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

Technical Report · May 2013

DOI: 10.13140/2.1.2977.1524

Yue Qiu
Max Planck Institute for Dynamics of Complex Technical Systems
23 PUBLICATIONS 58 CITATIONS
SEE PROFILE

M.B. van Gijzen
Delft University of Technology
112 PUBLICATIONS 1,162 CITATIONS
SEE PROFILE

J. W. van Wingerden
Delft University of Technology
199 PUBLICATIONS 3,387 CITATIONS
SEE PROFILE

Michel Verhaegen
Delft University of Technology
609 PUBLICATIONS 12,408 CITATIONS
SEE PROFILE

DELFT UNIVERSITY OF TECHNOLOGY

REPORT 13-04

Efficient Preconditioners for PDE-Constrained Optimization Problems with a Multilevel 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: December 1st, 2014

Abstract

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

1 Introduction

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 obtained 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 dimension reduction (`IDR(s)`) [7] methods. Their performance 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. 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

[*] 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`.

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, introduced by Dewilde et al. in [13] generalize the semiseparable matrices. Multilevel sequentially semiseparable matrices generalize the sequentially semiseparable matrices to the multi-dimensional cases. Systems that arise from the discretization of 1D partial differential equations typically have an SSS structure. Discretization of higher dimensional (2D or 3D) partial differential equations give rise to matrices that have an MSSS structure [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 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. As a result, the LU factorization is not of linear computational complexity. Model order reduction can be used to reduce the rank of the off-diagonal blocks, which yields an inexact LU decomposition of an MSSS matrix that can be used as a preconditioner.

In [15], Gondzio et al. first introduced the MSSS matrix computations for preconditioning of PDE-constrained optimization problems. They exploited the MSSS matrix structure of the blocks of the saddle-point system and performed an LU factorization for MSSS matrices to approximate the Schur complement of the saddle-point system. With this approximated Schur complement as a preconditioner, conjugate gradient iterations were performed to solve the saddle-point system block-by-block. As aforementioned, the model order reduction plays a vital role in obtaining a linear computational complexity of the LU factorization for MSSS matrices. In [15], Gondzio et al. used a standard model order reduction algorithm [13, 16] to reduce the computational complexity.

This paper extends [15] 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. This new model order reduction algorithm is motivated by the work in [17, 18]. In [17], the approximate balanced truncation was addressed for the model order reduction of linear time invariant (LTI) systems, while in [18] the recursive low-rank approximation was performed to compute the approximation of the gramians of LTV systems. In this paper, we use the low-rank approximation method in [18] and the approximate balanced truncation in [17] for the model order reduction for the SSS matrices. Compared with the model order reduction algorithms discussed in [13, 16], the approximate balanced truncation method for SSS matrices in this paper is computationally cheaper. 2) With these model order reduction algorithms, we can compute an inexact LU factorization for the MSSS matrix blocks of the saddle-point system in linear computational complexity ($\mathcal{O}(N)$). This yields a block preconditioner for the saddle-point systems. Exploiting the block structure of the saddle-point system is the standard preconditioning technique, which is described in [19]. However, only the single preconditioner for the last block of the saddle-point system is studied in [15]. 3) By permuting the blocks of the saddle-point system, we can also compute an inexact LU factorization of the global system with MSSS matrix computations in linear computational complexity. This gives a global MSSS preconditioner and this novel MSSS preconditioner gives mesh size and regularization parameter independent convergence. This is a big advantage over the block MSSS 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) Moreover, we extend these preconditioning techniques to the 3D cases.

Note that the convergence of the block preconditioners depends on the regularization parameter β for the PDE-constrained optimization problems [20]. For small β, block preconditioners do not give satisfied performance. Since all the blocks of the saddle-point matrix are MSSS matrices, we can permute the saddle-point matrix into a single MSSS matrix. Then we can compute an approximate LU factorization for the permuted saddle-point system using MSSS matrix computations in linear computational complexity. We call this approximate factorization for the permuted global matrix the global MSSS preconditioner. Block preconditioners often neglect the regularization term $\frac{1}{2\beta}M$, the convergence is often poor for small enough regularization parameter β. For global MSSS preconditioner, it is not necessary to neglect the regularization term, as a result the global MSSS preconditioner gives β independent convergence as well as the mesh size independent convergence. Numerical experiments verify this statement.

The outline of this paper is as follows: Section 2 formulates a distributed optimal control problem constrained by PDEs. This problem yields a linear saddle-point system. In Section 3, we use some definitions and algorithms for MSSS matrices and introduce the MSSS preconditioning technique. The new model order reduction algorithm for SSS matrices is also described. With MSSS matrix computations, we propose two types of preconditioners for saddle-point problem: the global MSSS preconditioner, and the block-diagonal MSSS preconditioner. In Section 4, we use the distributed optimal control of the convection-diffusion equation to illustrate the performance of these two preconditioners and the new model order reduction algorithm. Section 5 presents numerical experiments on 3D problems, we apply the global MSSS preconditioner and block-diagonal MSSS preconditioner to the optimal control of 3D Poisson equation. Section 6 draws the conclusions and describes the future work.

A companion technical report [21] is also available online and studies a wide class of PDE-constrained optimization problems. It contains more numerical experiments to illustrate the performance of this preconditioning technique. In [22], we extend our preconditioning technique to the computational fluid dynamics (CFD) problems and evaluate their performance on CFD benchmark problems using the Incompressible Flow and Iterative Solver Software (IFISS) [23].

2 Problem Formulation

2.1 PDE-Constrained Optimization Problem

Consider the following PDE-constrained optimization problem described by

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

where $\mathcal{L}$ is an operator, u is the system state, f is the system input and $\hat{u}$ is the desired state of the system, Ω is the domain and Γ_D is the corresponding bound, 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 the same solution while different answers are reached for the problem of optimal control of the convection-diffusion equation [20]. Since our focus is on preconditioning for such problems, 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,

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

where $K = [K_{i,j}] \in \mathbb{R}^{N\times N}$ is the stiffness matrix, $M = [M_{i,j}] \in \mathbb{R}^{N\times N}$, $M_{ij} = \displaystyle\int_\Omega \phi_i\phi_j d\Omega$ is the mass matrix and is symmetric positive definite, $b = [b_i] \in \mathbb{R}^N$, $b_i = \displaystyle\int_\Omega \hat{u}_i\phi_i d\Omega$, $c \in \mathbb{R}$, $c = \displaystyle\int_\Omega \hat{u}^2 d\Omega$, $d = [d_i] \in \mathbb{R}^N$, $d_i = -\displaystyle\sum_{j=N+1}^{N+\partial N} u_j \int_\Omega \nabla\phi_j\cdot\nabla\phi_i d\Omega$. The ϕ_i $(i = 1,\ 2,\ \dots\ N)$ and ϕ_j $(j = 1,\ 2,\ \dots\ N,\ N+1,\ \dots\ N+\partial N)$ 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

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

This yields the linear system

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

The system (3) is of the saddle-point system type [19], i.e., the system matrix $\mathcal{A}$ is symmetric and indefinite. It has the following structure

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

where $A \in \mathbb{R}^{n\times n}$ is symmetric positive definite, $B \in \mathbb{R}^{m\times n}$ has full rank. A relaxation condition is that A is symmetric positive semi-definite and positive definite on the null space of B , details for this condition can be found in [24].

The system matrix of the saddle-point system (3) is large and sparse. Preconditioned Krylov subspace methods, such as `MINRES` [6] and `IDR(s)` [7], are quite efficient for solving such systems.

2.2 Preconditioning of Saddle-Point Systems

The performance of iterative solution methods highly depends on the choice of the preconditioners [25]. For numerical methods to solve saddle-point system (3) and the construction of preconditioners, we refer to [19, 24] for an extensive survey. In this paper, we study two types of preconditioners. The first exploits the MSSS structure of the blocks of the saddle-point system, whereas the second type exploits the global MSSS structure of the addle-point system.

2.2.1 Block Preconditioners

Recall from (4), if A is nonsingular, then $\mathcal{A}$ admits the following LDL^T factorization given by

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

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

The most difficult part for this factorization is to compute the Schur complement S because computing the inverse of a large sparse matrix is expensive both in time and memory. Meanwhile, solving the system $Sx = b$ is also expensive since S is a large and full matrix. Note that all the blocks of (3) have a structure that is called multilevel sequentially semiseparable (MSSS), which will be introduced in the later section. Then the Schur complement S also has the MSSS structure but with a bigger semiseparable order. If we exploit the MSSS structure of (3), we can compute S in linear computational complexity.

In this paper, we first study the block-diagonal preconditioner $\mathcal{P}_1$ for the saddle-point system (3), where

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

where $\hat{M}$ is an approximation of the mass matrix M and $\hat{S}$ is an approximation of the Schur complement S. 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` computes the solution of the preconditioned system using at most three iterations.

To approximate $S = -\left(\frac{1}{2\beta}M + KM^{-1}K^T\right)$, $\hat{S} = -KM^{-1}K^T$ can be used for big to middle range of β while $\hat{S} = -\frac{1}{2\beta}M$ could be chosen for small β [20]. The block lower-triangular preconditioner $\mathcal{P}_2$, which has the following form

$$\mathcal{P}_2 = \begin{bmatrix} 2\beta\hat{M} & & \\ 0 & \hat{M} & \\ -M & K & \hat{S} \end{bmatrix}, \tag{6}$$

is studied in the technical report [21] and the performance comparison with the block-diagonal preconditioner $\mathcal{P}_1$ is also discussed.

2.2.2 Global Preconditioners

Since all the blocks of the saddle-point system (3) have the MSSS structure, there exists a permutation matrix Ψ that permutes the saddle-point matrix with MSSS blocks into a single MSSS matrix. This gives

$$\tilde{\mathcal{A}}\tilde{x} = \tilde{g}, \tag{7}$$

where $\tilde{\mathcal{A}} = \Psi\mathcal{A}\Psi^T$, $\tilde{x} = \Psi x$, and $\tilde{g} = \Psi g$ are permutations of $\mathcal{A}$, $\begin{bmatrix} f^T & u^T & \lambda^T \end{bmatrix}^T$, and $\begin{bmatrix} 0^T & b^T & d^T \end{bmatrix}^T$ in (3), respectively. This permutation will be introduced in the next section. After this permutation, the system matrix $\tilde{\mathcal{A}}$ is an MSSS matrix. We can compute an inexact LU factorization of $\tilde{\mathcal{A}}$ in linear computational complexity using MSSS matrix computations. This gives,

$$\tilde{\mathcal{A}} \approx \tilde{L}\tilde{U}, \tag{8}$$

which can be used as a preconditioner. We call this factorization (8) the global preconditioner. Since no information of β is neglected during the permutation and factorization, the global preconditioner gives β-independent convergence, while this property for the standard block preconditioners $\mathcal{P}_1$ in (5) or $\mathcal{P}_2$ in (6) do not hold. This is a big advantage of the global preconditioner over the standard block preconditioners. Numerical examples in Section 4 state this advantage.

3 Preconditioning Using Multilevel 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 Multilevel Sequentially Semiseparable Matrices

The generators representation of the sequentially semiseparable matrices are defined by Definition 3.1.

Definition 3.1 ([26]). *Let A be an $N \times N$ matrix with SSS matrix structure and let n positive integers $m_1,\ m_2,\ \cdots\ m_n$ satisfy $N = m_1 + m_2 + \cdots + m_n$ such that A can be written in the* *following block-partitioned form*

$$A_{ij} = \begin{cases} U_i W_{i+1} \cdots W_{j-1} V_j^T, & \text{if} \quad i < j; \\ D_i, & \text{if} \quad i = j; \\ P_i R_{i-1} \cdots R_{j+1} Q_j^T, & \text{if} \quad i > j. \end{cases} \tag{9}$$

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

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 defined in Definition 3.1, A can be denoted by

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

Table 1: Generators 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	$m_i \times k_i$	$k_{i-1} \times k_i$	$m_i \times k_{i-1}$	$m_i \times m_i$	$m_i \times l_i$	$l_{i-1} \times l_i$	$m_i \times l_{i+1}$

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

Remark 3.2. *For an SSS matrix, only its generators are stored. If l_i and k_i are bounded by a small constant. Then the memory consumption for storing such matrix is linear with respect to the matrix size. This property is also introduced in [26].*

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

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

With the generators representation of SSS matrices, basic operations such as addition, multiplication and inversion are closed under the SSS matrix structure and can be performed in linear computational complexity. Moreover, decompositions/factorizations such as the QR factorization [27, 28], the LU decomposition [12, 15], and the ULV decomposition [29] can also be computed in linear computational complexity and in a structure preserving way.

Similar to Definition 3.1 for SSS matrices, the generators representation for MSSS matrices, specifically the k-level SSS matrices, is defined in 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.*

Most operations for the SSS matrices can be extended to the MSSS matrices, which yields linear computational complexity for MSSS matrices. MSSS matrices have many applications, one of them is the discretized partial differential equations (PDEs) [30].

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

Lemma 3.1 explains how to permute a matrix with SSS blocks into a single SSS matrix. This property can be extended to matrices with MSSS blocks. This enables us to compute an LU factorization of the global saddle point matrix by using MSSS matrix computations in linear computational complexity.

Lemma 3.1 ([31])**.** *Let A, B, C and D be SSS matrices with the following generators representations*

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

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

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

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

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

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

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

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

$$\Psi = \begin{bmatrix} \begin{bmatrix} I_{m^a} \\ 0 \end{bmatrix} \otimes I_n & \begin{bmatrix} 0 \\ I_{m^d} \end{bmatrix} \otimes I_n \end{bmatrix}, \tag{11}$$

where $\otimes$ denotes the Kronecker product and I is the identify matrix with a proper size.

With the permutation matrix Ψ given by (11), the permuted matrix is given by

$$\mathcal{T} = \Psi \begin{bmatrix} A & B \\ C & D \end{bmatrix} \Psi^T. \tag{12}$$

It is not difficult to verify that the matrix $\mathcal{T}$ is an SSS matrix and its generators are given in Lemma 3.1.

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

$$\Psi = \begin{bmatrix} \text{blkdiag}\left(\left\{\begin{bmatrix} I_{m_i^a} \\ 0 \end{bmatrix}\right\}\right) & \text{blkdiag}\left(\left\{\begin{bmatrix} 0 \\ I_{m_i^d} \end{bmatrix}\right\}\right) \end{bmatrix}, \tag{13}$$

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

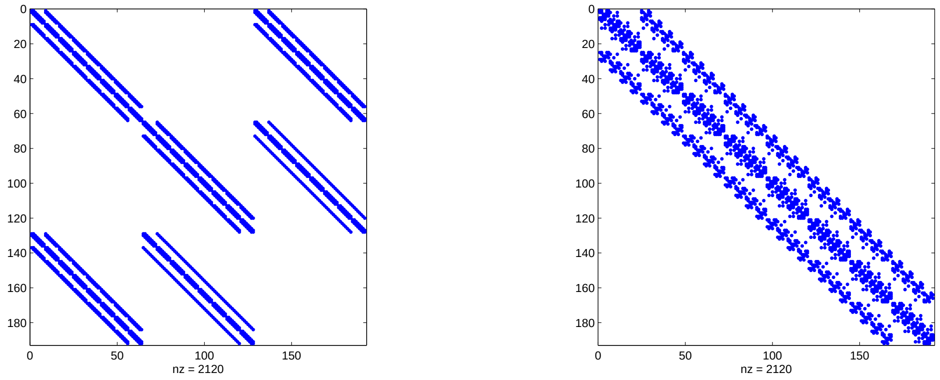
With the permutation matrix Ψ in (13), it is not difficult to show that the permuted matrix $\mathcal{T}$ given by (12) is an SSS matrix and its generators are given in Lemma 3.1. □

Remark 3.3. *To one matrix with SSS blocks, one can apply Lemma 3.1 to permute it into a single SSS matrix by using a permutation matrix Ψ. However, this permutation matrix is not explicitly multiplied on both sides of the matrix to be permuted. The generators of the permuted matrix are combinations of the generators of its SSS blocks. This is illustrated by the generators representation of the permuted matrix in Lemma 3.1. Such permutations are cheaper to compute due to the fact that there is no matrix-matrix multiplication.*

Remark 3.4. *Lemma 3.1 is for a 2×2 block matrix, but it generalizes the case of matrices with different numbers of blocks.*

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 permuted into a single $(k-1)$-level SSS matrix by applying Lemma 3.1 recursively.*

For the saddle-point system (3) derived from the optimal control of the convection-diffusion equation in 2D, discretizing using the Q_1 finite element method yields a saddle-point system that has MSSS (2-level SSS) matrix blocks. The structure of the saddle-point matrix before and after permutation for mesh size $h = 2^{-3}$ are shown in Fig. 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 Multilevel Sequentially Semiseparable Preconditioners

The most important part of the PDE-constrained optimization problem is to solve a linear system with MSSS structure in linear computational complexity. In the following part, we first introduce the LU factorization of MSSS matrices and then give a new model order reduction algorithm for SSS matrices, which is necessary in computing the LU factorization in linear computational complexity. For comparison, the conventional model order reduction algorithm [13] 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 Eidelman et al. studied this kind of structured matrices independently, sequentially semiseparable matrices named in [26] are called quasiseparable matrices in [32]. In this paper, 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*

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

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

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

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 quasiseparable order of A, respectively.

Definition 3.4 ([31])**.** *The SSS matrix A with lower and upper semiseparable order r^l and r^u is called block $(r^l,\ r^u)$ semiseparable.*

The semiseparable order for 1-level SSS matrices defined in Definition 3.3 can be directly extended to the multilevel cases, which leads to Definition 3.5.

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

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

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

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

The numbers $u_k (k = 1, 2, \ \cdots, N-1)$ are called the k-level upper order numbers of the k-level SSS 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.*

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

Lemma 3.2 ([12, 15])**.** *Let A be a strongly regular $N \times N$ block k-level sequentially semiseparable matrix of k-level block $(r^l, \ r^u)$ semiseparable and denoted by its generators representation $A = \mathcal{MSSS}(P_s, \ R_s, \ Q_s, \ D_s, \ U_s, \ W_s, \ V_s)$. Here we say that a matrix is strongly regular if the leading principal minors are nonsingular. Let $A = LU$ be its block LU factorization, then,*

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

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

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

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

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

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

Proof. For the proof of the lemma, we refer to [12, 15]. □

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.*

In Algorithm 1, for computing the LU factorization of a k-level SSS matrix, the matrix-matrix operations are performed on its $(k-1)$-level SSS generators, such as computing the recurrence of M_i in line 7 of Algorithm 1. Such operations lead to the growth of the $(k-1)$-level semiseparable order, which increases the computational complexity. This can be verified from the matrix-matrix operations introduced in [26, 32]. Take the 1-level SSS matrix A for example, the flops needed for computing A^2 is $\mathcal{O}(n^3N)$, where n is the semiseparable order and N is the number of blocks of A [26]. To be specific, the following lemma is introduced.

Lemma 3.3 ([32])**.** *Let* A_1*,* A_2 *be SSS matrices of lower semiseparable order* m_1 *and* n_1*, respectively. Then the product* A_1A_2 *is of lower semiseparable order at most* m_1+n_1*. Let* A_1*,* A_2 *be SSS matrices of upper semiseparable order* m_2 *and* n_2*, respectively. Then the product* A_1A_2 *is upper semiseparable of order at most* m_2+n_2*.*

Remark 3.7. *For a* k*-level SSS matrix, since the semiseparable order varies at different levels, results of Lemma 3.3 also hold for the* k*-level semiseparable order. But we do not know the exact upper bound of the* $(k-1)$*-level semiseparable order. We just know the* $(k-1)$*-level semiseparable order also increases.*

Lemma 3.3 gives rise to a question that whether there exists a minimal semiseparable order for a given SSS matrix such that the SSS matrix with a bigger semiseparable order is equal 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 the lower generators for 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* $A=$ *be an SSS 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 SSS matrices, the rank numbers are the minimal orders of the generators.*

Remark 3.8. *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.*

Remark 3.9. *Lemma 3.4 can also be applied to the* k*-level SSS matrices directly.*

The aim of model order reduction for a k-level SSS matrix A with its generators representation $A=\mathcal{MSSS}(P_s,\ R_s,\ Q_s,\ D_s,\ U_s,\ W_s,\ V_s)$ is to find $(k-1)$-level SSS matrices $\hat{P}_s$, $\hat{R}_s$, $\hat{Q}_s$, $\hat{U}_s$, $\hat{W}_s$, $\hat{V}_s$ of smaller order compared with P_s, R_s, Q_s, U_s, W_s, V_s, respectively such that $\hat{A}=\mathcal{MSSS}(\hat{P}_s,\ \hat{R}_s,\ \hat{Q}_s,\ D_s,\ \hat{U}_s,\ \hat{W}_s,\ \hat{V}_s)$ is of k-level semiseparable order smaller than or equal to the minimal k-level semiseparable order of A. Meanwhile, $\hat{A}$ is an approximation of A up to a small tolerance ϵ, i.e., $\|\hat{A}-A\|<\epsilon$.

Remark 3.10. *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.*

For discretized one-dimensional PDEs on a regular grid, the system matrix has a certain SSS structure. The LU factorization introduced in Lemma 3.2 could be performed as a direct solver. For discretized higher dimensional PDEs on regular grids, this LU factorization can be used as an efficient preconditioner.

3.2.2 Approximate Balanced Truncation for SSS Matrices

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

The SSS matrices have a realization of linear time-varying systems, which is studied by Dewilde et al. in [33]. Consider a mixed-causal system that is described by the following state-space model

$$\begin{aligned} \begin{bmatrix} x_{i+1}^c \\ x_{i-1}^a \end{bmatrix} &= \begin{bmatrix} R_i & \\ & W_i \end{bmatrix} \begin{bmatrix} x_i^c \\ x_i^a \end{bmatrix} + \begin{bmatrix} Q_i \\ V_i \end{bmatrix} u_i \\ y_i &= \begin{bmatrix} P_i & U_i \end{bmatrix} \begin{bmatrix} x_i^c \\ x_i^a \end{bmatrix} + D_i u_i, \end{aligned} \tag{14}$$

where x^c is the causal system state, x^a represents the anti-causal system state, u_i is the system input, and y_i is the system output. With zero initial system states and stack all the input and output as $\bar{u} = \begin{pmatrix} u_1^T, & u_2^T, & \ldots & u_N^T \end{pmatrix}^T$, $\bar{y} = \begin{pmatrix} y_1^T, & y_2^T, & \ldots & y_N^T \end{pmatrix}^T$, 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,

$$\mathcal{H} = \begin{bmatrix} D_i & U_1V_2 & U_1W_2V_3 & U_1W_2W_3V_4 \\ P_2Q_1 & D_2 & U_2V_3 & U_2W_3V_4 \\ P_3R_2Q_1 & P_3Q_2 & D_3 & U_3V_4 \\ P_4R_3R_2Q_1 & P_4R_3Q_2 & P_4Q_3 & D_4 \end{bmatrix}. \tag{15}$$

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

Lemma 3.5 ([34]). *The lower and upper semiseparable order for an SSS matrix with minimal LTV system realization are* $\max\{l_i\}_{i=2}^N$ *and* $\max\{k_i\}_{i=1}^{M-1}$*, respectively. Here* $\{l_i\}_{i=2}^M$ *and* $\{k_i\}_{i=1}^{M-1}$ *are defined in Table 3.1.*

We describe the lemma in [34] more exactly by restricting the realization of an SSS matrix to be minimal in Lemma 3.5. It is not difficult to set an example of an SSS matrix with small semiseparable order, but its LTV systems realization is of bigger order. Lemma 3.5 states that the order of the causal LTV system is equal to the lower semiseparable order of an SSS matrix, while the order of the anti-causal LTV system is equal to the upper semiseparable order. Thus, to reduce the semiseparable order of an SSS matrix is the same as reducing the order of its realization by mixed-causal LTV systems.

Model order reduction for LTV systems is studied in [35, 36]. In [36], a linear matrix inequality (LMI) approach was introduced to solve the Lyapunov inequalities to compute the controllability and observability gramians. In [35], the low-rank Smith method was presented to approximate the square-root of the controllability and observability gramians of LTV systems.

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

$$\begin{cases} x_{k+1} = R_k x_k + Q_k u_k \\ \quad y_k = P_k x_k, \end{cases} \tag{16}$$

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

$$\begin{aligned} \mathcal{G}_c(k+1) &= R_k \mathcal{G}_c(k) R_k^T + Q_k Q_k^T, \\ \mathcal{G}_o(k) &= R_k^T \mathcal{G}_o(k+1) R_k + P_k^T P_k, \end{aligned} \tag{17}$$

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 and often have a large jump at an early stage [37, 38]. This suggests to approximate these two Gramians at each step by a low-rank approximation. In this paper, we just consider the case that the LTV systems are uniformly completely controllable and observable over the time interval, which means that the gramians $\mathcal{G}_c$ and $\mathcal{G}_o$ are positive definite. This is reasonable because the SSS matrices considered in this paper correspond to uniformly completely controllable and observable LTV systems. The low-rank approximation of the controllability and observability gramians for LTV systems is introduced in Algorithm 2.

Algorithm 2 Low-Rank Approximation of the Gramians for LTV Systems

```
Initialize: LTV system $\{P_k\}_{k=2}^{N}$, $\{R_k\}_{k=2}^{N-1}$, $\{Q_k\}_{k=1}^{N-1}$, reduced LTV system order $m$
for $k = 2 : N$ do
    if $k == 2$ then
        $\left[\begin{array}{c} Q_{k-1} \end{array}\right] = U_c \Sigma_c V_c^T$ (singular value decomposition)
    else
        $\left[\begin{array}{c|c} Q_{k-1} & R_{k-1}\tilde{L}_c(k-1) \end{array}\right] = U_c \Sigma_c V_c^T$ (singular value decomposition)
    end if
    Partition $U_c = \left[\begin{array}{c|c} U_{c1} & U_{c2} \end{array}\right]$, $\Sigma_c = \left[\begin{array}{c|c} \Sigma_{c1} & \\ \hline & \Sigma_{c2} \end{array}\right]$, $U_{c1} \in \mathbb{R}^{M \times m}$, $\Sigma_{c1} \in \mathbb{R}^{m \times m}$.
    Let $\tilde{L}_c(k) = U_{c1}\Sigma_{c1} \in \mathbb{R}^{M \times m}$
end for
for $k = N : 2$ do
    if k==N then
        $\left[\begin{array}{c} P_k^T \end{array}\right] = U_o \Sigma_o V_o^T$ (singular value decomposition)
    else

        $\left[\begin{array}{c|c} P_k^T & R_k^T\tilde{L}_o(k+1) \end{array}\right] = U_o \Sigma_o V_o^T$ (singular value decomposition)
    end if
    Partition $U_o = \left[\begin{array}{c|c} U_{o1} & U_{o2} \end{array}\right]$, $\Sigma_o = \left[\begin{array}{c|c} \Sigma_{o1} & \\ \hline & \Sigma_{o2} \end{array}\right]$, $U_{o1} \in \mathbb{R}^{M \times m}$, $\Sigma_{o1} \in \mathbb{R}^{m \times m}$.
    Let $\tilde{L}_o(k) = U_{o1}\Sigma_{o1} \in \mathbb{R}^{M \times m}$
end for
Output: Approximated fators $\left\{\tilde{L}_c(k)\right\}_{k=2}^{N}$, $\left\{\tilde{L}_o(k)\right\}_{k=2}^{N}$
```

Below we show how to obtain such low-rank approximations. Since the controllability gramian $\mathcal{G}_c(k)$ and observability gramian $\mathcal{G}_o(k)$ have similar structure, we will only use the controllability gramian $\mathcal{G}_c(k)$ to introduce the basic idea.

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

$$\mathcal{G}_c(k) = L_c(k)L_c^T(k), \tag{18}$$

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

$$\tilde{\mathcal{G}}_c(k) = \tilde{L}_c(k)\tilde{L}_c^T(k), \tag{19}$$

with $\tilde{L}_c(k) \in \mathbb{R}^{M \times m_k}$ where m_k is the ϵ-rank of $\mathcal{G}_c(k)$, and $m_k < M$. Typically, m_k is set to be constant, i.e., $m_k = m$ at each step. It can be also chosen adaptively by setting a threshold ϵ for the truncated singular values. If $\mathcal{G}_c(k)$ is of low numerical rank, it is reasonable to use the rank m_k approximation (19) to approximate $\mathcal{G}_c(k)$.

In [17], the recursive low-rank gramians method was used to approximate the gramians of the linear time-invariant (LTI) systems. Such methods can also be applied to approximate the gramians of the LTV systems. This is studied by the same author in an earlier reference [18]. In this manuscript, we study the connections between LTV systems and SSS matrices. Meanwhile, we extend the model order reduction algorithm for LTV systems to the model order reduction for SSS matrices. The low-rank approximation method in [17, 18] was used to approximate the gramians of the LTV systems that the SSS matrix corresponds to and the approximate balanced truncation method was applied for the model order reduction. Even the low-rank approximation method in this manuscript and the one in [18] are quite similar, the novelty is that this algorithm has never been applied to reduce the rank of the off-diagonal blocks of structured matrices.

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

$$\mathcal{H}_k = \mathcal{O}_k \mathcal{C}_k, \tag{20}$$

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

$$\begin{aligned} \mathcal{G}_c(k) &= \mathcal{C}_k \mathcal{C}_k^T, \\ \mathcal{G}_o(k) &= \mathcal{O}_k^T \mathcal{O}_k, \end{aligned} \tag{21}$$

where in (21) $\mathcal{G}_c(k)$ and $\mathcal{G}_o(k)$ are the controllability gramian and observability gramian defined in (17), respectively.

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

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

It was shown in [39] that for any two positive definite matrices, there always exits a so-called contragredient transformation such that

$$\Lambda_k = T_k^{-1} \mathcal{G}_c(k) T_k^{-T} = T_k^T \mathcal{G}_o(k) T_k, \tag{22}$$

where Λ_k is a diagonal matrix and

$$\Lambda_k = \operatorname{diag}(\lambda_{k_1},\ \lambda_{k_2},\ \cdots,\ \lambda_{k_M}). \tag{23}$$

With this contragredient transformation, we have

$$T_k^{-1} \mathcal{G}_c(k) \mathcal{G}_o(k) T_k = \Lambda_k^2. \tag{24}$$

This states that $\{\lambda_{k_i}\}_{i=1}^M$ are the Hankel singular values at time instant k. Such contragredient transformation brings the systems into a "balanced" form, which means that the controllability gramian and observability gramian of the system are equal to a diagonal matrix. For the LTV system (16), after such a transformation, the balanced LTV system is,

$$\begin{cases} \bar{x}_{k+1} = T_{k+1}^{-1} R_k T_k \bar{x}_k + T_{k+1}^{-1} Q_k u_k \\ \quad y_k = P_k T_k \bar{x}_k. \end{cases} \tag{25}$$

Since for system (25), the controllability and observability gramians are balanced, truncation can be performed to truncate the Hankel singular values that are below a set threshold. This could be done by using the left and right multipliers L_l and L_r that are defined by

$$L_l = \begin{bmatrix} I_m & 0 \end{bmatrix},\ L_r = \begin{bmatrix} I_m \\ 0 \end{bmatrix}, \tag{26}$$

where I_m is an $m \times m$ identity matrix and m is the reduced system dimension size. Then the reduced LTV system is

$$\begin{cases} \tilde{x}_{k+1} = \Pi_l(k+1) R_k \Pi_r(k) \tilde{x}_k + \Pi_l(k+1) Q_k u_k \\ \quad y_k = P_k \Pi_r(k) \tilde{x}_k, \end{cases} \tag{27}$$

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

The reduced LTV system (27) is computed via a projection method with the projector defined by $\pi(k) = \pi_r(k)\pi_l(k)$. This is because

$$\pi_l(k)\pi_r(k) = L_l T_k^{-1} T_k L_r = I_m,$$

and

$$\pi(k)^2 = \pi_r(k)\pi_l(k)\pi_r(k)\pi_l(k) = \pi(k).$$

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

$$\Lambda'_k = \bar{T}_k^{-1}\tilde{\mathcal{G}}_c(k)\bar{T}_k^{-T} = \bar{T}_k^T\tilde{\mathcal{G}}_o(k)\bar{T}_k, \tag{28}$$

where Λ'_k is a diagonal matrix and

$$\Lambda'_k = \operatorname{diag}(\lambda'_{k_1},\ \lambda'_{k_2},\ \cdots,\ \lambda'_{k_m},\ 0,\ \cdots,\ 0). \tag{29}$$

Lemma 3.6 ([39], Theorem 3)**.** *Let symmetric positive semidefinite $Q,\ P \in \mathbb{R}^{M\times M}$ satisfy*

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

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

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

Proof. For the proof of the lemma, we refer to [39]. □

We have already explained that the diagonal entries of the matrix Λ'_k in (29) are the Hankel singular values of the approximate Hankel map in (20). It was therefore not difficult to show that if the controllability gramian $\mathcal{G}_c(k)$ and the observability gramian $\mathcal{G}_o(k)$ are well approximated by $\tilde{\mathcal{G}}_c(k)$ and $\tilde{\mathcal{G}}_o(k)$ separately, then $\tilde{\mathcal{G}}_c(k)\tilde{\mathcal{G}}_o(k)$ also approximates $\mathcal{G}_c(k)\mathcal{G}_o(k)$ well. This means that the approximate Hankel singular values $\left\{\lambda'_{k_i}\right\}_{i=1}^m$ are close to the original Hankel singular values $\{\lambda_{k_i}\}_{i=1}^m$. In the Algorithm 3, we show how to use the approximated gramians $\tilde{\mathcal{G}}_c(k)$ and $\tilde{\mathcal{G}}_o(k)$ to compute the reduced system. By using the approximated gramians, this method is called the approximate balanced truncation.

Algorithm 3 Approximate Balanced Truncation for LTV systems

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

Proof. According to Lemma 3.6, there exists a contragredient transformation $\bar{T}_k \in \mathbb{R}^{M\times M}$ such that

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

where Λ_k' is a diagonal matrix and

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

Here

$$\Sigma_k = \operatorname{diag}(\lambda_{k_1}',\ \lambda_{k_2}',\ \cdots,\ \lambda_{k_m}'),$$

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

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

According to the proof of Lemma 3.6, the contragredient transformation $\bar{T}_k$ are computed via

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

and the inverse of such transformation is computed by

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

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

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

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

$$\Pi_l(k) = \begin{bmatrix}I_m & 0\end{bmatrix}T_k^{-1} = \begin{bmatrix}I_m & 0\end{bmatrix}\begin{bmatrix}\Sigma_k & 0\\ 0 & \bar{\Sigma}_k\end{bmatrix}^{-\frac{1}{2}}\begin{bmatrix}V_k^T & 0\\ 0 & \bar{U}_k^T\end{bmatrix}\begin{bmatrix}\tilde{L}_o(k)^T\\ N_c(k)^T\end{bmatrix} = \Sigma_k^{-\frac{1}{2}}V_k^T\tilde{L}_o^T(k),$$
$$\Pi_r(k) = T_k\begin{bmatrix}I_m\\ 0\end{bmatrix} = \begin{bmatrix}\tilde{L}_c(k) & N_o(k)\end{bmatrix}\begin{bmatrix}U_k & 0\\ 0 & \bar{V}_k\end{bmatrix}\begin{bmatrix}\Sigma_k & 0\\ 0 & \bar{\Sigma}_k\end{bmatrix}^{-\frac{1}{2}}\begin{bmatrix}I_m\\ 0\end{bmatrix} = \tilde{L}_c(k)U_k\Sigma_k^{-\frac{1}{2}}.$$

And the projection is defined via

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

since $\Pi_l(k)\Pi_r(k) = I_m$ and $\Pi_k^2 = \Pi_k$. □

For an SSS matrix $A = \mathcal{SSS}(P_s,\ R_s,\ Q_s,\ D_s,\ U_s,\ W_s,\ V_s)$ with lower semiseparable order M, we have already explained its LTV system realization. Thus, Algorithm 2 and Algorithm 3 can be performed to reduce the order of the causal LTV system (16), which corresponds to reduce the lower semiseparable order. This yields the approximated SSS matrix $\tilde{A} = \mathcal{SSS}(\tilde{P}_s,\ \tilde{R}_s,\ \tilde{Q}_s,\ D_s,\ U_s,\ W_s,\ V_s)$. For the strictly upper-triangular part of A, we first transpose it to the strictly lower-triangular form then perform Algorithm 2 and Algorithm 3. After this reduction, we transpose the reduced strictly lower-triangular part to the strictly upper-triangular form.

3.2.3 Hankel Blocks Approximation

To compare with our model order reduction method for SSS matrices, we describe the standard model order reduction algorithm in this part. It is called the Hankel blocks approximation in [13, 26]. The Hankel blocks of an SSS matrix 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 block SSS matrix A for example, the Hankel blocks for the strictly lower-triangular part are shown in Fig. 2 by H_2, H_3 and H_4.

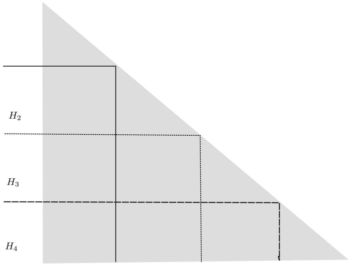


Figure 2: Hankel blocks of a 4×4 block SSS matrix

It is easy to very that for the Hankel blocks $H_i, \ (i = 2, \ \ldots, \ N)$, the following relation hold

$$H_i = \mathcal{O}_i \mathcal{C}_i, \ (i = 2, \ , \ldots, \ N), \tag{30}$$

where $\mathcal{O}_i$ and $\mathcal{C}_i$ are the current state to the current and future output map and the past input to the current state map for system (16), respectively. Moreover, the following relation hold for $\mathcal{O}_i$ and $\mathcal{C}_i$.

$$\mathcal{O}_{i-1} = \begin{bmatrix} P_{i-1} \\ \mathcal{O}_i R_{i-1} \end{bmatrix}, \ (i = 2, \ , \ldots, \ N-1), \ \mathcal{O}_N = P_N \tag{31}$$

$$\mathcal{C}_{i+1} = \begin{bmatrix} R_i \mathcal{C}_i & Q_i \end{bmatrix}, \ (i = 2, \ , \ldots, \ N-1), \ \mathcal{C}_2 = Q_1 \tag{32}$$

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

$$\mathcal{G}_c(i) = \mathcal{C}_i \mathcal{C}_i^T, \ \mathcal{G}_o(i) = \mathcal{O}_i^T \mathcal{O}_i, \tag{33}$$

where $\mathcal{G}_c(i)$ and $\mathcal{G}_o(i)$ are the controllability gramian and observability gramian that satisfy (17).

The rank of the Hankel map H_i at time step i, i.e., the rank of the i-th Hankel block is the order of the states x_i of system (16) [29]. The standard way to reduce the semiseparable order is given in [13, 26]. This standard approach is based on the realization theory of a given Hankel map for LTV systems that is introduced in [29, 33], i.e., according to the given Hankel map $\{H_i\}_{i=2}^N$, find a triple $\{P_k, \ R_k, \ Q_k\}$ that satisfy (30) (31) (32). By using the realization theory, it is also possible to get the reduced triple $\left\{\hat{P}_k, \ \hat{R}_k, \ \hat{Q}_k\right\}$ that approximates the Hankel map H_i in (30).

To do this approximation, first we need to transform the map $\mathcal{O}_i$ to the form that has orthonormal columns and transform the map $\mathcal{C}_i$ to the form that has orthonormal rows. These two form are called left proper form and right proper form [13, 26], respectively. We use the change of $\mathcal{C}_i$ to introduce the basic idea. The first step is to do a singular value decomposition (SVD) of the starting point $\mathcal{C}_2$, which gives

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

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

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

then the Hankel map

$$\bar{H}_2 = \bar{\mathcal{O}}_2 \bar{\mathcal{C}}_2 = \mathcal{O}_2 U_2 \Sigma_2 V_2^T = \mathcal{O}_2 \mathcal{C}_2 = H_2.$$

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

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

and

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

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

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

$$\mathcal{C}_{i+1} = \begin{bmatrix} \bar{R}_i\bar{\mathcal{C}}_i & Q_i \end{bmatrix} = \begin{bmatrix} \bar{R}_i & Q_i \end{bmatrix} \begin{bmatrix} \bar{C}_i & \\ & I \end{bmatrix}. \tag{34}$$

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

$$\begin{bmatrix} \bar{R}_i & Q_i \end{bmatrix} = U_i\Sigma_i V_i^T, \tag{35}$$

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

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

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

$$\bar{\mathcal{C}}_{i+1} = \begin{bmatrix} \bar{R}_i\bar{\mathcal{C}}_i & \bar{Q}_i \end{bmatrix} = \begin{bmatrix} \bar{R}_i & \bar{Q}_i \end{bmatrix} \begin{bmatrix} \bar{\mathcal{C}}_i & \\ & I \end{bmatrix} = V_i^T \begin{bmatrix} \bar{\mathcal{C}}_i & \\ & I \end{bmatrix}, \tag{36}$$

which also has orthonormal rows. This is due to

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

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

$$\mathcal{C}_{i+1} = U_i\Sigma_i\bar{\mathcal{C}}_{i+1}, \tag{37}$$

which can be checked by associating (34) and (35) with (36).

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

$$\bar{\mathcal{O}}_{i+1} = \mathcal{O}_{i+1}U_i\Sigma_i. \tag{38}$$

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

$$\bar{\mathcal{O}}_{i+1} = \begin{bmatrix} \bar{P}_{i+1} \\ \mathcal{O}_{i+2}\bar{R}_{i+1} \end{bmatrix}. \tag{39}$$

And by checking (37)(38)(39), it is easy to get the unchanged Hankel map at time step $i+1$. Similar procedure can be applied to transform the map $\mathcal{O}_i$ to the form that has orthonormal columns. The procedure for transforming $\mathcal{C}_i$ and $\mathcal{O}_i$ is shown by Algorithm 4.

Algorithm 4 Hankel Blocks Approximation

```
Initialize: LTV system $\{P_k\}_{k=2}^{N}$, $\{R_k\}_{k=2}^{N-1}$, $\{Q_k\}_{k=1}^{N-1}$, reduced system order $m$
for $i = 2 : N$ do
    if $i == 2$ then
        $Q_{i-1} = U_i \Sigma_i V_i^T$ (singular value decomposition)
        Let $Q_{i-1} = V_i^T$, $P_i = P_i U_i \Sigma_i$, and $R_i = R_i U_i \Sigma_i$
    else if $i == N$ then
        $\begin{bmatrix} R_{i-1} & Q_{i-1} \end{bmatrix} = U_i \Sigma_i V_i^T$ (singular value decomposition)
        Partition $V_i^T = \begin{bmatrix} V_{i_1}^T & V_{i_2}^T \end{bmatrix}$ such that the size of $Q_{i-1}$ and $V_{i_2}^T$ match
        Let $R_{i-1} = V_{i_1}^T$, $Q_{i-1} = V_{i_2}^T$, $P_i = P_i U_i \Sigma_i$
    else
        $\begin{bmatrix} R_{i-1} & Q_{i-1} \end{bmatrix} = U_i \Sigma_i V_i^T$ (singular value decomposition)
        Partition $V_i^T = \begin{bmatrix} V_{i_1}^T & V_{i_2}^T \end{bmatrix}$ such that the size of $Q_{i-1}$ and $V_{i_2}^T$ match
        Let $R_{i-1} = V_{i_1}^T$, $Q_{i-1} = V_{i_2}^T$, $P_i = P_i U_i \Sigma_i$ and $R_i = R_i U_i \Sigma_i$
    end if
end for
for $i = N : 2$ do
    if $i == N$ then
        $P_i = U_i \Sigma_i V_i^T$ (singular value decomposition)
        Let $P_i = U_i$, $R_{i-1} = \Sigma_i V_i^T R_{i-1}$, $Q_{i-1} = \Sigma_i V_i^T Q_{i-1}$
    else if $i == 2$ then
        $\begin{bmatrix} P_i \\ R_i \end{bmatrix} = U_i \Sigma_i V_i^T$ (singular value decomposition)
        Partition $U_i = \begin{bmatrix} U_{i_1} \\ U_{i_2} \end{bmatrix}$ such that the size of $U_{i_2}$ and $R_i$ match
        Let $P_i = U_{i_1}$, $R_i = U_{i_2}$, $Q_{i-1} = \Sigma_i V_i Q_{i-1}$
    else
        $\begin{bmatrix} P_i \\ R_i \end{bmatrix} = U_i \Sigma_i V_i^T$ (singular value decomposition)
        Partition $U_i = \begin{bmatrix} U_{i_1} \\ U_{i_2} \end{bmatrix}$ such that the size of $U_{i_2}$ and $R_i$ match
        Let $P_i = U_{i_1}$, $R_i = U_{i_2}$, $Q_i = \Sigma_i V_i Q_i$
    end if
end for
for $i = 1 : N$ do
    if $i == 1$ then
        Partition $Q_i = \left[\begin{array}{c} Q_{i_1} \\ \hline (\cdot) \end{array}\right]$ with $Q_{i_1} \in \mathbb{R}^{m \times (\cdot)}$, let $\hat{Q}_i = Q_{i_1}$
    else if i==N then
        Partition $P_i = \left[\begin{array}{c} P_{i_1} \\ \hline (\cdot) \end{array}\right]$ with $P_{i_1} \in \mathbb{R}^{(\cdot) \times m}$, let $\hat{P}_i = P_{i_1}$
    else
        Partition $P_i = \left[\begin{array}{c} P_{i_1} \\ \hline (\cdot) \end{array}\right]$ with $P_{i_1} \in \mathbb{R}^{(\cdot) \times m}$, let $\hat{P}_i = P_{i_1}$
        Partition $R_i = \left[\begin{array}{c|c} R_{i_1} & (\cdot) \\ \hline (\cdot) & (\cdot) \end{array}\right]$ with $R_{i_1} \in \mathbb{R}^{m \times m}$, let $\hat{R}_i = R_{i_1}$
        Partition $Q_i = \left[\begin{array}{c} Q_{i_1} \\ \hline (\cdot) \end{array}\right]$ with $Q_{i_1} \in \mathbb{R}^{m \times (\cdot)}$, let $\hat{Q}_i = Q_{i_1}$
    end if
end for
Output: Reduced LTV systems $\left\{\hat{P}_k\right\}_{k=2}^{N}$, $\left\{\hat{R}_k\right\}_{k=2}^{N-1}$, $\left\{\hat{Q}_k\right\}_{k=1}^{N-1}$
```

After transforming the map $\mathcal{O}_i$ and $\mathcal{C}_i$ into the form with orthogonal column basis and row basis, we can do the truncation to truncate unimportant column spaces and row spaces of $\mathcal{O}_i$ and $\mathcal{C}_i$, which gives the approximated Hankel map, $\hat{H}_i = \hat{\mathcal{O}}_i \hat{\mathcal{C}}_i$.

Remark 3.11. *For the approximate balanced truncation,the map $\mathcal{O}_i$ and $\mathcal{C}_i$ are approximated separately via the low-rank approximation Algorithm 2, if the maps are not well balanced, the Hankel map H_i is not well approximated. As we have introduced in Algorithm 4, this approximation satisfies*

$$\|H_k - \hat{H}_k\|_2 = \sigma_{m+1},$$

where σ_{m+1} is the $(m+1)$th Hankel singular values (singular values of the Hankel map). This equality illustrates that the Hankel blocks approximation method in Algorithm 4 is an optimal Hankel model order reduction method.

Since the balanced truncation does not give the reduced model in any optimality of norm, it is clear that the Hankel blocks approximation gives much more accurate reduced system than the approximate balanced truncation. For the error bound for the approximate balanced truncation, we refer to [40].

Remark 3.12. *As introduced aforementioned, the Hankel blocks approximation gives a more accurate approximation than the approximate balanced truncation, in the following part, it is shown that the approximate balanced truncation is computationally cheaper than the Hankel blocks approximation method. In this paper, we focus on the preconditioning technique, therefore we do not only care about the accuracy, but also concern with the computational efficiency.*

3.2.4 Operations Count for the Two Model Order Reduction Methods

Given an SSS matrix $A = \mathcal{SSS}(P_S,\ R_s,\ Q_s,\ D_s,\ U_s,\ W_s,\ V_s)$, to compare the operations count of the approximate balanced truncation described by Algorithm 2-3 and the Hankel blocks approximation introduced in Algorithm 4, we assume that the generators sizes in Table 3.1 are uniform, i.e., $m_i = n$ and $k_i = l_i = M$. Here N is the number of SSS blocks (LTV system time steps), M is the unreduced LTV system order, and $N \gg M \gg n$. The reduced SSS matrix $\tilde{A} = \mathcal{SSS}(\hat{P}_s,\ \hat{R}_s,\ \hat{Q}_s,\ D_s,\ \hat{U}_s,\ \hat{W}_s,\ \hat{V}_s)$, where $\hat{k}_i = \hat{l}_i = m$, m is the reduced semiseparable order and $m \ll M$.

In this paper, we measure the operations count by the floating-point operations (flops). To compute the operations count of the approximate balanced truncation, first we compute the operations count for the low-rank approximation in Algorithm 2. In the forward recursion, the singular value decomposition uses

$$m^2M + (m+n)^2M(N-2) \tag{40}$$

flops. In this recursion, two matrix-matrix product are computed in each iteration, this consumes

$$mM^2(N-2) + m^2M(N-1) \tag{41}$$

flops. Adding (40) and (41) gives the total flops count for the forward low-rank approximation,

$$m^2M + (m+n)^2M(N-2) + mM^2(N-2) + m^2M(N-1), \tag{42}$$

Since the forward low-rank approximation and the backward low-rank approximation are symmetric in computing, the flops count for the backward low-rank approximation is equal to (42). Then the flops count $\mathcal{F}_l$ for the low-rank approximation Algorithm 2 is

$$\mathcal{F}_l = 2mM^2N + 4m^2MN + 4mnMN + 2n^2MN - 4(m+n)^2M - 4mM^2. \tag{43}$$

Next we compute the operations count of the approximate balanced truncation Algorithm 3. First, to compute $\Pi_l(k)$ and $\Pi_r(k)$, the flops count is

$$\left(m^2M + m^3 + 2(m^2 + m^2M)\right)(N-1), \tag{44}$$

and the flops count to compute the reduced LTV system is

$$2mnM(N-1) + (mM^2 + m^2M)(N-2). \tag{45}$$

Thus the total flops count $\mathcal{F}_a$ for the approximate balanced truncation is the summation of (44) and (45), which gives

$$\mathcal{F}_a = (M^2 + mM + 2nM)N - 2mn(M + m + n). \tag{46}$$

Then we have the total flops count $\mathcal{F}_{la}$ of the approximate balanced truncation by adding (43) to (46). Since we have $N \gg M \gg m,\ n$, we just use the $\mathcal{O}(\cdot)$ to denote the total flops count. Thus, we have

$$\mathcal{F}_{la} = \mathcal{O}\left((2m + 1)M^2 N\right). \tag{47}$$

Similarly, we can compute the operations count $\mathcal{F}_h$ for the Hankel blocks approximation in Algorithm 4 denoted by

$$\mathcal{F}_h = 4M^3N + 6nM^2N + 2(n^2 + 2n)MN - 8M^3 - 12nM^2 + 2(n^2 - 3n)M + 2n^2, \tag{48}$$

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

$$\mathcal{F}_h = \mathcal{O}(4M^3N). \tag{49}$$

Since $N \gg M \gg m$, by comparing the flops count $\mathcal{F}_{la}$ for the approximate balanced truncation in (47) with the flops count $\mathcal{F}_h$ for the Hankel blocks approximation in (48), we see that the approximate balanced truncation algorithm is computationally cheaper than the Hankel blocks approximation for the model order reduction of SSS matrices.

Remark 3.13. *By checking the flops count $\mathcal{F}_{la}$ for the approximate balanced truncation in (47) with the flops count $\mathcal{F}_h$ for the Hankel blocks approximation in (48), we can see that the flops count is linear with N for both method, where N denotes the number of blocks of an SSS matrix. Moreover, the size of the SSS matrix equals to nN and $n \ll N$. Thus, both methods give computational complexity that is linear with the matrix size.*

3.2.5 Flowchart of Preconditioning by Using MSSS Matrix Computations

We have already described the MSSS matrix computations and how to compute a preconditioner using such matrix computations. In this part, we use a flowchart to illustrate how to compute a preconditioner for the PDE-constrained optimization problem (1). This flowchart is shown in Fig. 3.

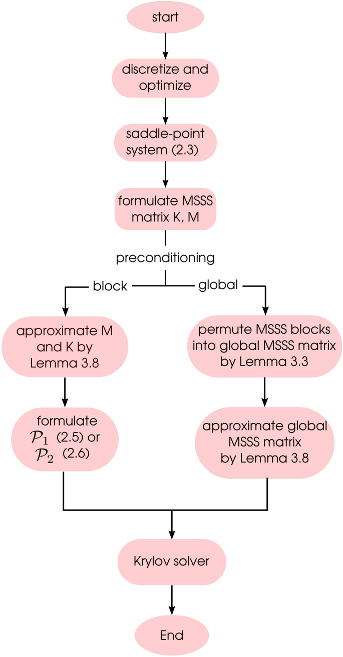

start
discretize and optimize
saddle-point system (2.3)
formulate MSSS matrix K, M
preconditioning
block
global
approximate M and K by Lemma 3.8
permute MSSS blocks into global MSSS matrix by Lemma 3.3
formulate $\mathcal{P}_1$ (2.5) or $\mathcal{P}_2$ (2.6)
approximate global MSSS matrix by Lemma 3.8
Krylov solver
End

Figure 3: Flowchart for MSSS preconditioning of PDE-constrained optimization problem

4 Numerical Experiments

In this section, we study the problem of optimal control of the convection-diffusion equation that is introduced in Example 4.1. First, we compare the performance of our model order reduction algorithm with the conventional model order reduction algorithm. Next we test the global MSSS preconditioner and the block diagonal MSSS preconditioner. Numerical experiments in [21] also show the advantage of the global MSSS preconditioner over the lower-triangular block MSSS preconditioner for the PDE-constrained optimization problems. The superiority of the global MSSS preconditioner to the block preconditioners that are computed by the multigrid methods for computational fluid dynamics (CFD) problems is illustrated in [22].

Example 4.1 ([20]). *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,* $\overrightarrow{\omega} = (cos(\theta), \ sin(\theta))^T$ *is the unit directional vector and the prescribed state* $\hat{u} = 0$.

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

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 by Example 4.1. For the block-diagonal preconditioner $\mathcal{P}_1$ that is computed by the approximate balanced truncation algorithm and the Hankel blocks approximation method, the results for different ϵ and β are shown in Table 2 - 9 while θ is set to be $\frac{\pi}{5}$ for all the experiments.

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

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

Table 3: Results for Hankel blocks approximation for $\beta = 10^{-1}$, $\epsilon = 10^{-1}$

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

Table 4: Results for approximate balanced truncation for $\beta = 10^{-1}$, $\epsilon = 10^{-2}$

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

Table 5: Results for Hankel blocks approximation for $\beta = 10^{-1}$, $\epsilon = 10^{-2}$

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

Table 6: Results for approximate balanced truncation for $\beta = 10^{-2}$, $\epsilon = 10^{-1}$

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

Table 7: Results for Hankel blocks approximation for $\beta = 10^{-2}$, $\epsilon = 10^{-1}$

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

Table 8: Results for approximate balanced truncation for $\beta = 10^{-2}$, $\epsilon = 10^{-2}$

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

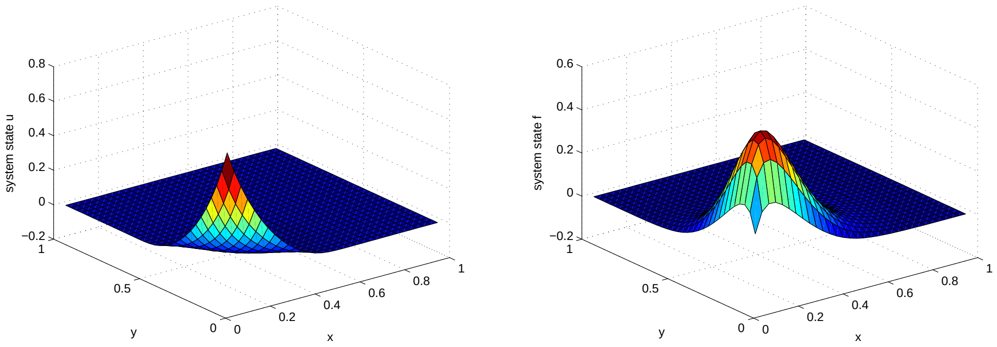
Table 9: Results for Hankel blocks approximation for $\beta = 10^{-2}$, $\epsilon = 10^{-2}$

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

The results in Table 2 - 9 show that the time to compute the preconditioner and iteratively solve the saddle-point system is linear in the problem size, which states that the MSSS preconditioning technique has linear computational complexity. This shows that for the same group of ϵ and β, the block MSSS preconditioners computed by the approximate balanced truncation and Hankel blocks approximation methods give mesh size independent convergence. Moreover, the number of iterations for the block MSSS preconditioners computed by both model order reduction algorithms are the same.

Remark 4.1. *As shown by* (47) *and* (48)*, the approximate balanced truncation is computationally cheaper than the Hankel blocks approximation and both algorithms have linear computational complexity. This is illustrated by the time to compute the preconditioners by these two methods for the same group of β and ϵ in Table 2 - 9.*

The optimal solution of the system states and input for $\beta = 10^{-2}$, $\epsilon = 10^{-1}$ and $h = 2^{-5}$ are shown in Fig. 4(a) and Fig. 4(b).



(a) Optimal system states u.

(b) Optimal system input f.

Figure 4: Solution of the system states and input for $\beta = 10^{-2}$, $\epsilon = 10^{-1}$ and $h = 2^{-5}$.

The block-diagonal and block lower-triangular MSSS preconditioners computed by these two model order reduction algorithms are also tested on the problems of optimal control of the Poisson equation and optimal control of the convection-diffusion equation in [21]. These results in [21] are consistent with the conclusions we draw in this part on the performance of these two model order reduction algorithms and the performance of such block MSSS preconditioners.

4.2 Comparison of Preconditioners

In this part, we test the performance of the block-diagonal MSSS preconditioner and the global MSSS preconditioner. For the block diagonal MSSS preconditioner, from Table 2 - 9 we have seen 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, which plays an increasingly more important role with the decrease of β, while this term is often neglected in the preconditioner $\mathcal{P}_1$ in (5) for big and middle value of β [20]. If we continue decreasing β, we obtain the computational results for the block-diagonal MSSS preconditioner in Table 10-11. For the preconditioners tested in this part, the Hankel blocks approximation method is chosen as the model order reduction algorithm. Results for the preconditioners computed by the approximate balanced truncation can be found in [21].

Table 10: Results for the block-diagonal MSSS preconditioner (5) for $\beta = 10^{-3}$, $\epsilon = 10^{-1}$

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

Table 11: Results for the block-diagonal MSSS preconditioner (5) for $\beta = 10^{-4}$, $\epsilon = 10^{-1}$

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

As shown in Table 10 - 11, with the decrease of β from 10^{-3} to 10^{-4}, the number of iterations more than doubles. Clearly if β decreases too much, the block-diagonal MSSS preconditioner $\mathcal{P}_1$ cannot give satisfactory results. Alternatively, for small β, we can choose the block-diagonal MSSS preconditioner as follows

$$\mathcal{P}_1 = \begin{bmatrix} 2\beta\hat{M} & & \\ & \hat{M} & \\ & & -\frac{1}{2\beta}\hat{M} \end{bmatrix}. \tag{50}$$

The computational results of this preconditioner for $\beta = 10^{-4}$ are given in Table 12. The maximum number of iterations is set to 100.

Table 12: Results for the block-diagonal MSSS preconditioner (50) for $\beta = 10^{-4}$, $\epsilon = 10^{-1}$

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

The results in Table 11 - 12 show that the block-diagonal MSSS preconditioner do not give satisfactory performance when β becomes so small. Here in the table, ”no convergence” means that the 2-norm of the residual does not converge to desired accuracy within 100 iterations. This is due to the fact that the Schur complement is difficult to approximate for small β.

Recall from Section 2 that we can permute the saddle-point system with MSSS blocks into a global MSSS system. Due to the indefiniteness of the global MSSS preconditioner, `MINRES` is not suitable to iteratively solve the preconditioned saddle-point system, the induced dimension reduction (`IDR(s)`) method [41] is chosen as the Krylov solver. To compare with the results for the block-diagonal MSSS preconditioner in Table 10 - 12, we apply the global MSSS preconditioner to the same test case. The results are given in Table 13 - 14.

Table 13: Results for the global MSSS preconditioner for $\beta = 10^{-3}$ and $\epsilon = 10^{-1}$

problem size	iterations	preconditioning (sec.)	IDR(4) (sec.)	total (sec.)
3.07e+03 (4)	2	0.38	0.13	0.51
1.23e+04 (6)	2	1.16	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 14: Results for the global MSSS preconditioner for $\beta = 10^{-4}$ and $\epsilon = 10^{-1}$

problem size	iterations	preconditioning (sec.)	IDR(4) (sec.)	total (sec.)
3.07e+03 (4)	2	0.38	0.13	0.51
1.23e+04 (6)	2	1.15	0.24	1.39
4.92e+04 (7)	2	4.23	0.64	4.87
1.97e+05 (9)	2	17.87	2.18	20.05

Even though it takes slightly longer time to compute the global MSSS preconditioner than to compute the block-diagonal MSSS preconditioner, much less time is needed for the `IDR(s)` method to solve the preconditioned system by the global MSSS preconditioner. Meanwhile, the time to compute both preconditioners and to solve the preconditioned system by such preconditioners scales linearly with the problem size.

Remark 4.2. *By comparing the computational results of the global MSSS preconditioner with that of the block-diagonal MSSS preconditioner, we find that for the same numerical test with the same group of β and ϵ that the numer of iterations is reduced significantly by the global MSSS preconditioner. Meanwhile, the global MSSS preconditioner gives both mesh size and β independent convergence. This makes the global MSSS preconditioner superior to the block preconditioners.*

5 Preconditioning for Optimal Control of 3D Problems

As analyzed in Section 3.2.1, to do an *LU* factorization of a k-level SSS matrix, the model order reduction of a $(k-1)$-level SSS matrix is needed. To compute a preconditioner for 3D problems using MSSS matrix computations, model order reduction for 2-level SSS matrices is needed. Since the model order reduction for 2 and higher level SSS matrices is still an open problem. Only preliminary results for optimal control of the 3D Poisson equation in Example 5.1 are given in this section. Even these are prelimiary results, the method introduced in this part for computing the preconditioner for 3D problems using MSSS matrix computations alrady gives satisfactory results.

Example 5.1. *Consider the following problem of optimal control of the 3D Poisson equation*

$$\begin{aligned} &\min_{u,f} \frac{1}{2}\|u-\hat{u}\| + \frac{\beta}{2}\|f\|^2 \\ &s.t. -\nabla^2 u = f \ in \ \Omega \\ &\qquad u = u_D \ on \ \partial\Omega, \end{aligned} \tag{51}$$

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

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

The discretized analog of problem (51) is

$$\begin{aligned} &\min_{u,\ f} \frac{1}{2}\|u-\hat{u}\|^2 + \beta\|f\|^2 \\ &\qquad s.t.\ Ku = Mf + d, \end{aligned} \tag{52}$$

where

$$K = \begin{bmatrix} D & -L & & & \\ -L & D & -L & & \\ & -L & D & \ddots & \\ & & \ddots & \ddots & -L \\ & & & -L & D \end{bmatrix}, \tag{53}$$

the matrices D and L in K are 2-level SSS matrices, M is the 3D mass matrix that has the same structure with K, d is a vector that satisfies the given boundary condition. To compute the optimal solution of Example 5.1, system of the form (3) needs to be solved. Here we also study two types of preconditioners, the block-diagonal MSSS preconditioner and the global MSSS preconditioner.

5.1 Block-Diagonal Preconditioner

In this subsection, we test the block-diagonal preconditioner for big and middle β, then the block-diagonal preconditioner $\mathcal{P}_1$ (5) is chosen. Here the Schur complement is approximated by $\hat{K}M^{-1}\hat{K}^T$ where $\hat{K}$ is an approximation of K by MSSS matrix computations.

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

$$\begin{cases} S_k = D, & \text{if } k = 1, \\ S_k = D - LS_{k-1}^{-1}L, & \text{if } k \geq 2. \end{cases} \tag{54}$$

Since D and L are 2-level SSS matrices, S_k is also a 2-level SSS matrix. In the recurrence (54), both the 2-level and 1-level semiseparable orders of S_k increase as k goes up. Model order reduction for 2-level and 1-level SSS matrices are necessary, of which the model order reduction for 2-level SSS matrix is still an open problem. Here we use an alternative method to approximate the Schur complement with lower 2-level semiseparable order.

As pointed out in [42], for a symmetric positive definite matrix from the discretization of PDEs with constant coefficients, all its subsequent Schur complements are also symmetric positive definite and converge to a fixed point matrix S_∞ with a fast rate. In [14], Dewilde et al. used a hierarchical partitioning of for the 3D matrix K (53) and did computations on 2D matrices using the 1-level SSS matrix computations for preconditioning 3D Poisson equation on an $8 \times 8 \times 8$ regular grid. Due to the fact that 1-level SSS matrix computations were performed on 2D matrices, the linear computational complexity is lost. Note that there is no numerical experiment in [14] to study the performance of such preconditioning technique for a certain Krylov solver.

In this manuscript, we extend such method in [14] to the optimal control of 3D Poisson equation in the following ways. Instead of using the hierarchical partitioning of a 3D matrix, we use the 3-level SSS matrix formulation. This avoids cutting on "layers" that is introduced in [14] to bound the 2-level semiseparable order. We exploit the fast convergence property of the Schur complements of symmetric positive definite matrices to bound the 2-level semiseparable order. As analyzed in Section 3.1, the 1-level and 2-level semiseparable order both grow in computing the Schur complements in (54). To reduce the 1-level semiseparable order, we can apply the approximate balanced truncation or the Hankel blocks approximation that are introduced in Section 3.2. Since the model order reduction for 2-level SSS matrices is still an open problem, we use an alternative approach to bound the 2-level semiseparable order. This method exploits the fast convergence property of the Schur complements. We compute the Schur complements of the first k_r steps, where k_r is a small constant than can be chosen freely, using MSSS matrix computations. Then use the Schur complement at step k_r to replace the Schur complements afterwards, i.e., we have the following recursions for the Schur complements

$$\begin{cases} S_k = D, & \text{if } k = 1, \\ S_k = D - LS_{k-1}^{-1}L, & \text{if } 2 \leq k \leq k_r, \\ S_k = S_{k_r}, & \text{if } k > k_r. \end{cases} \tag{55}$$

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

In this part, we apply the block-diagonal MSSS preconditioner (5) and `MINRES` method to iteratively solve the saddle-point system. The computational results are reported in Table 15 - 17. Note that if the mesh size h is halved, the problem size grows by a factor of 8. Besides, there are three unknowns on each grid point. The 1-level semiseparable order is set to be 6 for all the numerical experiments in this part. The iterative solver is stopped if the 2-norm of the residual is reduced by a factor of 10^{-6}. The model order reduction is chosen as the Hankel blocks approximation method. The "preconditioning" column, "MINRES" column and the "total" column represent the same with the tables in Section 4.

Table 15: Block MSSS preconditioner for optimal control of 3D Poisson equation with $\beta = 10^{-1}$

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

Table 16: Block MSSS preconditioner for optimal control of 3D Poisson equation with $\beta = 10^{-2}$

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

Table 17: Block MSSS preconditioner for optimal control of 3D Poisson equation with $\beta = 10^{-3}$

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

According to the computational results for different β and h in Table 15 - 17, we can see that for fixed h and β, the number of iterations decreases as k_r goes up, and a small k_r is enough to compute an efficient preconditioner. Due to the growth of k_r, the time to compute the preconditioner increases. Since only the Schur complements in the first k_r steps are computed, the time to compute the preconditioner increases less than linear when halving the mesh size h. This is illustrated by the “preconditioning” columns in Table 15 - 17. Moreover, by choosing proper k_r, the block MSSS preconditioner also gives virtually mesh size independent convergence and regularization parameter almost independent convergence while the computational complexity is smaller than linear. This property is obtained by carefully checking the computational results given in Table 15 - 17.

5.2 Global Preconditioners

In the previous part, we study the performance of the block MSSS preconditioner for the optimal control of 3D Poisson equation. This is based on the fast convergence property of the Schur complements for symmetric positive definite matrix from the discretization of PDEs with constant coefficients. The fast convergence property is discussed in [42]. In this part, we are going to extend such property to the symmetric indefinite matrix from the discretization of PDEs with constant coefficients case. Even the analysis in [42] only holds for the symmetric positive definite matrix case, our numerical experiments illustrate that the fast convergence property of the Schur complements also hold for the symmetric indefinite matrix case.

The saddle-point system obtained by using the *discretize-then-optimize* approach has the following form,

$$\mathcal{A} = \begin{bmatrix} 2\beta M & 0 & -M \\ 0 & M & K^T \\ -M & K & 0 \end{bmatrix}, \tag{56}$$

where K is the stiffness matrix in (53), and M is the mass matrix. Since all these matrices are from the discretization of 3D PDEs, K and M have the same 3-level SSS structure as shown in (53). Here we can apply Lemma 3.1 again to permute the global saddle-point matrix $\mathcal{A}$ (56) into a global MSSS matrix $\bar{\mathcal{A}}$. The permuted global saddle-point matrix $\bar{\mathcal{A}}$ has the same MSSS structure with its MSSS blocks, i.e.,

$$\bar{\mathcal{A}} = \begin{bmatrix} \bar{D} & \bar{L} & & & \\ \bar{L} & \bar{D} & \bar{L} & & \\ & \bar{L} & \bar{D} & \ddots & \\ & & \ddots & \ddots & \bar{L} \\ & & & \bar{L} & \bar{D} \end{bmatrix}, \tag{57}$$

where $\bar{D}$ and $\bar{L}$ are obtained via Lemma 3.1. Note that $\bar{\mathcal{A}}$ is a symmetric indefinite matrix, its Schur complements in computing the LU factorization are also indefinite. To compute an LU factorization, the Schur complements are computed via the following recursions,

$$\begin{cases} \bar{S}_k = \bar{D}, & \text{if } k = 1, \\ \bar{S}_k = \bar{D} - \bar{L}\bar{S}_{k-1}^{-1}\bar{L}, & \text{if } k \geq 2. \end{cases} \tag{58}$$

Due to the indefiniteness of $\bar{D}$, the Schur complements are also indefinite. Since the model order reduction for 2-level SSS matrix is still an open problem, we apply the same method introduced in the previous part. We compute the Schur complements of the first k_r steps, and use the Schur complemnt at step k_r to approximate the Schur complements afterwards. This gives,

$$\begin{cases} \bar{S}_k = \bar{D}, & \text{if } k = 1, \\ \bar{S}_k = \bar{D} - \bar{L}\bar{S}_{k-1}^{-1}\bar{L}, & \text{if } 2 \leq k \leq k_r, \\ \bar{S}_k = \bar{S}_{k_r}, & \text{if } k > k_r. \end{cases} \tag{59}$$

This approximation only computes the first k_r steps of the Schur complements, this bounds the 2-level semiseparable order while the 1-level semiseparable order can be reduced by the approximate balanced truncation or the Hankel blocks approximation that are introduced in Section 3.1. By using this approximation for the permuted global system, we can compute the global MSSS preconditioner and apply it to iteratively solve the saddle-point system. The computational results are given in Table 18 - 20, where the columns have the same representation with Table 15 - 17.

Table 18: Global MSSS preconditioner for the optimal control of 3D Poisson equation for $\beta = 10^{-1}$

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

Table 19: Global MSSS preconditioner for the optimal control of 3D Poisson equation for $\beta = 10^{-2}$

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

Table 20: Global MSSS preconditioner for the optimal control of 3D Poisson equation for $\beta = 10^{-3}$

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

Since we just compute the first few steps of the Schur complements for the permuted saddle-point system by MSSS matrix computations, the computational complexity of the global MSSS preconditioner is smaller than than linear. This is stated by the "preconditioning" columns for the same k_r in Table 18 - 20. The number of iterations decreases as k_r goes up for the same β and h. By using a small k_r, we have already reduced the number of iterations significantly by the global MSSS preconditioner compared with the block-diagonal MSSS preconditioner. Moreover, the global MSSS preconditioner gives virtually mesh size h and regularization parameter β independent convergence for properly chosen k_r.

Remark 5.1. *Compare the results for the block-diagonal MSSS preconditioner in Table 15 - 17 with that of the global MSSS preconditioner in Table 18 - 20, the global MSSS preconditioner reduces the number of iterations significantly. Even though more time is spent in computing the global MSSS preconditioner for the same group of numerical experiment, the time to iteratively solve the preconditioned system is much reduced due to the fact that fewer iterations are needed. Moreover, the total computation time for the global MSSS preconditioner is less than that for the block MSSS preconditioner.*

Remark 5.2. *Since there is no efficient model order reduction to reduce the 2-level semiseparable order, the 2-level semiseparable order continues growing before k_r is reached. It is shown in Table 18 - 20, when k_r goes from 3 to 4 for $h = 2^{-6}$, the time to compute the global MSSS preconditioner increases dramatically. This is due to the fact that when k_r goes from 3 to 4, the 2-level semiseparable order is not bounded by a small number, but by a moderate constant. However, the computational complexity increases briefly bigger than linear when h goes from 2^{-5} to 2^{-6} for $k_r = 4$. Moreover, the global MSSS preconditioner already gives satisfactory performance by choosing $k_r = 3$ for $h = 2^{-6}$.*

6 Conclusions

In this manuscript, we have studied the multilevel sequentially semiseparable (MSSS) preconditioners for saddle-point systems that arise from the PDE-constrained optimization problems. By exploiting the MSSS structure of the blocks of the saddle-point system, we are able to construct the preconditioners and solve the preconditioned system in linear computational complexity for 2D problems while for 3D problems we have computational complexity smaller than linear. To reduce the computational complexity of computing the preconditioners, we have proposed a new model order reduction algorithm based on the approximate balanced truncation for SSS matrices. We evaluated the performance of the new model order reduction algorithm by comparing with the standard model order reduction algorithm, which is called the Hankel blocks approximation. Numerical experiments illustrate that our model order reduction algorithm is computationally cheaper than the standard method. Besides, it shows that for the optimal control of 2D PDEs, the global preconditioner reduced the number of iterations significantly compared with the block preconditioners. Both preconditioners give mesh size independent convergence and have linear computational complexity. Moreover, the global MSSS preconditioner yields regularization parameter independent convergence while the block MSSS preconditioner does not have this property.

For PDE-constrained optimization problem in 3D, since efficient model order reduction algorithm for 2- or higher- level SSS matrices is still an open problem, we apply an alternative approach to bound the 2-level semiseparable order. Numerical experiments also illustrate that the global MSSS preconditioner gives mesh size and regularization parameter virtually independent convergence while the block MSSS preconditioner just yields mesh size almost independent convergence. Moreover, the computational complexity for both preconditioners by MSSS matrix computations is smaller than linear. To further improve the convergence of the MSSS preconditioners for optimal control of 3D PDEs, the computational complexity gets slightly bigger than linear.

The next step of this research will be focused on applying this preconditioning technique to the optimal control of the Navier-Stokes equation. This has a wide range of applications such as control a wind farm 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 Martin B. van Gijzen. IDR(s): A family of simple and fast algorithms for solving large nonsymmetric systems of linear equations. *SIAM Journal on Scientific Computing*, 31(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] P. Dewilde, H.Y. Jiao, and S. Chandrasekaran. Model reduction in symbolically semi-separable systems with application to preconditioners for 3D sparse systems of equations. In *Characteristic Functions, Scattering Functions and Transfer Functions*, volume 197 of *Operator Theory: Advances and Applications*, pages 99–132. Birkhäser Basel, 2010.

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

[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] 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.

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

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

[21] Y. Qiu, Martin B. van Gijzen, Jan-Willem van Wingerden, M. Verhaegen, and C. Vuik. Efficient preconditioners for PDE-constrained optimization problems with a multilevel sequentially semiseparable matrix structure. Technical Report 13-04, Delft Institution of Applied Mathematics, Delft University of Technology, 2013. available at http://ta.twi.tudelft.nl/nw/users/yueqiu/publications.html.

[22] Y. Qiu, Martin B. van Gijzen, Jan-Willem van Wingerden, M. Verhaegen, and C. Vuik. Evaluation of multilevel sequentially semiseparable preconditioners on CFD benchmark problems using IFISS. Technical Report 13-11, Delft Institution of Applied Mathematics, Delft University of Technology, 2013. available at http://ta.twi.tudelft.nl/nw/users/yueqiu/publications.html.

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

[24] Yvan Notay. A new analysis of block preconditioners for saddle point problems. *SIAM Journal on Matrix Analysis and Applications*, 35(1):143–173, 2014.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

[29] Alle-Jan van der Veen. *Time-Varying System Theory and Computational Modeling*. PhD thesis, Delft University of Technology, 1993.

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

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

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

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

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

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

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

[37] 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.

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

[39] Dragan Žigić and Layne T. Watson. Contragredient transformations applied to the optimal projection equations. *Linear Algebra and its Applications*, 188-189(0):665–676, 1993.

[40] Y. Chahlaoui. *Low-rank Approximation and Model Reduction*. PhD thesis, Université catholique de Louvain, December 2003.

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

[42] 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 ([15]). *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 $\Gamma_N = \{x = 0, 0 \leq y \leq 1\}$ 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

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

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 21 - 26.

Table 21: 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 22: 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 23: 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 24: 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

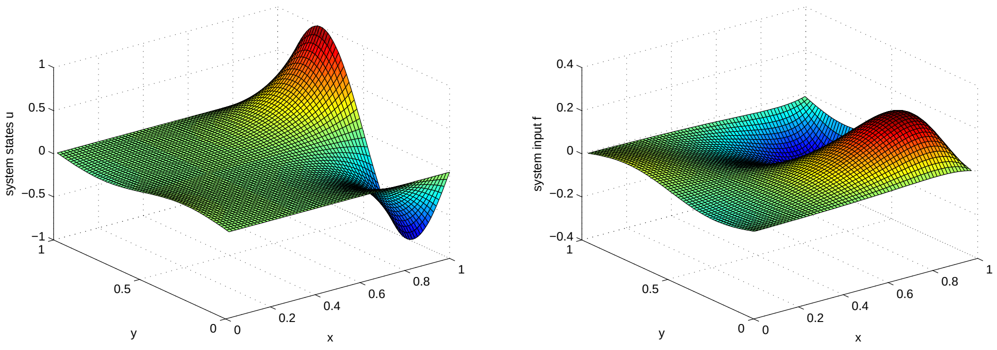
Table 25: By approximate balanced truncation for $\beta = 10^{-3}$

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 26: 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 5(a) and Figure 5(b).



(a) Optimal system states u.

(b) Optimal system input f.

Figure 5: Solution of the system states and input when $\beta = 10^{-2}$ and $h = 2^{-6}$.

Remark A.1. *Table 21 - 26 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.*

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 27 - 36.

Table 27: 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 28: 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 29: 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 30: 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 31: 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 32: 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 33: 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 34: 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 35: 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 36: 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.2. *From Table 27-36, 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.3. *From the preconditioning columns of Table 27-36 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.4. *Compare the computational results of the block-diagonal preconditioner $\mathcal{P}_1$ and MINRES in Table 2-9 with that of the block lower-triangular preconditioner $\mathcal{P}_2$ and IDR(s) in Table 27-36, 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 37-38.

Table 37: 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 38: 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. 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 39-42 show the results of the block-diagonal preconditioner $\mathcal{P}_1$ in (5) for the optimal control of the Poisson equation.

Table 39: With $\mathcal{P}_1$ in (5) 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 40: With $\mathcal{P}_1$ in (5) 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 41: With $\mathcal{P}_1$ in (5) 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 42: With $\mathcal{P}_1$ in (5) 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 39-42, 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 [20], for "smaller" β ($\beta \leq 10^{-5}$), the block-diagonal conditioner could be chosen as

$$\mathcal{P}_1 = \begin{bmatrix} 2\beta M & & \\ & M & \\ & & \frac{1}{2\beta} M \end{bmatrix} \tag{60}$$

With this preconditioner, the computational results are shown in Table 43-44. The maximum number of iterations is set to 100.

Table 43: With $\mathcal{P}_1$ in (60) 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 44: With $\mathcal{P}_1$ in (60) 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

As shown in Table 43-44, the block diagonal preconditioner $\mathcal{P}_1$ in (60) 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 (5) by the approximate balanced truncation are shown in Table 45-46.

Table 45: With $\mathcal{P}_1$ in (5) 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 46: With $\mathcal{P}_1$ in (5) 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 47-51.

Table 47: 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 48: 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 49: 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 50: 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 51: 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.1. *Table 47-51 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.2. *As shown in Table 47-51, 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.3. *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 47-51. 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 52-55.

Table 52: 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 53: 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 54: 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 55: 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.4. *In Table 54 and 55, 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.5. *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 52-55.*

Remark B.6. *According to Table 52-55, 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.*