c(k) = \tilde{L}_k \tilde{L}_k^T, \tag{17}$$

with $\tilde{L}_k \in \mathbb{R}^{N\times n_k}$ where n_k is the numerical rank of $\mathcal{G}_c(k)$ and $N > n_k$ at each step k. Typically, n_k is set to be constant, i.e., $n_k = n$ at each step. Since $\mathcal{G}_c(k)$ is of low numerical rank, it is reasonable to use the rank n_k factor $\tilde{L}_k$ to approximate $\mathcal{G}_c(k)$.

In [17], a recursive low-rank Gramian method was introduced to approximate the Gramians of a linear time-invariant system. Here, we extend that method to the linear time-varying systems, which is similar with the method in [36]. This method is shown in Algorithm 2. From [17], we know that $\tilde{\mathcal{G}}_c(i) = \tilde{L}_c(i)\tilde{L}_c(i)^T$ and $\tilde{\mathcal{G}}_o(i) = \tilde{L}_o(i)\tilde{L}_o(i)^T$ in Algorithm 2 are the best rank n approximations to $\mathcal{G}_c(i)$ and $\mathcal{G}_o(i)$.

Algorithm 2 Low-rank approximation of the Gramians

Initialize: $\tilde{\mathcal{G}}_c(1) \leftarrow 0 \in \mathbb{R}^{M\times n}$, $\tilde{\mathcal{G}}_o(N+1) \leftarrow 0 \in \mathbb{R}^{M\times n}$, N is the number of time steps, M is the unreduced order, n is the numerical rank.

for i=2: N **do**

Compute the singular value decompositions

$\left[\begin{array}{c|c} Q_{i-1} & R_{i-1}\tilde{\mathcal{G}}_c(i-1) \end{array}\right] = U_c\Sigma_c V_c^T$, $\left[\begin{array}{c|c} P_i^T & R_i^T\tilde{\mathcal{G}}_o(i+1) \end{array}\right] = U_o\Sigma_o V_o^T$.

Let

$U_c = \left[\begin{array}{c|c} U_{c1} & U_{c2} \end{array}\right]$, $\Sigma_c = \left[\begin{array}{c|c} \Sigma_{c1} & \\ \hline & \Sigma_{c2} \end{array}\right]$ with $U_{c1} \in \mathbb{R}^{M\times n}$ and $\Sigma_{c1} \in \mathbb{R}^{n\times n}$.

$U_o = \left[\begin{array}{c|c} U_{o1} & U_{o2} \end{array}\right]$, $\Sigma_o = \left[\begin{array}{c|c} \Sigma_{o1} & \\ \hline & \Sigma_{o2} \end{array}\right]$ with $U_{o1} \in \mathbb{R}^{M\times n}$ and $\Sigma_{o1} \in \mathbb{R}^{n\times n}$.

Make

$\tilde{L}_c(i) = U_{c1}\Sigma_{c1}$, $\tilde{L}_o(i) = U_{o1}\Sigma_{o1}$.

end for

Output: $\tilde{L}_c(i) \in \mathbb{R}^{M\times n}$ and $\tilde{L}_o(i) \in \mathbb{R}^{M\times n}$.

With the approximate controllability Gramian $\mathcal{G}_c(i)$ and observability Gramian $\mathcal{G}_o(i)$, the balanced truncation could be performed to reduce the order of the LTV system. For the approximate balanced truncation, the key is to use the low-rank approximation of the factors of Gramians to provide an approximation to the balanced truncation.

For the LTV system (13), to do a balanced truncation, first the system states are transformed by the nonsingular transformation $x_k = T_k\bar{x}_k$ to get a "balanced" system,

$$\begin{cases} \bar{x}_{k+1} = T_{k+1}^{-1}R_kT_kx_k + T_{k+1}^{-1}Q_ku_k \\ y_k = P_kT_kx_k, \end{cases} \tag{18}$$

where the states $\bar{x}_k = \begin{pmatrix}\tilde{x}_k^T & \hat{x}_k^T\end{pmatrix}^T$. The kept system states are $\tilde{x}_k = \begin{bmatrix} I_n & 0\end{bmatrix}\bar{x}_k$ where n is the system order after reduction. The reduced LTV system of (13) is

$$\begin{cases} \tilde{x}_{k+1} = \Pi_l(k+1)R_k\Pi_r(k)\tilde{x}_k + \Pi_l(k+1)Q_ku_k \\ y_k = P_k\Pi_r(k)\tilde{x}_k, \end{cases} \tag{19}$$

where $\Pi_l(k+1) = \begin{bmatrix} I_n & 0\end{bmatrix}T_{k+1}^{-1}$ and $\Pi_r(k) = T_k\begin{bmatrix} I_n \\ 0\end{bmatrix}$.

Next, we extend the balanced truncation algorithm to the linear time-varying case. This method is described in Algorithm 3.

Remark 3.14. *The second loop of Algorithm 3 ensures that $\Pi_l(i)$ and $\Pi_r(i)$ are "balanced". This is vital since we approximate the controllability and observability Gramians independently.*

Remark 3.15. *With Algorithm 2 and Algorithm 3, the LTV system (13) was reduced to (19) by the low-rank approximate balanced truncation.*

Remark 3.16. *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, Algorithm 2 and Algorithm 3 could be performed to the strictly lower-triangular part of A to reduce the lower semiseparable order to n, yielding the approximate 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, first transpose it to be strictly lower-triangular then perform Algorithm 2 and Algorithm 3. After the reduction, transpose the reduced strictly lower-triangular part to be strictly upper-triangular.*

Algorithm 3 Approximate balanced truncation for LTV systems

Procedure: Set the numerical rank n.
Use the low-rank approximation Algorithm 2 to compute the rank n approximations to the controllability Gramian $\mathcal{G}_c(i)$ and observability Gramian $\mathcal{G}_o(i)$, denoted by $\tilde{\mathcal{G}}_c(i)$ and $\tilde{\mathcal{G}}_o(i)$, respectively.
loop
Compute the singular value decomposition
$\tilde{\mathcal{G}}_c^T(i)\tilde{\mathcal{G}}_o(i) = U_i\Sigma_i V_i^T$.
end loop
loop
Let
$\Pi_l(i) = \tilde{\mathcal{G}}_o(i)V_i\Sigma_i^{-\frac{1}{2}}$, $\Pi_r(i) = \tilde{\mathcal{G}}_c(i)U_i\Sigma_i^{-\frac{1}{2}}$.
end loop
End Procedure
Output: $\Pi_l(i) \in \mathbb{R}^{n\times M}$ and $\Pi_r(i) \in \mathbb{R}^{M\times n}$.

3.2.3 Hankel Blocks Approximation

The model order reduction algorithms for SSS matrices in [13] [21] [27] approximate the Hankel blocks of the SSS matrices, where the Hankel blocks of an SSS matrix A are defined by Definition 3.8.

Definition 3.8. *[13] 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 SSS matrix A for example, the Hankel blocks for the strictly upper triangular part are shown in Figure 2 by H_1, H_2 and H_3.

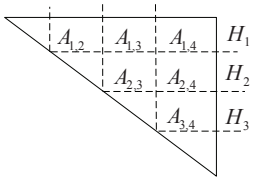


Figure 2: Hankel blocks of a SSS matrix A

The model order reduction algorithms in [13] [21] [27] are *Hankel norm* optimal order reduction [29] algorithms. That is, given an SSS matrix A with a lower semiseparable order r_L and an upper semiseparable order r_U, we can get an approximate SSS matrix $\hat{A}$ with a lower semiseparable order $\tilde{r}_L$ and an upper semiseparable order $\tilde{r}_U$ where $r_L > \tilde{r}_L$, $r_U > \tilde{r}_U$ achieves

$$\inf\ \|A - \hat{A}\|_H\ ,$$

where $\|A\|_H = \max_i \|H_i(A)\|_2$ and $H_i(A)$ are the Hankel blocks of A defined in Definition 3.8.

For comparison, this model order reduction algorithm to the strictly upper-triangular part of SSS matrices is listed in Algorithm 4 [13].

Algorithm 4 Hankel blocks approximation for SSS matrices

Initialize: $H \leftarrow 0 \in \mathbb{R}^{M\times M}$, $G \leftarrow 0 \in \mathbb{R}^{M\times M}$, M is the upper semiseparable order before reduction, set the reduced upper semiseparable order m and the number of blocks N.

perform the forward recursion

for $i = 1 : N-1$ **do**

Compute the singular value decomposition (SVD) $\begin{bmatrix} H \\ U_i \end{bmatrix} = U\Sigma V^T$ and partition $U = \left[\begin{array}{c|c} U_T \\ \hline U_K \end{array}\right] = \left[\begin{array}{c|c} U_a & (\cdot) \\ \hline (\cdot) & (\cdot) \end{array}\right]$ with $U_K \in \mathbb{R}^{n_u\times(\cdot)}$, $U_a \in \mathbb{R}^{M\times M}$, where n_u is the number of rows of U_i.

Let,

$U_i = U_K$, $W_i = U_a$, $V_{i+1} = \Sigma V^T V_{i+1}$ and $H = \Sigma V^T W_{i+1}$.

end for

perform the backward recursion

for $i = N : -1 : 2$ **do**

Compute the singular value decomposition (SVD) $\begin{bmatrix} V_i \\ G^T \end{bmatrix} = U\Sigma V^T$ and partition $U = \left[\begin{array}{c|c} U_a & (\cdot) \\ \hline U_b & (\cdot) \end{array}\right]$ with $U_a \in \mathbb{R}^{n_v\times M}$, $U_b \in \mathbb{R}^{M\times M}$, $\Sigma = \left[\begin{array}{c} \Sigma_a \\ \hline (\cdot) \end{array}\right]$ with $\Sigma_a \in \mathbb{R}^{M\times M}$, where n_v is the number of rows of V_i.

Let,

$V_i = U_a$, $U_{i-1} = U_{i-1}V\Sigma_a^T$, $W_i = U_b^T$, $G = W_{i-1}V\Sigma_a^T$.

end for

do the truncation

for $i = 1 : N$ **do**

Partition

$U_i = \left[\begin{array}{c|c} U_{i1} & (\cdot) \end{array}\right]$ with $U_{i1} \in \mathbb{R}^{(\cdot)\times m}$,

$W_i = \left[\begin{array}{c|c} W_{i1} & (\cdot) \\ \hline (\cdot) & (\cdot) \end{array}\right]$ with $W_{i1} \in \mathbb{R}^{m\times m}$,

$V_i = \left[\begin{array}{c|c} V_{i1} & (\cdot) \end{array}\right]$ with $V_{i1} \in \mathbb{R}^{m\times(\cdot)}$.

Let,

$\tilde{U}_i = U_{i1}$, $\tilde{W}_i = W_{i1}$, $\tilde{V}_i = V_{i1}$.

end for

Output: $\tilde{U}_i \in \mathbb{R}^{(\cdot)\times m}$, $\tilde{W}_i \in \mathbb{R}^{m\times m}$ and $\tilde{V}_i \in \mathbb{R}^{m\times(\cdot)}$.

Remark 3.17. *With the Hankel blocks approximation, we can also construct an efficient preconditioner for two-dimensional PDE-constrained optimization problem, which will be studied in the next section.*

Remark 3.18. *For an SSS matrix A with lower and upper semiseparable order r_l and r_u, respectively. The bigger the semiseparable order $\hat{r}_l$ and $\hat{r}_u$ after model order reduction by Algorithm 2-3 or 4 is, the closer the reduced SSS matrix $\hat{A}$ is to A. For a proper semiseparable order set, the model order reduction is accurate enough. This makes the LU factorization of the 2-level SSS matrix by Algorithm 1 accurate enough that can be performed as a direct solver. Numerical experiments in the next section illustrate this.*

Given an SSS matrix $A = \mathcal{SSS}(P_S,\ R_s,\ Q_s,\ D_s,\ U_s,\ W_s,\ V_s)$, to compare the flops of the approximate balanced truncation in Algorithm 2-3 and the Hankel blocks approximation Algorithm 4, we assume that the generators sizes in Table 3.1 are $m_i = n$ and $k_i = l_i = M$ where N is the number of SSS blocks and $N \gg M \gg n$. This is easy to verify from the

matrix-matrix operations in [13] [24] such as the multiplication and addition. The reduced SSS matrix $\tilde{A} = \mathcal{SSS}(\tilde{P}_S,\ \tilde{R}_s,\ \tilde{Q}_s,\ D_s,\ \tilde{U}_s,\ \tilde{W}_s,\ \tilde{V}_s)$, where $\tilde{k}_i = \tilde{l}_i = m$, m is the reduced semiseparable order and $m \ll M$. For Algorithm 2-3, the flops count $\mathcal{F}_N$ are

$$\mathcal{F}_N = \mathcal{O}\left((3m^2 + 4mn + n^2)MN + (m+n)M^2N\right), \tag{20}$$

while the flops count $\mathcal{F}_C$ for Algorithm 4 is

$$\mathcal{F}_C = \mathcal{O}\left(M^3N + (2m+n)M^2N + 2mnMN\right). \tag{21}$$

Since $N \gg M \gg m,\ n$, we describe that

$$\mathcal{F}_N = \mathcal{O}\left(M^2N\right), \tag{22}$$

and

$$\mathcal{F}_C = \mathcal{O}\left(M^3N\right). \tag{23}$$

Remark 3.19. *From (22) and (23), it is obvious that both model order reduction algorithm for SSS matrices have linear computational complexity $\mathcal{O}(N)$, while the approximate balanced truncation (flops count denoted by $\mathcal{F}_N$) is computationally cheaper than the Hankel blocks approximation (flops count denoted by $\mathcal{F}_C$) for large enough M. This will also be illustrated by numerical experiments in the next section.*

Remark 3.20. *As stated in [32], the balanced truncation yields an optimal induced L_2-norm approximation. Thus for the approximate balanced truncation, the reduced control system is close to the optimal L_2-norm approximation. The Algorithm 4 returns the optimal Hankel-norm approximation. Thus, both algorithms for model order reduction of SSS matrices will yield an accurate approximation. Since the inequality $\|Z\|_H \leqslant \|Z\|_2 \leqslant \sqrt{N}\|Z\|_H$ for all $Z \in \mathbb{R}^{n\times n}$ holds [29], the Hankel blocks approximation Algorithm 4 yields a more accurate approximation than the approximate balanced truncation Algorithm 2-3 in theory. But the accuracy of Algorithm 4 and Algorithm 2-3 are comparable, which will be shown by numerical experiments in the next section.*

4 Numerical Experiments

We study two test examples for optimal control of 2D PDEs in this section, i.e., optimal control of the convection-diffusion equation in Example 4.1 and optimal control of the Poisson equation in Example A.1 in the appendix. We apply the block-diagonal preconditioner $\mathcal{P}_1$ in (5) for the MINRES method and the lower-triangular preconditioner $\mathcal{P}_2$ in (5) for the IDR(s) method to both examples. The global preconditioner $\hat{\mathcal{A}}$ in (9) is also performed for the two test examples to show its superior performance over the standard block preconditioners for saddle-point system.

Example 4.1. *[19] Let $\Omega = \{(x,y)|0 \leq x \leq 1, 0 \leq y \leq 1\}$ and consider the problem*

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

where $\Gamma_D = \partial\Omega$ and

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

ϵ is a positive scalar, $\overrightarrow{\omega}$ is the unit directional vector that $\overrightarrow{\omega} = (cos(\theta),\ sin(\theta))^T$ 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 4Gb memory with Matlab R2010b. The stop tolerance of the 2-norm of the relative residual is set to be 10^{-6} for all the numerical experiments. 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 is in the brackets following the problem size. The time to compute the preconditioners and iterative solution methods time is measured in seconds.

4.1 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 in Example 4.1. With the block-diagonal preconditioner $\mathcal{P}_1$ by approximate balanced truncation and the Hankel blocks approximation methods, the results for different values of ϵ and β are shown in Table 3 - 10, while θ was set to be $\frac{\pi}{5}$. The preconditioning column represents the time to compute the preconditioners.

Table 3: By approximate balanced truncation for $\beta = 10^{-1}$, $\epsilon = 10^{-1}$

problem size	iterations	preconditioning	MINRES	total
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 4: By Hankel blocks approximation for $\beta = 10^{-1}$, $\epsilon = 10^{-1}$

problem size	iterations	preconditioning	MINRES	total
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 5: By approximate balanced truncation for $\beta = 10^{-1}$, $\epsilon = 10^{-2}$

problem size	iterations	preconditioning	MINRES	total
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 6: By Hankel blocks approximation for $\beta = 10^{-1}$, $\epsilon = 10^{-2}$

problem size	iterations	preconditioning	MINRES	total
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 7: By approximate balanced truncation for $\beta = 10^{-2}$, $\epsilon = 10^{-1}$

problem size	iterations	preconditioning	MINRES	total
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 8: By Hankel blocks approximation for $\beta = 10^{-2}$, $\epsilon = 10^{-1}$

problem size	iterations	preconditioning	MINRES	total
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

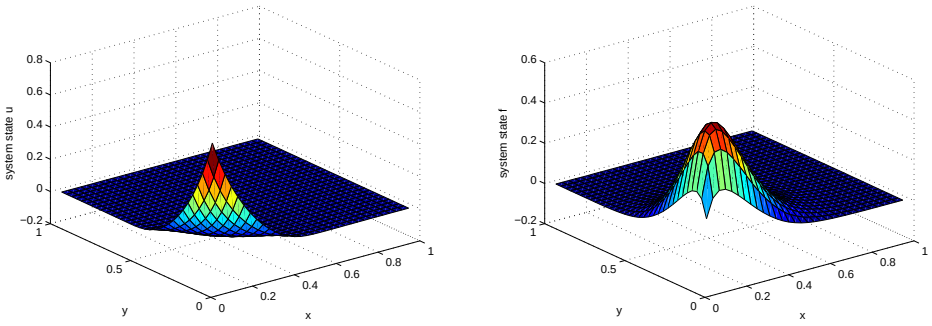
Table 9: By approximate balanced truncation for $\beta = 10^{-2}$, $\epsilon = 10^{-2}$

problem size	iterations	preconditioning	MINRES	total
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 10: By Hankel blocks approximation for $\beta = 10^{-2}$, $\epsilon = 10^{-2}$

problem size	iterations	preconditioning	MINRES	total
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 optimal solution of the system states and input for $\beta = 10^{-2}$, $\epsilon = 10^{-1}$ and $h = 2^{-5}$ are shown in Figure 3(a) and 3(b).



(a) Optimal system states u.

(b) Optimal system input f.

Figure 3: Solution of the system states and input for $\beta = 10^{-2}$, $\epsilon = 10^{-1}$ and $h = 2^{-5}$.

Remark 4.1. *As shown by (22) and (23), 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 preconditioner for the same values of β and ϵ in Table 3 - 10.*

For the results of the block-diagonal preconditioners with two model reduction algorithms for optimal control of the Poisson equation, please refer to appendix A.1. For the comparison results of the two model order reduction algorithms for the block lower-triangular preconditioner, please refer to appendix A.2. These results coincide with the conclusions for the performance of the two model order reduction algorithms in Table 3 - 10.

4.2 Comparison of Preconditioners

In this part, we compare the performance of the block-diagonal preconditioner and the global preconditioner. From Table 3 - 10, we see that with the decrease of β, the number of iterations increases slightly for the same problem size and ϵ. This is due to the $\frac{1}{2\beta}M$ term plays an increasing important rule with the decrease of β. This term is neglected in the preconditioner $\mathcal{P}_1$ in (6) for big and middle value of β [19]. If we continue decreasing β for the optimal control of the convection-diffusion equation, we have the computational results in Table 11-12. In this part, the model order reduction algorithm is chosen as the Hankel blocks approximation method. For the results of approximate balanced truncation, please refer to appendix B.

Table 11: By the block-diagonal preconditioner in (6) for $\beta = 10^{-3}$, $\epsilon = 10^{-1}$

problem size	iterations	preconditioning	MINRES	total
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 12: By the block-diagonal preconditioner in (6) for $\beta = 10^{-4}$, $\epsilon = 10^{-1}$

problem size	iterations	preconditioning	MINRES	total
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 11-12, with the decrease of β from 10^{-3} to 10^{-4}, the number of iterations increase from 34 to 82. It is not difficult to imagine that when β continues decreasing, the performance of the block-diagonal preconditioner $\mathcal{P}_1$ in (6) cannot give satisfied performance. Next we test the performance of the preconditioner $\mathcal{P}_1$ in (7) for $\beta = 10^{-4}$. The computational results are shown in Table 13. The maximum number of iterations is set to 100.

Table 13: By the block-diagonal preconditioner in (7) for $\beta = 10^{-4}$, $\epsilon = 10^{-1}$

problem size	iterations	preconditioning	MINRES	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

As shown by Table 12-13, the block-diagonal preconditioner does not work well for small β. Since the global preconditioner does not neglect any information of β, we test the performance of the global preconditioner in the following part.

Recall that in Section 2, we can permute the saddle-point system with MSSS matrix blocks to a single MSSS matrix system. Since the saddle-point system is indefinite, the

global preconditioner is also indefinite. Thus the MINRES method is not suitable for the preconditioned system with the global preconditioner. Here we use the IDR(s) method to solve the saddle-point system. Table 14-15 show the computational results of $\beta = 10^{-3}$ and 10^{-4} for comparison with the results of the block-diagonal preconditioner in Table 11-13. Results of different values of β for the optimal control of the convection-diffusion equation and the Poisson equation with the global preconditioner and the blocks-diagonal preconditioner can be found in appendix B.2.

Table 14: By the global preconditioner for $\beta = 10^{-3}$ and $\epsilon = 10^{-1}$

problem size	iterations	preconditioning	IDR(4)	total
3.07e+03 (4)	2	0.38	0.13	0.51
1.23e+04 (6)	2	1.16	0.24	1.40
4.92e+04 (8)	2	4.46	0.66	5.12
1.97e+05 (10)	2	18.29	2.21	20.50

Table 15: By the global preconditioner for $\beta = 10^{-4}$ and $\epsilon = 10^{-1}$

problem size	iterations	preconditioning	IDR(4)	total
3.07e+03 (4)	2	0.38	0.13	0.51
1.23e+04 (6)	2	1.15	0.24	1.39
4.92e+04 (7)	2	4.23	0.64	4.87
1.97e+05 (9)	2	17.87	2.21	20.08

Remark 4.2. *Compare the computational results of the global preconditioner in Table 14-15 with the results of the block-diagonal preconditioner in Table 11-13, it can be seen that the number of iterations is reduced significantly and independent of β for the global preconditioner. Even the global preconditioner consumes more time in preconditioning than the block-diagonal preconditioner, it needs less time in IDR(4) time and the total time is much less than that of the block-diagonal preconditioner. Numerical experiments results in the appendix for the optimal control of the Poisson equation also support that advantage of the the global preconditioner over the block-diagonal preconditioner.*

5 Preconditioning for Optimal Control of 3D Problems

As analyzed in Section 3.1, to do an LU factorization of a k-level SSS matrix, the model order reduction of $(k-1)$-level SSS matrix is needed. Since the model order reduction for 2-level and higher level SSS matrices is a big challenge, there exist no method that works well to the best knowledge of the authors, some first-step work for optimal control of 3D Poisson equation in Example 5.1 are discussed in this section.

Example 5.1. *Consider the problem of optimal control of the Poisson equation*

$$\begin{aligned} \min_{u,f} \frac{1}{2}\|u-\hat{u}\| + \frac{\beta}{2}\|f\|^2 \\ s.t. -\nabla^2 u = f \ \ in\ \Omega \\ u = u_D \ \ on\ \partial\Omega, \end{aligned} \tag{24}$$

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

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

The discretized analog of problem (24) is

$$\begin{aligned} \min_{u,\ f}\ & \frac{1}{2}\|u-\hat{u}\|^2+\beta\|f\|^2 \\ & s.t.\ Ku=Mf+d, \end{aligned} \tag{25}$$

where

$$K=\begin{bmatrix} D & -L & & & \\ -L & D & -L & & \\ & -L & D & \ddots & \\ & & \ddots & \ddots & -L \\ & & & -L & D \end{bmatrix}, \tag{26}$$

and the matrices D and L in K are 2-level SSS matrices. To get the optimal solution of Example 5.1, the type of saddle-point system (3) needs to be solved. Here we also have two types of preconditioners, one is the block-diagonal preconditioner and the other is the global preconditioner.

5.1 Block-Diagonal Preconditioners

In this subsection, we test the block-diagonal preconditioners for big and middle size of β, then the block-diagonal preconditioner $\mathcal{P}_1$ is chosen as in (6) where $\hat{K}$ is approximated by MSSS matrix computations.

To compute the LDU factorization of the matrix K with MSSS matrix computations, the Schur complement at the $k-th$ step is

$$\begin{cases} S_0=D, \\ S_{k+1}=D-LS_k^{-1}L. \end{cases} \tag{27}$$

Since D and L are 2-level SSS matrices, S_k is also a 2-level SSS matrix. During the recurrence of computing the Schur complement S_k, both the 2-level and 1-level semiseparable orders increase. Model order reduction for 2-level and 1-level SSS matrices are necessary, of which the 2-level model order reduction is still an open problem. Here we use another method to approximate the Schur complement with lower 2-level semiseparable order.

As pointed out in [37], for a symmetric positive definite matrix from discretization of PDEs with constant coefficients, all subsequent Schur complements are also symmetric positive definite and will converge to a fixed point matrix S_∞ with a fast convergence rate. In [15], Dewilde et. al. used the hierarchical partition of the matrix K and computed the Schur complement at the first $k_s(k_s \le 3)$ iteration steps. Then replace the Schur complements S_k $(k > k_s)$ with S_{k_s} to approximate the Schur complements afterwards for the preconditioning of the Poisson equation on an $8\times 8\times 8$ regular grid. Due to limited (only k_s) steps for computation of the Schur complements, the 2-level semiseparable order is bounded by a small number. Note that in [15], there are no numerical experiments to test the performance of the preconditioners for the Krylov subspace method.

In this paper, we extend the methods in [15] for hierarchical partition of the matrix to the 3-level SSS matrix partition case to compute the block-diagonal preconditioner. Here we just compute the Schur complements of the first four steps and use S_4 to approximate the Schur complements afterwards, which corresponds to the third order approximation in [15]. With the block-diagonal preconditioner $\mathcal{P}_1$ in (6) and solve the preconditioned system by the MINRES method, the computational results are shown in Table 16-17. The problem sizes 1.54e+03, 1.23e+04, 9.83e+04, and 7.86e+05 correspond to the mesh size 2^{-3}, 2^{-4}, 2^{-5}, and 2^{-6}, respectively. The maximum semiseparable order is in the brackets that follow the problem size. Here for the block-diagonal preconditioner, we test two model order reduction

algorithms and the MOR columns of Table 16-17 list the time spent in model order reduction of corresponding algorithms.

Table 16: By approximate balanced truncation for $\beta = 10^{-1}$

problem size	iterations	Preconditioning	MOR	MINRES	total
1.54e+03 (4)	4	4.83	1.83	3.15	8.03
1.23e+04 (4)	8	12.28	5.71	23.93	36.21
9.83e+04 (8)	20	38.13	22.24	263.93	302.06
7.86e+05 (8)	34	178.41	116.04	2351.70	2530.11

Table 17: By Hankel blocks approximation for $\beta = 10^{-1}$

problem size	iterations	Preconditioning	MOR	MINRES	total
1.54e+03 (4)	4	16.27	13.52	3.01	19.28
1.23e+04 (4)	8	39.76	33.50	23.85	63.61
9.83e+04 (8)	16	122.98	106.03	213.31	336.29
7.86e+05 (8)	34	551.25	490.16	2277.50	2828.75

Since we just compute the Schur complements of the first k_s steps with MSSS matrix computations, the computational complexity is less than linear. The growth rate of the time to compute the preconditioner in Table 16-17 is smaller than 8, which illustrate this property. The computational results in Table 16-17 also verify that the approximate balanced truncation is computationally cheaper than the Hankel blocks approximation and the preconditioners computed by these two model order reduction algorithms give almost the same number of iterations. This covers the results in Section 4.1.

5.2 Global Preconditioners

In the previous part, we extend the methods in [15] for symmetric positive definite by hierarchical partition to the 3-level SSS partitioned symmetric positive definite matrix. In this part, we extend this method to the 3-level SSS partitioned symmetric but indefinite matrix. For the global preconditioner, we also compute the first 4 steps of the Schur complements in the LDU factorization of the global 3-level SSS matrix. The computational results are shown in Table 18. Due to the indefiniteness of the global preconditioner, IDR(16) was chosen as the iterative solver. Compare results of the global preconditioner by the Hankel blocks approximation in Section 4.2 with that by the approximate balanced truncation in appendix A.3, the Hankel blocks approximation perform better. Thus, in this section the model order reduction algorithm for the global preconditioner is chosen as the Hankel blocks approximation.

Table 18: By the global preconditioner for $\beta = 10^{-1}$

problem size	iterations	preconditioning	IDR(16)	total
1.54e+03 (6)	15	6.89	3.81	10.70
1.23e+04 (6)	25	18.21	40.68	58.89
9.83e+04 (6)	45	119.23	863.73	982.96

Due to limited steps for computing the Schur complements, the time to compute the global preconditioner is also less than linear, which is illustrated by Table 18. Due to the indefiniteness of the sallde-point system matrix, the Schur complements in the LDU factorization is also indefinite. As pointed out in [15], the Schur complements for symmetric positive definite matrices from discretized PDEs of constant coefficients are also symmetric positive definite and have a fast rate of convergence, while the convergence of the Schur

complements for the indefinite matrix is not guaranteed. This is illustrated by the number of iterations for the global preconditioner in Table 18 is bigger than that of the block-diagonal preconditioner in Table 16-17.

Comparing the results of the block-diagonal preconditioner in Table 16-17 with the results of the global preconditioner in Table 18, we will conclude that for the optimal control of 3D problems with MSSS matrix computations, the block-diagonal preconditioner is recommended. If the model order reduction algorithm for 2- or higher- level SSS matrices are well-established, we believe that the global preconditioner will perform better than the block preconditioner for 3D problems.

6 Conclusions

In this paper, we have studied the global preconditioner and the block preconditioners for the saddle-point systems from the PDE-constrained optimization problems. By exploiting the multilevel sequentially semiseparable (MSSS) structure of the blocks of the saddle-point systems, we have constructed preconditioners and solved the preconditioned system in linear computational complexity. To compute the preconditioners with MSSS matrix computations, the approximate balanced truncation model order reduction algorithm for MSSS matrix computations has been proposed. The standard model order reduction algorithm, i.e., the Hankel blocks approximation is also studied. Numerical experiments illustrate that for the optimal control of 2D PDEs, the global preconditioner reduced the number of iterations significantly compared with the block preconditioners, while both preconditioners yield results independent of the mesh size. Moreover, the global preconditioner is independent of the regularization parameter while the block preconditioners are not. Thus, for optimal control of 2D PDEs, the global preconditioner by the Hankel blocks approximation is recommended. Since well-established model order reduction algorithm for 2- or higher- level SSS matrices is still an open problem, block preconditioners by approximate balanced truncation are preferred for the optimal control of 3D problems.

The next step of this research is to apply this preconditioning technique to the optimal control of the flow in a domain, such as optimal control of the Stokes equation and optimal control of the Navier-Stokes equation. This has a wide range of applications such as control of the wind farms to optimize the output power.

References

[1] G. Biros and O. Ghattas. Parallel Lagrange–Newton–Krylov–Schur methods for PDE-constrained optimization. part I: The Krylov–Schur solver. *SIAM Journal on Scientific Computing*, 27(2):687–713, 2005.

[2] G. Biros and O. Ghattas. 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(2):714–739, 2005.

[3] G.S. Abdoulaev, K. Ren, and A.H. Hielscher. Optical tomography as a PDE-constrained optimization problem. *Inverse Problems*, 21(5):1507–1530, 2005.

[4] L.T. Biegler, O. Ghattas, M. Heinkenschloss, D. Keyes, and B. van Bloemen Waanders. *Real-time PDE-constrained Optimization*, volume 3. Society for Industrial and Applied Mathematics, philadelphia, 2007.

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

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

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

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

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

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

[11] 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, Proc. SPIE 4791, Advanced Signal Processing Algorithms, Architectures, and Implementations XII*, volume 4791, pages 162–170. International Society for Optics and Photonics, 2002.

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

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

[14] J. Gondzio and P. Zhlobich. Multilevel quasiseparable matrices in PDE-constrained optimization. *arXiv preprint arXiv:1112.6018*, 2011.

[15] P. Dewilde, H.Y. Jiao, and S. Chandrasekaran. Model reduction in symbolically semiseparable systems with application to preconditioners for 3D sparse systems of equations. In *Characteristic Functions, Scattering Functions and Transfer Functions*, volume 197 of *Operator Theory: Advances and Applications*, pages 99–132. Birkhäser Basel, 2010.

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

[17] Y. Chahlaoui. Two efficient SVD/Krylov algorithms for model order reduction of large scale systems. *Electronic Transactions on Numerical Analysis*, 38:113–145, 2011.

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

[19] T. Rees. *Preconditioning Iterative Methods for PDE-Constrained Optimization*. PhD thesis, University of Oxford, 2010.

[20] T. Rees, H.S. Dollar, and A.J. Wathen. Optimal solvers for PDE-constrained optimization. *SIAM Journal on Scientific Computing*, 32(1):271–298, 2010.

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

[22] Y. Eidelman, I. Gohberg, and V. Olshevsky. The QR iteration method for Hermitian quasiseparable matrices of an arbitrary order. *Linear Algebra and Its Applications*, 404(0):305–324, 2005.

[23] Y. Eidelman and I. Gohberg. A modification of the Dewilde-van der Veen method for inversion of finite structured matrices. *Linear Algebra and Its Applications*, 343-344(0):419–450, 2002.

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

[25] S. Chandrasekaran, M. Gu, and T. Pals. A fast ULV decomposition solver for hierarchically semiseparable representations. *SIAM Journal on Matrix Analysis and Applications*, 28(3):603–622, 2006.

[26] A.J. van der Veen. *Time-Varying System Theory and Computational Modeling.* PhD thesis, Delft University of Technology, 1993.

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

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

[29] J.K. Rice and M. Verhaegen. Distributed control: A sequentially semi-separable approach for spatially heterogeneous linear systems. *Automatic Control, IEEE Transactions on*, 54(6):1270–1283, 2009.

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

[31] Y. Chahlaoui and P. Van Dooren. Estimating gramians of large-scale time-varying systems. In *IFAC World Congress*, volume 15, pages 540–545, 2002.

[32] H. Sandberg and A. Rantzer. Balanced truncation of linear time-varying systems. *Automatic Control, IEEE Transactions on*, 49(2):217–229, 2004.

[33] A.C. Antoulas, D.C. Sorensen, and Y. Zhou. On the decay rate of Hankel singular values and related issues. *Systems & Control Letters*, 46(5):323–342, 2002.

[34] T. Penzl. Eigenvalue decay bounds for solutions of Lyapunov equations: the symmetric case. *Systems & Control Letters*, 40(2):139–144, 2000.

[35] P. Benner. Solving large-scale control problems. *Control Systems Magazine, IEEE*, 24(1):44–59, 2004.

[36] Y. Chahlaoui and P. Van Dooren. Model reduction of time-varying systems. In P. Benner, D.C. Sorensen, and V. Mehrmann, editors, *Dimension Reduction of Large-Scale Systems*, volume 45 of *Lecture Notes in Computational Science and Engineering*, pages 131–148. Springer Berlin Heidelberg, 2005.

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

Appendix

A Comparison of Two Model Order Reduction Algorithms

A.1 Block-Diagonal Preconditioner

Consider the problem of optimal control of the Poisson equation in Example A.1,

Example A.1. *[14] Let* $\Omega = [0,\ 1]^2$ *and consider the problem*

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

where $\Gamma_N = \{x = 0, 0 \le y \le 1\}$ *and* $\Gamma_D = \partial\Omega \backslash \Gamma_N$, $\overrightarrow{n}$ *is the normal vector on the bounds that point outwards,* $\hat{u} = 0$ *is the prescribed system state,* $u_N = sin(2\pi y)$ *and*

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

The computational results for optimal control of the Poisson equation by MINRES method with the preconditioner $\mathcal{P}_1$ by the approximate balanced truncation Algorithm 2-3 and the Hankel blocks approximation Algorithm 4 for different values of β are shown in Table 19 - 24.

Table 19: By approximate balanced truncation for $\beta = 10^{-1}$

problem size	iterations	preconditioning	MINRES	total
3.07e+03 (3)	6	0.20	0.61	0.84
1.23e+04 (3)	8	0.57	1.76	2.33
4.92e+04 (5)	8	2.09	5.06	7.15
1.97e+05 (6)	8	8.92	18.90	27.82

Table 20: By Hankel blocks approximation for $\beta = 10^{-1}$

problem size	iterations	preconditioning	MINRES	total
3.07e+03 (3)	6	0.46	0.59	1.03
1.23e+04 (3)	8	0.69	1.79	2.48
4.92e+04 (5)	6	2.83	4.20	7.03
1.97e+05 (6)	8	10.81	18.79	29.60

Table 21: By approximate balanced truncation for $\beta = 10^{-2}$

problem size	iterations	preconditioning	MINRES	total
3.07e+03 (4)	8	0.21	0.78	0.99
1.23e+04 (4)	8	0.72	2.00	2.72
4.92e+04 (5)	8	2.53	6.28	8.81
1.97e+05 (6)	10	9.53	25.12	34.65

Table 22: By Hankel blocks approximation for $\beta = 10^{-2}$

problem size	iterations	preconditioning	MINRES	total
3.07e+03 (4)	8	0.31	0.83	1.14
1.23e+04 (4)	8	0.98	2.07	3.05
4.92e+04 (5)	6	3.49	4.67	8.16
1.97e+05 (6)	8	14.67	20.31	34.98

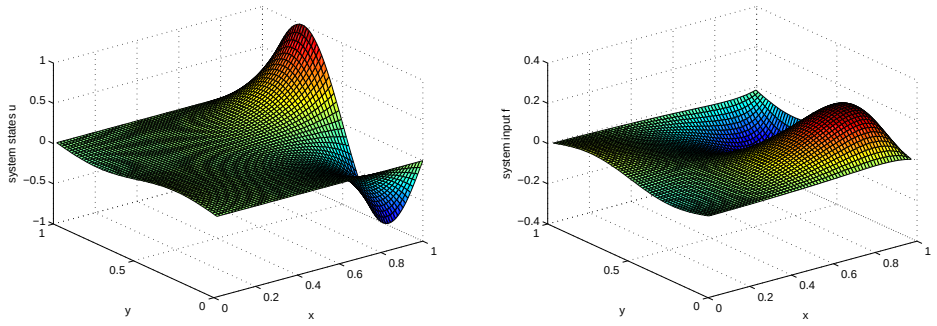
Table 23: By approximate balanced truncation for $\beta = 10^{-3}$

problem size	iterations	preconditioning	MINRES	total
3.07e+03 (4)	12	0.23	1.14	1.37
1.23e+04 (4)	12	0.67	2.92	3.59
4.92e+04 (6)	12	2.75	7.89	10.64
1.97e+05 (7)	12	11.50	28.92	40.42

Table 24: By Hankel blocks approximation for $\beta = 10^{-3}$

problem size	iterations	Preconditioning	MINRES	total
3.07e+03 (4)	12	0.34	1.23	1.57
1.23e+04 (4)	12	0.76	2.97	3.73
4.92e+04 (6)	12	3.68	8.59	12.27
1.97e+05 (7)	12	14.43	28.94	43.37

The optimal solution of the system states and input for $\beta = 10^{-2}$ and $h = 2^{-6}$ are shown in Figure 4(a) and 4(b).



(a) Optimal system states u.

(b) Optimal system input f.

Figure 4: Solution of the system states and input when $\beta = 10^{-2}$ and $h = 2^{-6}$.

Remark A.1. *Table 19 - 24 show that the number of iterations for the block-diagonal preconditioner with approximate balanced truncation and the Hankel blocks approximation are virtually independent of the mesh size. For the same semiseparable order setup, computation of the preconditioner with approximate balanced truncation is computationally cheaper than preconditioning with the Hankel blocks approximation, while both algorithms have linear computational complexity with respect to the problem size. The time of the MINRES method is also linear with respect to the problem size for both model order reduction algorithms.*

Remark A.2. *As shown in remark 3.20, the Hankel blocks approximation Algorithm 4 yields a more accurate approximation than the approximate balanced truncation Algorithm 2-3 while*

both methods return an approximate of satisfied accuracy. This is illustrated in Table 19-24. For the same problem size, the number of iterations is very limited while the average number of iterations of the Hankel blocks approximation is equal to or a little smaller than that of the approximate balanced truncation.

A.2 Block Lower-Triangular Preconditioner

This part gives the performance of the block lower-triangular preconditioner for optimal control of the convection-diffusion equation in Example 4.1. Take the block lower-triangular preconditioner $\mathcal{P}_2$ in (5) by the approximate balanced truncation Algorithm 2-3 and the Hankel blocks approximation Algorithm 4, solve the unsymmetric preconditioned system with IDR(s) method. The computational results are shown in Table 25 - 34.

Table 25: By approximate balanced truncation for $\beta = 10^{-1}$, $\epsilon = 10^{-1}$

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 26: By Hankel blocks approximation for $\beta = 10^{-1}$, $\epsilon = 10^{-1}$

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 27: By approximate balanced truncation for $\beta = 10^{-1}$, $\epsilon = 10^{-2}$

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 28: By Hankel blocks approximation for $\beta = 10^{-1}$, $\epsilon = 10^{-2}$

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 29: By approximate balanced truncation for $\beta = 10^{-1}$, $\epsilon = 10^{-2}$

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 30: By Hankel blocks approximation for $\beta = 10^{-1}$, $\epsilon = 10^{-2}$

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 31: By approximate balanced truncation for $\beta = 10^{-2}$, $\epsilon = 10^{-1}$

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 32: By Hankel blocks approximation for $\beta = 10^{-2}$, $\epsilon = 10^{-1}$

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 33: By approximate balanced truncation for $\beta = 10^{-2}$, $\epsilon = 10^{-2}$

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 34: By Hankel blocks approximation for $\beta = 10^{-2}$, $\epsilon = 10^{-2}$

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

Remark A.3. *From Table 25-34, we can see that for the fixed values of β and ϵ, the number of iterations is very limited, almost constant and independent of the mesh size. Meanwhile, both preconditioners have linear computational complexity, which is illustrated by the preconditioning time columns. The preconditioned system can also be solved in linear complexity, which is verified by the IDR(s) time columns.*

Remark A.4. *From the preconditioning columns of Table 25-34 for the same experiment settings, we can see that the approximate balanced truncation method for SSS matrices is computationally cheaper than the Hankel blocks approximation method.*

Remark A.5. *Compare the computational results of the block-diagonal preconditioner $\mathcal{P}_1$ and MINRES in Table 3-10 with that of the block lower-triangular preconditioner $\mathcal{P}_2$ and*

IDR(s) in Table 25-34, we can see that both preconditioners are comparable. For the same settings of β and ϵ, the semiseparable order needs to be set bigger for the IDR(s) method with $\mathcal{P}_2$ than the MINRES method with $\mathcal{P}_1$. This makes the preconditioning time and the iterative solution time of $\mathcal{P}_2$ bigger than that of $\mathcal{P}_1$.

A.3 Global Preconditioner

For the global preconditioner by the approximate balanced truncation, the computational results for the optimal control of the Poisson equation is shown in Table 35-36.

Table 35: By approximate balanced truncation for $\beta = 10^{-1}$

problem size	iterations	preconditioning	IDR(4)	total
3.07e+03 (10)	4	0.48	0.19	0.67
1.23e+04 (13)	4	1.69	0.43	2.12
4.92e+04 (16)	4	6.39	1.34	7.73
1.97e+05 (20)	6	29.34	10.28	39.62

Table 36: By approximate balanced truncation for $\beta = 10^{-2}$

problem size	iterations	preconditioning	IDR(4)	total
3.07e+03 (11)	3	0.50	0.16	0.66
1.23e+04 (14)	4	1.75	0.43	2.18
4.92e+04 (16)	3	5.96	1.52	7.48
1.97e+05 (22)	4	31.84	8.08	39.92

Due to the ill-condition of the saddle-point system, it is difficult to compute a good approximation of the indefinite saddle-point system. To get a good approximation of the saddle-point system with MSSS matrix computations, bigger semiseparable order is needed. Based on remark 3.20, the approximate balanced truncation yields a reduced SSS matrix less accurate than the Hankel blocks approximation, bigger semiseparable order is needed for the approximate balanced truncation than the Hankel blocks approximation. This is illustrated by the results in Table 45-46 for the Hankel blocks approximation with the results in Table 35-36 for the approximate balanced truncation. The increase of the semiseparable order leads to the increase of computational complexity. This makes the global preconditioner by the approximate balanced truncation more computationally expensive than the global preconditioner by the Hankel blocks approximation. Here we do not compare the performance of the two different model order reduction algorithms for other experiment setup.

B Comparison of Preconditioners

B.1 Block-Diagonal Preconditioner

In this part, the performance of the block-diagonal preconditioner for small size of β for the optimal control of the Poisson equation and the convection-diffusion equation is studied. Table 37-40 show the results of the block-diagonal preconditioner $\mathcal{P}_1$ in (6) for the optimal control of the Poisson equation.

Table 37: With $\mathcal{P}_1$ in (6) by approximate balanced truncation for $\beta = 10^{-5}$

problem size	iterations	preconditioning	MINRES	total
3.07e+03 (5)	42	0.28	2.51	2.79
1.23e+04 (5)	42	0.76	6.52	7.28
4.92e+04 (5)	42	2.48	21.23	23.71
1.97e+05 (5)	42	11.13	83.34	94.47

Table 38: With $\mathcal{P}_1$ in (6) by Hankel blocks approximation for $\beta = 10^{-5}$

problem size	iterations	preconditioning	MINRES	total
3.07e+03 (5)	42	0.28	2.45	2.73
1.23e+04 (5)	42	0.81	6.57	7.38
4.92e+04 (5)	42	3.48	21.28	24.76
1.97e+05 (5)	42	12.43	84.75	97.18

Table 39: With $\mathcal{P}_1$ in (6) by approximate balanced truncation for $\beta = 10^{-6}$

problem size	iterations	preconditioning	MINRES	total
3.07e+03 (5)	100	0.27	5.31	5.58
1.23e+04 (5)	96	0.87	14.71	15.58
4.92e+04 (5)	95	2.87	49.32	52.19
1.97e+05 (5)	90	11.27	195.47	206.74

Table 40: With $\mathcal{P}_1$ in (6) by Hankel blocks approximation for $\beta = 10^{-6}$

problem size	iterations	preconditioning	MINRES	total
3.07e+03 (5)	100	0.27	5.31	5.58
1.23e+04 (5)	96	0.96	14.60	15.56
4.92e+04 (5)	95	3.60	49.68	53.28
1.97e+05 (5)	90	12.33	195.35	207.68

From Table 37-40, we can see that with the decrease of β, the number of iterations is constant with the mesh size but increases dramatically. As introduced in [19], for "smaller" β ($\beta \leq 10^{-5}$), the block-diagonal conditioner could be chosen as $\mathcal{P}_1$ in (7). With this preconditioner, the computational results are shown in Table 41-42. The maximum number of iterations is set to 100.

Table 41: With $\mathcal{P}_1$ in (7) by Hankel blocks approximation for $\beta = 10^{-5}$

problem size	iterations	preconditioning	MINRES	convergence
3.07e+03 (5)	100	0.33	6.62	no convergence
1.23e+04 (5)	100	1.08	14.66	no convergence
4.92e+04 (5)	100	3.93	38.04	no convergence
1.97e+05 (5)	100	15.65	118.32	no convergence

Table 42: With $\mathcal{P}_1$ in (7) by Hankel blocks approximation for $\beta = 10^{-6}$

problem size	iterations	preconditioning	MINRES	convergence
3.07e+03 (5)	100	0.33	6.52	no convergence
1.23e+04 (5)	100	1.07	14.57	no convergence
4.92e+04 (5)	100	3.93	39.25	no convergence
1.97e+05 (5)	100	15.14	118.92	no convergence

Remark B.1. *As shown in Table 41-42, the block diagonal preconditioner $\mathcal{P}_1$ in (7) does not work well for the smaller β. This preconditioner cannot yield the satisfied solution of the saddle-point system within the maximum number of iterations.*

For small size of β of the optimal control of the convection-diffusion equation, the computational results of the block-diagonal preconditioner $\mathcal{P}_1$ in (6) by the approximate balanced truncation are shown in Table 43-44.

Table 43: With $\mathcal{P}_1$ in (6) by approximate balanced truncation for $\beta = 10^{-3}$, $\epsilon = 10^{-1}$

problem size	iterations	preconditioning)	MINRES	total
3.07e+03 (3)	34	0.34	2.93	3.27
1.23e+04 (3)	34	0.94	7.31	8.25
4.92e+04 (3)	34	2.34	19.38	21.72
1.97e+05 (5)	34	10.39	61.12	71.51

Table 44: With $\mathcal{P}_1$ in (6) by approximate balanced truncation for $\beta = 10^{-4}$, $\epsilon = 10^{-1}$

problem size	iterations	preconditioning	MINRES	total
3.07e+03 (3)	82	0.35	5.02	5.37
1.23e+04 (3)	82	0.91	11.78	12.69
4.92e+04 (3)	80	2.67	33.98	36.65
1.97e+05 (5)	80	10.81	132.98	143.79

B.2 Global Preconditioners

For optimal control of the Poisson equation, the computational results of the global preconditioner by Hankel blocks approximation are shown in Table 45-49.

Table 45: Global Preconditioner for $\beta = 10^{-1}$

problem size	iterations	preconditioning	IDR(4)	total
3.07e+03 (4)	2	0.39	0.13	0.52
1.23e+04 (4)	3	1.13	0.34	1.47
4.92e+04 (6)	3	3.98	0.96	4.94
1.97e+05 (6)	3	14.39	3.11	17.50

Table 46: Global Preconditioner for $\beta = 10^{-2}$

problem size	iterations	preconditioning	IDR(4)	total
3.07e+03 (4)	3	0.38	0.15	0.52
1.23e+04 (4)	3	1.08	0.31	1.39
4.92e+04 (6)	3	3.87	0.89	4.76
1.97e+05 (6)	3	14.58	3.13	17.71

Table 47: Global Preconditioner for $\beta = 10^{-3}$

problem size	iterations	preconditioning	IDR(4)	total
3.07e+03 (4)	3	0.38	0.15	0.52
1.23e+04 (5)	3	1.12	0.31	1.43
4.92e+04 (7)	2	4.19	0.64	4.76
1.97e+05 (7)	4	15.95	4.11	20.06

Table 48: Global Preconditioner for $\beta = 10^{-5}$

problem size	iterations	preconditioning	IDR(4)	total
3.07e+03 (5)	2	0.39	0.12	0.51
1.23e+04 (7)	3	1.20	0.31	1.51
4.92e+04 (7)	3	4.12	0.89	5.01
1.97e+05 (9)	4	15.86	4.44	20.30

Table 49: Global Preconditioner for $\beta = 10^{-6}$

problem size	iterations	preconditioning	IDR(4)	total
3.07e+03 (4)	3	0.37	0.15	0.52
1.23e+04 (6)	2	1.12	0.33	1.45
4.92e+04 (8)	3	4.20	1.64	5.84
1.97e+05 (10)	3	17.94	6.63	24.57

Remark B.2. *Table 45-49 show that the global preconditioner has linear computational complexity that makes time to compute the preconditioner and IDR(4) time scale linearly with the problem size. Furthermore, the performance of the global preconditioner is mesh size independent.*

Remark B.3. *As shown in Table 45-49, the global preconditioner is independent of the regularization parameter β. For different β, the number of iterations is independent of β. Compared with the results for the block-diagonal preconditioner, the global preconditioner is computationally cheaper than the block-diagonal preconditioner.*

Remark B.4. *As the condition number of the saddle-point system is proportional to $\frac{1}{\beta}$, with the decrease of β, for the same problem size, the saddle-point system becomes more ill-conditioned. This makes it much more difficult to compute an accurate approximate LU factorization of the global saddle-point system. This is illustrated by the slightly increase of the maximum semiseparable order in this factorization for the same problem size with decrease of β in Table 45-49. Due to this slightly increase of the semiseparable order, the time to compute the preconditioner and iterative solution method also increase slightly, but they are still linear with the problem size.*

With the global preconditioner, the computational results for optimal control of the convection-diffusion equation for big β are shown in Table 50-53.

Table 50: Global Preconditioner for $\beta = 10^{-1}$ and $\epsilon = 10^{-1}$

problem size	iterations	preconditioning	IDR(4)	total
3.07e+03 (4)	2	0.38	0.15	0.53
1.23e+04 (4)	2	1.11	0.23	1.34
4.92e+04 (6)	3	3.92	0.91	4.83
1.97e+05 (6)	3	14.84	3.15	17.99

Table 51: Global Preconditioner for $\beta = 10^{-2}$ and $\epsilon = 10^{-1}$

problem size	iterations	preconditioning	IDR(4)	total
3.07e+03 (4)	2	0.38	0.13	0.51
1.23e+04 (4)	3	1.11	0.32	1.43
4.92e+04 (6)	2	3.92	0.63	4.55
1.97e+05 (6)	3	15.11	3.12	18.23

Table 52: Global Preconditioner for $\beta = 10^{-1}$ and $\epsilon = 10^{-2}$

problem size	iterations	preconditioning	IDR(4)	total
3.07e+03 (4)	1	0.38	0.09	0.47
1.23e+04 (4)	1	1.11	0.15	1.26
4.92e+04 (6)	1	3.89	0.36	4.25
1.97e+05 (6)	2	14.77	2.14	16.91

Table 53: Global Preconditioner for $\beta = 10^{-2}$ and $\epsilon = 10^{-2}$

problem size	iterations	preconditioning	IDR(4)	total
3.07e+03 (4)	1	0.38	0.09	0.47
1.23e+04 (4)	1	1.11	0.15	1.26
4.92e+04 (6)	1	3.95	0.36	4.31
1.97e+05 (6)	2	14.92	2.14	17.06

Remark B.5. *In Table 52 and 53, with a small maximum semiseparable setup for the problems in the first three rows, the global preconditioner is already accurate enough that can be performed as a direct solver.*

Remark B.6. *Due to the condition number of the saddle-point system is proportional to* $\frac{1}{\beta}$*, the saddle-point system becomes ill-conditioned with the decrease of* β*. This makes it difficult to compute an accurate approximation close to the saddle-point system. Thus the maximum semiseparable should be increased slightly. The slightly increase of the maximum semiseparable order does not change the linear computational complexity. This is illustrated in Table 50-53.*

Remark B.7. *According to Table 50-53, the number of iterations for the global preconditioner is independent of the regularization parameter* β*, while for the block-diagonal preconditioner, this property does not hold.*