A ROM-accelerated parallel-in-time preconditioner for solving all-at-once systems in unsteady convection-diffusion PDEs

Jun Liu[a], Zhu Wang[b,*]

[a]*Department of Mathematics and Statistics, Southern Illinois University Edwardsville, Edwardsville, IL 62026, USA.*
[b]*Department of Mathematics, University of South Carolina, Columbia, SC 29208, USA.*

Abstract

In this paper we propose a model reduction technique to speed up the diagonalization-based parallel-in-time (ParaDIAG) preconditioner, for iteratively solving all-at-once systems from evolutionary PDEs. In particular, we use the reduced basis method to seek a low-dimensional approximation to the sequence of complex-shifted systems arising from Step-(b) of the ParaDIAG preconditioning procedure. Different from the standard reduced order modeling that uses the separation of offline and online stages, we have to build the reduced order model (ROM) online for the considered systems at each iteration. Therefore, several heuristic acceleration techniques are introduced in the greedy basis generation algorithm, that is built upon a residual-based error indicator, to further boost up its computational efficiency. Several numerical experiments are conducted, which illustrate the favorable computational efficiency of our proposed ROM-accelerated ParaDIAG preconditioner, in comparison with the multigrid-based one.

Keywords: Parallel-in-time, α-circulant preconditioner, model order reduction, reduced basis method, FGMRES

1. Introduction

With the advent of massively parallel processors, various parallel-in-time (PinT) algorithms have been developed in the last few decades for simulating time-dependent partial differential equations (PDEs) [16, 45]. Such PinT algorithms with successful implementations can provide significant speed up over the traditional sequential time-stepping schemes. However, the design of effective PinT algorithms is more challenging than their counterparts in space, such as spatial domain decomposition, because of the sequential nature of forward time evolution/marching. Thus far, there are several different types of PinT algorithms in literature[1], including the parareal algorithm [36], the multigrid reduction in time (MGRIT) algorithm [14], the space-time parallel multigrid algorithm[19], and the more recent diagonalization-based ParaDIAG algorithm [18], etc. The mechanism of each method varies greatly, which leads to significant difference with respect to application scopes, convergence properties and parallel efficiency. Among them, the ParaDIAG algorithms [18] are built upon the diagonalization of the time discretization matrix or its approximations within the all-at-once system arising from solving all the time steps simultaneously. Extensive numerical results given in [18, 22] indicate that the ParaDIAG algorithms have a very promising parallel efficiency for both parabolic [8, 35, 63] and hyperbolic PDEs [17, 37, 61]. Especially, there is a sequence of sparse complex-shifted linear systems to be solved in the algorithm. These linear systems are independent, thus can be solved by parallel computing. However, the size of the individual linear systems can be tremendous

*Corresponding author
Email addresses: juliu@siue.edu (Jun Liu), wangzhu@math.sc.edu (Zhu Wang)

[1]For an extensive list of PinT-related literature, see the website http://parallel-in-time.org.

in real-world applications, as it is determined by the number of spatial degrees of freedom. Solving these systems represents the major computational cost in numerical simulations of evolutionary PDEs by the ParaDIAG algorithms. Therefore, in order to accelerate the ParaDIAG algorithms, we propose to synthesize it with model reduction techniques.

The reduced order models (ROMs) have been widely used in approximating large-scale linear and nonlinear dynamical systems, with successful applications to numerical simulations, control and optimization problems. Common model reduction techniques include but not limited to the reduced basis method (RBM), proper orthogonal decomposition (POD), dynamical model decomposition (DMD), and interpolatory methods [1, 2, 7, 27, 28, 49]. Although they provide quite different ways to construct ROMs, they all are date-driven approaches and usually share a common computational strategy: (i) finding a low-dimensional approximation to the solution manifold of the underlying system's dynamics and building a ROM at an offline stage; (ii) using the ROM for fast simulations at an online stage. The efficacy of the low-dimensional approximation can be quantified by the Kolmogorov n-width [5, 48]. Successful ROMs can dramatically reduce the online simulation cost. In the context of parallel-in-time simulations with the parareal method, the ROM and FOM has been used together in [9, 15, 38], where the ROM on the fly is regarded as a coarse propagator. Recently, the ROMs have also been integrated with preconditioners in Krylov-subspace iterative solvers. For instance, preconditioned Conjugate Gradient (PCG) methods are developed using a POD-based preconditioner in [39, 47] and a RBM-based one in [44], which are able to speed up the traditional iterative methods in serial computing.

In this work, we pursue in this direction and improve the efficiency of the ParaDIAG algorithms by online reduced order modeling. We focus on addressing the sequence of linear systems in Step-(b) of ParaDIAG via building their low-dimensional approximation by the RBM. In particular, a greedy algorithm is used to find a reduced basis and the Galerkin projection is applied in constructing the ROM. Once the ROM is constructed, the computational complexity for solving the systems only depends on the dimension of the ROM, which is much smaller than the number of degrees of freedom in space and time of the full order models. Since the build stage is part of the approach, we propose several algorithmic improvements on the greedy basis generation. The proposed approach leads to significant computational savings in Step-(b) and, as a consequence, notably decreases the overall computational cost of the ParaDIAG algorithms. Our current paper focuses on the serial computing case with Step-(b) dominates the total computations.

The rest of the paper is organized as follows. In the next section, a block α-circulant type ParaDIAG algorithm is briefly reviewed for solving an unsteady convection-diffusion equation based on a standard upwind finite difference scheme. Our proposed ROM-based ParaDIAG preconditioner for approximately solving the linear systems in Step-(b) is introduced in Section 3, where a greedy algorithm for selecting the reduced basis is explained in detail and brief discussions on computational complexity are presented. Extensive 1D and 2D numerical examples are presented in Section 4 to demonstrate the promising performance of our proposed ROM-based ParaDIAG preconditioner in contrast with the multigrid-based ParaDIAG preconditioner. Finally, some conclusions are drawn in the last section.

2. The ParaDIAG algorithm for fully discretized evolutionary PDEs

In this work, we consider the unsteady convection-diffusion equation [43]:

$$\begin{cases} u_t(\mathbf{x},t) = \nabla\cdot(a(\mathbf{x})\nabla u) - \boldsymbol{c}(\mathbf{x})\cdot\nabla u(\mathbf{x},t) + f(\mathbf{x},t), & \text{in } \Omega\times(0,T], \\ u(\mathbf{x},t) = g(\mathbf{x},t), & \text{on } \partial\Omega\times(0,T], \\ u(\mathbf{x},0) = u_0(\mathbf{x}), & \text{in } \Omega\times\{0\}, \end{cases} \tag{1}$$

where $\Omega\subset\mathbb{R}^d$ is the spatial domain, $a(\mathbf{x})\geq a_0>0$ is a variable diffusion coefficient, $\boldsymbol{c}(\mathbf{x})$ is a convection or velocity field, f is a source term, g is a Dirichlet boundary condition, and u_0 is an initial condition. To

illustrate the ParaDIAG algorithm, we restrict ourselves to the case in which $\Omega = (0,1)^2$ ($d = 2$) and discretize the space and time using finite difference methods (FDM). However, we emphasize the proposed algorithm can be applied to problems defined in a one-dimensional or three-dimensional domain; it can also be applied to problems defined in an irregular domain for which finite element method can be employed for spatial discretization. Furthermore, we assume all data are regular enough such that the solution is unique and sufficiently smooth to assure the convergence of finite difference schemes. We refer to the monographs [26, 43, 51] for more discussion on various different discretization schemes. For readers' convenience and following the conventions in [23], we summarize in Table 1 the list of frequently used operator and MATLAB notations across the paper. We will also use the well-known matrix Kronecker product (denoted by $\otimes$) property [23, p. 28] that $\texttt{vec}(AXB) = (B^{\mathsf{T}} \otimes A)\texttt{vec}(X)$ for any matrices A, B, X of compatible sizes.

Table 1: List of frequently used operator and MATLAB notations (mostly following [23])

Notation	Definition or Meaning	Notation	Definition or Meaning
A^{T}	non-conjugate transpose of A	$A(:, j)$	j-th column of the matrix A
A^*	complex conjugate transpose of A	$A(i, :)$	i-th row of the matrix A
vec	reshape a matrix into a vector	$A(i:j, :)$	i-th till j-th rows of the matrix A
mat	reshape a vector into a matrix	$Z = \text{diag}(\boldsymbol{z})$	Z is a diagonal matrix with $Z_{k,k} = \boldsymbol{z}_k$
$z^{\pm}$	$z^+ := \max\{z, 0\}$, $z^- := \min\{z, 0\}$	$[A\ \ B]$	concatenate matrices A and B horizontally
h, τ	spatial and time mesh sizes	$\begin{bmatrix} A \\ B \end{bmatrix}$	concatenate matrices A and B vertically
I_h, I, I_t	identity matrices of various sizes	$\begin{bmatrix} A & C \\ B & D \end{bmatrix}$	concatenate matrices A,B,C, and D
Re(z)	real part of z	$\Theta \backslash \Theta_{in}$	the set of elements in Θ that are not in Θ_{in}.

For the discretization of (1), we use the backward Euler scheme in time, the conservative central finite difference scheme for the diffusion term, and the upwind scheme for the convection term [26, 34]. Given a positive integer N, let $h = 1/(N+1)$ be the spatial mesh size and the 2D square domain $\overline{\Omega} = [0,1]^2$ be partitioned uniformly by grid points $(x_i = ih, y_j = jh)$ with $0 \le i, j \le N+1$. We denote the set of all interior grid points by $\Omega_h = \{(x_i, y_j)\}_{i,j=1}^{i,j=N}$ and also define the cell center points $(x_{i\pm1/2} = (i \pm 1/2)h, y_{j\pm1/2} = (j \pm 1/2)h)$ with $1 \le i, j \le N$. Given the final time $T > 0$ and another positive integer K, let $\tau = T/K$ be the time step size and the time domain $[0, T]$ be uniformly divided by grid points $t_k = k\tau$ with $0 \le k \le K$. Let $U_{i,j}^k$ represent the finite difference approximation of $u(x_i, y_j, t_k)$. Assuming $\boldsymbol{c}(\mathbf{x}) = (p(x,y), q(x,y))$, the full finite difference discretization of the PDE (1) on the set Ω_h of interior grid points reads: for $1 \le i, j \le N$, $1 \le k \le K$,

$$
\begin{aligned}
\frac{U_{i,j}^k - U_{i,j}^{k-1}}{\tau} =& \frac{1}{h}\left(a_{i+1/2,j} \frac{U_{i+1,j}^k - U_{i,j}^k}{h} - a_{i-1/2,j} \frac{U_{i,j}^k - U_{i-1,j}^k}{h} \right) \\
&+ \frac{1}{h}\left(a_{i,j+1/2} \frac{U_{i,j+1}^k - U_{i,j}^k}{h} - a_{i,j-1/2} \frac{U_{i,j}^k - U_{i,j-1}^k}{h} \right) \\
&- \left(p_{i,j}^+ \frac{U_{i,j}^k - U_{i-1,j}^k}{h} + p_{i,j}^- \frac{U_{i+1,j}^k - U_{i,j}^k}{h} \right) \\
&- \left(q_{i,j}^+ \frac{U_{i,j}^k - U_{i,j-1}^k}{h} + q_{i,j}^- \frac{U_{i,j+1}^k - U_{i,j}^k}{h} \right) + f_{i,j}^k,
\end{aligned} \tag{2}
$$

where $a_{i\pm1/2,j\pm1/2} = a(x_{i\pm1/2}, y_{j\pm1/2})$, $p_{i,j} = p(x_i, y_j)$, $q_{i,j} = q(x_i, y_j)$, and $f_{i,j}^k = f(x_i, y_j, t_k)$. Under suitable regularity assumptions [26, Thm. 5.17], the above scheme (2) has a first-order accuracy in space and time.

Let $\mathbf{U}^k, \mathbf{F}^k$ be column vectors containing the ordered values of $U_{i,j}^k, f_{i,j}^k$ over all the interior grids in Ω_h

at time t_k, respectively. Upon enforcing the Dirichlet boundary condition by shifting boundary nodes to the right-hand-side, the above finite difference scheme (2) can be written into the following sequential time stepping formulation (marching from the given $\mathbf{U}^0$ to $\mathbf{U}^1$, then to $\mathbf{U}^2$, till the last time step $\mathbf{U}^K$):

$$(\tau^{-1} I_h - L_h)\mathbf{U}^k = \tau^{-1}\mathbf{U}^{k-1} + \mathbf{F}^k + \mathbf{G}^k, \quad 1 \le k \le K, \tag{3}$$

where $I_h \in \mathbb{R}^{N^2 \times N^2}$ is an identity matrix, L_h is the coefficient matrix representing the discretized spatial differential operators (involving a, p, q), $\mathbf{U}^0$ is given by the initial condition $u_0(x,y)$, and $\mathbf{G}^k$ comes from the Dirichlet boundary conditions. Explicitly, the coefficient matrix L_h has the following expression in terms of Kronecker products:

$$\begin{aligned} L_h = &-\frac{1}{h^2}\left[(I \otimes D^{\mathsf{T}})A_x(I \otimes D) + (D^{\mathsf{T}} \otimes I)A_y(D \otimes I)\right] \\ &-\frac{1}{h}\left[P^+(I \otimes \widehat{D}) + P^-(I \otimes \check{D}) + Q^+(\widehat{D} \otimes I) + Q^-(\check{D} \otimes I)\right], \end{aligned} \tag{4}$$

where $I \in \mathbb{R}^{N \times N}$ is an identity matrix,

$$D = \begin{bmatrix} 1 & & & & \\ -1 & 1 & & & \\ 0 & -1 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 0 & -1 & 1 \\ & & & 0 & -1 \end{bmatrix} \in \mathbb{R}^{(N+1)\times N}, \quad \widehat{D} = D(1:N,:), \quad \check{D} = D(2:N+1,:),$$

$$A_x = \operatorname{diag}\left(\operatorname{vec}\left([a_{i-1/2,j}]_{i=1,j=1}^{N+1,N}\right)\right), \quad A_y = \operatorname{diag}\left(\operatorname{vec}\left([a_{i,j-1/2}]_{i=1,j=1}^{N,N+1}\right)\right),$$

$$P^{\pm} = \operatorname{diag}\left(\operatorname{vec}\left([p_{i,j}^{\pm}]_{i=1,j=1}^{N,N}\right)\right), \quad Q^{\pm} = \operatorname{diag}\left(\operatorname{vec}\left([q_{i,j}^{\pm}]_{i=1,j=1}^{N,N}\right)\right).$$

Here $Z = [z_{i,j}]_{i=1,j=1}^{m,n}$ defines an $m \times n$ matrix with $z_{i,j}$ as its (i,j)-th entry, $\texttt{vec}(Z)$ reshapes an $m \times n$ matrix Z into a column vector $\mathbf{Z} \in \mathbb{R}^{\mathbf{mn}}$ by stacking all columns together one after another, and $\operatorname{diag}(\mathbf{Z})$ denotes a diagonal matrix with the vector $\mathbf{Z}$ as its main diagonal entries. Here $\widehat{D}$ and $\check{D}$ are defined via MATLAB syntax of selecting the first and last N rows of the $(N+1) \times N$ rectangular difference matrix D, respectively.

Different from the sequential time stepping method marching from the initial time step to the final, the all-at-once approach tries to simultaneously solve all time steps in one-shot without explicit time marching. More specifically, by stacking all the K systems in (3) together and using Kronecker product notation, we can obtain the following all-at-once non-symmetric $N^2K \times N^2K$ linear system

$$(\tau^{-1} B \otimes I_h + I_t \otimes (-L_h))\mathbf{U} = \mathbf{F}, \tag{5}$$

where $I_t \in \mathbb{R}^{K \times K}$ is an identity matrix,

$$B = \begin{bmatrix} 1 & & & & \\ -1 & 1 & & & \\ 0 & -1 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 0 & -1 & 1 \end{bmatrix} \in \mathbb{R}^{K \times K}, \quad \mathbf{U} = \begin{bmatrix} \mathbf{U}^1 \\ \mathbf{U}^2 \\ \mathbf{U}^3 \\ \vdots \\ \mathbf{U}^K \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} (\mathbf{F}^1 + \mathbf{G}^1) + \frac{1}{\tau}\mathbf{U}^0 \\ (\mathbf{F}^2 + \mathbf{G}^2) \\ (\mathbf{F}^3 + \mathbf{G}^3) \\ \vdots \\ (\mathbf{F}^K + \mathbf{G}^K) \end{bmatrix}. \tag{6}$$

By exploiting the Toeplitz matrix B due to the uniform time step size, the Strang-type block α-circulant

preconditioner [35, 41] for (5) has the following Kronecker product form

$$\mathscr{P}_\alpha := \tau^{-1} C_\alpha \otimes I_h + I_t \otimes (-L_h), \tag{7}$$

where the Toeplitz matrix B is replaced by a Strang-type α-circulant matrix (with $\alpha \in (0,1]$)

$$C_\alpha = \begin{bmatrix} 1 & & & & -\alpha \\ -1 & 1 & & & \\ 0 & -1 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 0 & -1 & 1 \end{bmatrix} \in \mathbb{R}^{K\times K}. \tag{8}$$

It is shown in [35] that, for pure diffusion equations, the GMRES equipped with the above preconditioner $\mathscr{P}_\alpha$ for a sufficiently small $\alpha = O(\sqrt{\tau}) \in (0,1)$ can solve (5) effectively and it achieves a provable mesh-independent convergence rate. Here C_α is indeed a rank-one perturbation of B satisfying $\lim_{\alpha\to 0}\|C_\alpha - B\| = 0$, which implies the preconditioner $\mathscr{P}_\alpha$ is expected to lead faster convergence rate as α gets smaller if not considering the possible round-off errors due to inverting $\mathscr{P}_\alpha$ during the preconditioning step. In practice, a small $\alpha = 0.01$ seems to be very effective.

Let $\mathbb{F} = \frac{1}{\sqrt{K}}\left[\omega^{(l_1-1)(l_2-1)}\right]_{l_1,l_2=1}^{K}$ (with $\mathrm{i} = \sqrt{-1}$ and $\omega = e^{\frac{2\pi \mathrm{i}}{K}}$) be the discrete Fourier matrix and define a diagonal matrix $\Gamma_\alpha = \mathrm{diag}\left(1, \alpha^{\frac{1}{K}}, \cdots, \alpha^{\frac{K-1}{K}}\right)$ for $\alpha \in (0,1)$. Let $\mathbb{F}^*$ represents the complex conjugate transpose of $\mathbb{F}$. The above so-called α-circulant matrix C_α can be explicitly diagonalized [6] according to its spectral decomposition

$$C_\alpha = V \Lambda V^{-1}, \tag{9}$$

where $V = \Gamma_\alpha^{-1}\mathbb{F}^* \in \mathbb{C}^{K\times K}$ and $\Lambda = \mathrm{diag}(d_1, \cdots, d_n, \cdots, d_K) := \mathrm{diag}\left(\sqrt{K}\mathbb{F}\Gamma_\alpha C_\alpha(:,1)\right)$ with $C_\alpha(:,1)$ being the first column of C_α. More specifically, let $\theta_n = 2(n-1)\pi/K \in [0, 2\pi)$ with $1 \le n \le K$, the explicit expressions for the K complex eigenvalues of C_α are

$$d_n = 1 - \alpha^{\frac{1}{K}} e^{\mathrm{i}\theta_n} = (1 - \alpha^{\frac{1}{K}}\cos\theta_n) - \mathrm{i}\alpha^{\frac{1}{K}}\sin\theta_n, \quad 1 \le n \le K. \tag{10}$$

With the above explicit diagonalization $C_\alpha = V\Lambda V^{-1}$, we can easily factorize $\mathscr{P}_\alpha$ in into

$$\mathscr{P}_\alpha = \underbrace{(V \otimes I_h)}_{\text{Step-(a)}} \underbrace{\left(\tau^{-1}\Lambda \otimes I_h + I_t \otimes (-L_h)\right)}_{\text{Step-(b)}} \underbrace{(V^{-1} \otimes I_h)}_{\text{Step-(c)}},$$

where the matrix of Step-(b) has the following block diagonal structure

$$\tau^{-1}\Lambda \otimes I_h + I_t \otimes (-L_h) = \begin{bmatrix} (d_1/\tau)I_h - L_h & & & & \\ & \ddots & & & \\ & & (d_n/\tau)I_h - L_h & & \\ & & & \ddots & \\ & & & & (d_K/\tau)I_h - L_h \end{bmatrix}.$$

Such a block diagonal structure is the key for designing the diagonalization-based ParaDIAG algorithms. Suppose the preconditioner $\mathscr{P}_\alpha$ is used in a preconditioned Krylov subspace method, it is required to compute or approximate the preconditioning step $\mathscr{P}_\alpha^{-1}s$ at each iteration, where $s \in \mathbb{R}^{N^2K}$ is the residual vector.

Let $S := \mathtt{mat}(s) \in \mathbb{R}^{N^2 \times K}$, the preconditioning step $z := \mathscr{P}_\alpha^{-1} s$ can be implemented via the following 3 steps:

$$\begin{aligned} &\text{Step-(a)} \quad S_1 = S(V^{-1})^{\mathsf{T}}, \\ &\text{Step-(b)} \quad S_2(:,n) = ((d_n/\tau)I_h - L_h)^{-1} S_1(:,n), \quad n = 1,2,\ldots,K, \\ &\text{Step-(c)} \quad z = \mathtt{vec}(S_2 V^{\mathsf{T}}), \end{aligned} \tag{11}$$

where $S_1(:,n)$ and $S_2(:,n)$ denote the n-th column of S_1 and S_2, respectively. Here we used the fact $\mathtt{vec}(AXB) = (B^{\mathsf{T}} \otimes A)\mathtt{vec}(X)$ to avoid the explicit use of the Kronecker products.

Since $V = \Gamma_\alpha^{-1}\mathbb{F}^*$ and hence $V^{-1} = (\mathbb{F}^*)^{-1}\Gamma_\alpha = \mathbb{F}\Gamma_\alpha$, Steps-(a) and (c) in (11) can be computed efficiently via FFT in time direction at the computational complexity $\mathscr{O}(N^2 K \log K)$ operations. Step-(b) needs to solve K complex-shifted elliptic systems of size $N^2 \times N^2$ with different right-hand-sides. Since these systems are independent of each other, parallel computing can be used to significantly reduce the wall-clock time. If N is large, these systems can be costly if solved by direct sparse solvers. Therefore, another efficient iterative solver has to be applied in Step-(b). This essentially leads to inexact GMRES [58] or flexible GMRES (FGMRES) [52, 57]. For instance, the authors in [35] employ one geometric multigrid V-cycle with ILU smoother to approximately solve the sparse linear systems in Step-(b), which needs slightly more outer GMRES iterations than solving these systems by the costly sparse direct solver. If the preconditioned GMRES converges in l iterations for a given tolerance, the overall computational cost of such a multigrid-based ParaDIAG (ParaDIAG-MG) preconditioner for solving (5) is about $\mathscr{O}(N^2 K \log K + l N^2 K)$ operations. Other efficient iterative solvers, such as domain decomposition algorithms [20, 33, 59], can also be used in Step-(b), but the overall computational complexity remains high. Thus, it is natural to consider model reduction techniques to derive a low-dimensional surrogate model for these systems that could greatly improve the computational efficiency.

2.1. A motivating example

Before presenting the proposed ROM-based solver for Step-(b), we motivate it by considering the heat equation with a constant diffusion coefficient, defined on the unit square domain. In particular, the equation is obtained from (1) by setting $a(x,y) = \varepsilon = 0.01$, $\boldsymbol{c}(x,y) = (0,0)$, $f(x,y,t) = 0$, $g(x,y,t) = 0$ and $u_0(x,y) = x(x-1)y(y-1)$:

$$\begin{cases} u_t = \varepsilon \Delta u, & \text{in } \Omega \times (0,T), \\ u = 0, & \text{on } \partial\Omega \times (0,T), \\ u = x(x-1)y(y-1), & \text{in } \Omega \times \{0\}, \end{cases} \tag{12}$$

with $\varepsilon = 0.01$ and the final time $T = 10$. The known exact solution is used for measuring approximation errors. We set $h = \tau = 1/64$ in the FDM discretization and use one multigrid V-cycle to approximately solve the linear systems of Step-(b) in serial. The FGMRES preconditioned by the ParaDIAG-MG with ILU smoothing converges after 4 iterations, which achieves the accuracy 5.7×10^{-5}. Meanwhile, Step-(b) dominates the total computational cost: the simulation is completed in 6.2 seconds, while 5.5 s of them is spent in solving Step-(b) that takes over 88% of the total CPU time. Hence the central task for improving the overall efficiency is to speed up the system solving in Step-(b).

The Step-(b) outputs S_2 during the four iterations are shown in Fig. 1: real part in the left column and imaginary part in the middle column, where relatively large structures are observed mainly near both ends. We perform the singular value decomposition (SVD) of S_2 and display the singular values of S_2 greater than 10^{-15} in the right column of Fig. 1. It is seen that the singular values decay quickly, which indicates the solution manifolds have low-dimensional structures. This further motivates us to use a model reduction technique such as RBM to extract the low-rank trail spaces and accelerate the computation of Step-(b) by solving reduced order systems. We remark that the exponential or algebraic decay rate of singular values

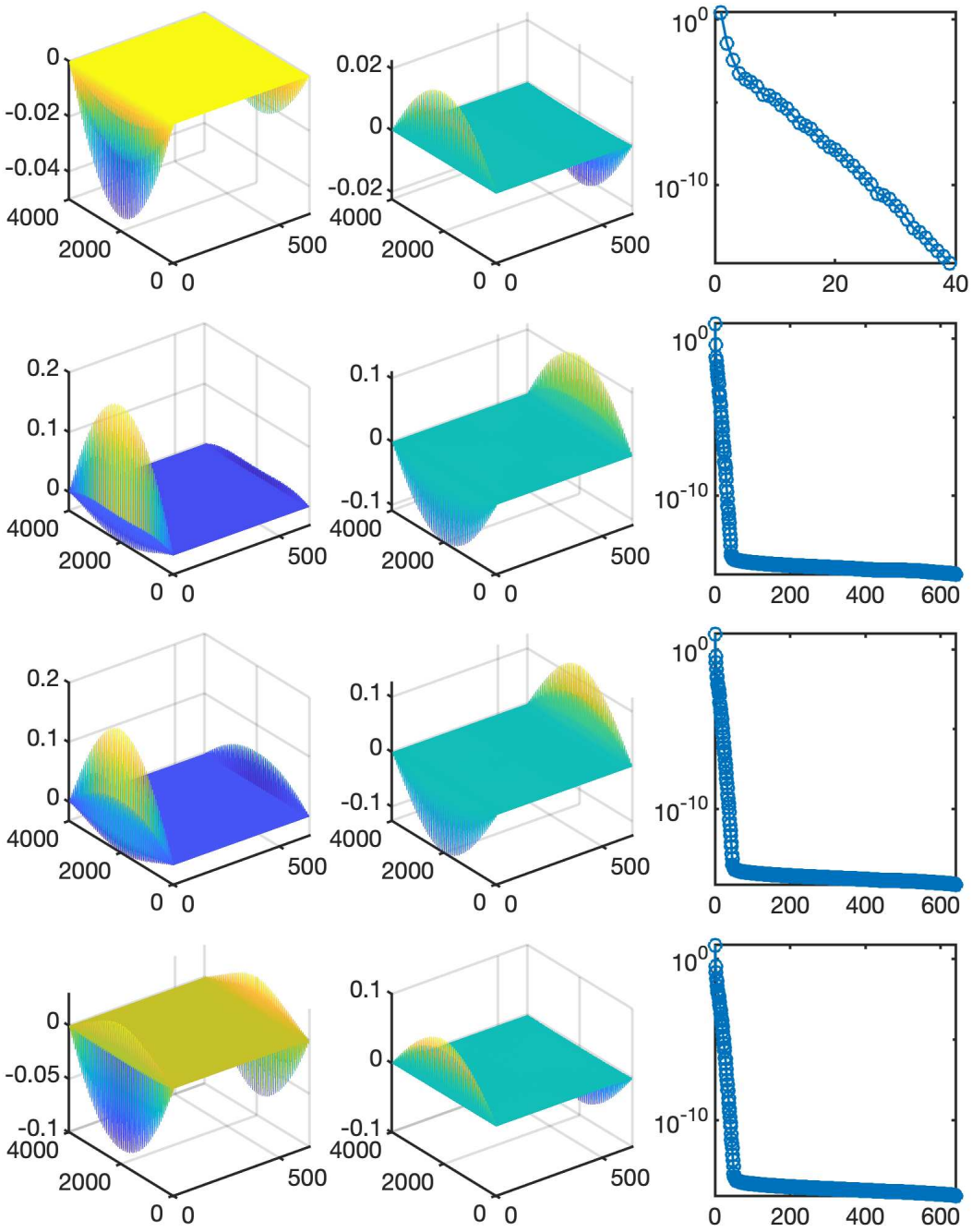

Figure 1: The visualization of Step-(b) outputs S_2 at iterations 1 to 4 (from top to bottom): (left column) real part of S_2, (middle column) imaginary part of S_2, (right column) singular values of S_2 greater than 10^{-15}.

of S_2 highly depends on the underlying physical model, which usually shows much slower decay rate in convection-dominated problems.

3. ROM-based Fast Solver for Step-(b)

The independent linear systems to be solved in Step-(b) can be regarded as discrete systems associated to a parametric PDE (pPDE) with a varying source term. Solving them is equivalent to querying numerical solutions of the pPDE for multiple times. Reduced basis method (RBM), as one of the popular model reduction techniques, is designed for efficiently completing such a computation task [4, 24, 27, 49]. Typically, it has two separate computation stages: (i) at the offline stage, a low-dimensional subspace is determined that could approximate the pPDE solution manifold subject to a user-defined error tolerance, and a reduced order model is built on this subspace; (ii) then at the online stage, the ROM as a surrogate model to the original pPDE is simulated at a very cheap cost. However, in the context of our considered ParaDIAG preconditioning, it is not straightforward to construct the ROM offline, because it is difficult to parametrize the varying source terms. Moreover, the source terms consist of the residual vectors that would change very irregularly during the GMRES iterations. As a result, we have to move the offline stage of building the ROM to the online stage. However, if the solution stays within a low dimensional manifold, one can still expect the total computational cost is significantly less than the full order simulations, but some special attention are needed to reduce the computation cost of the 'offline' stage during each iteration. Fortunately, the required numbers of preconditioned GMRES iterations are usually very small.

We first recast the systems in Step-(b) into a general form: for $n = 1, \ldots, K$,

$$A_n x_n := ((d_n/\tau) I_h - L_h) x_n = b_n, \tag{13}$$

where $A_n \in \mathbb{C}^{N^2 \times N^2}$ with $d_n = 1 - \alpha^{\frac{1}{K}} e^{\mathrm{i}\theta_n}$ being a complex number. The model is referred to as the full order model (FOM). They have a continuous pPDE counterpart:

$$(d(\theta)/\tau - L)\, v(\mathbf{x};\theta) = b(\mathbf{x};\theta), \quad \text{for } \theta \in [0, 2\pi), \tag{14}$$

where $d(\theta) = 1 - \alpha^{\frac{1}{K}} e^{\mathrm{i}\theta}$, $L(u) = \nabla \cdot (a(\mathbf{x})\nabla u)$ if there is no convection term in (1); and $L(u) = \nabla \cdot (a(\mathbf{x})\nabla u) - \boldsymbol{c}(\mathbf{x}) \cdot \nabla u(\mathbf{x},t) + \frac{h}{2}[p(x,y)u_{xx} + q(x,y)u_{yy}]$ if there is convection. Note that the extra dissipation appears in the latter case due to the use of upwind finite difference [50]. However, we shall focus on the discrete set of equations (13), corresponding to the pPDE at the prescribed discrete parameter set $\{\theta_1, \ldots, \theta_K\}$ specified in (10). For a long time simulation, we expect K to be very large and hence give rise to a large discrete parameter set.

Assume there exists a reduced space spanned by the basis vectors $\{\phi_1, \phi_2, \ldots, \phi_r\}$, the reduced state can be defined by $\widetilde{x}_n = \Phi_r \widehat{x}_n$ with $\Phi_r := [\phi_1 \ \phi_2 \ \ldots \ \phi_r] \in \mathbb{C}^{N^2 \times r}$. Replacing the state in (13) with the reduced approximation $\widetilde{x}_n \approx x_n$ and applying the Galerkin projection, we obtain the following reduced system: to find $\widehat{x}_n$ such that

$$\Phi_r^* A_n \Phi_r \widehat{x}_n = \Phi_r^* b_n, \tag{15}$$

where Φ_r^* is the conjugate transpose of Φ_r. Once $\widehat{x}_n$ is found, the original state variable can be approximated by the reduced state. The size of this reduced system matrix $\Phi_r^* A_n \Phi_r$ is $r \times r$. As r is much less than N^2, it can be solved cheaply using a direct solver with the complexity of $\mathscr{O}(r^3)$.

The remaining task is to efficiently determine the reduced basis matrix Φ_r. For this purpose, RBM uses a greedy strategy based on *a posteriori* error estimations. Such error estimations are usually derived from the equivalence between the error norm and a dual norm of the residual vector. Next, we focus on the discrete systems in Step-(b) and derive an error indicator by establishing the relation between the unknown error norms and measurable residual norms.

A residual-based error indicator

Note that $\alpha \in (0,1)$, the constant $d_n = 1 - \alpha^{\frac{1}{K}} e^{\mathrm{i}\theta_n}$ given in (10) has a positive real part

$$\mathsf{Re}(d_n) = (1 - \alpha^{\frac{1}{K}} \cos\theta_n) \geq (1 - \alpha^{\frac{1}{K}}) > 0, \tag{16}$$

which implies the matrix $A_n = (d_n/\tau) I_h - L_h$ is strictly diagonally dominant by rows and columns since $(-L_h)$ is irreducibly diagonally dominant [26, Thms. 5.17]. Define two positive constants

$$\beta_R := \min_{1\leq k\leq N^2} \left(|(d_n/\tau) - (L_h)_{k,k}| - \sum_{j\neq k} |(L_h)_{k,j}| \right) \tag{17}$$

and

$$\beta_C := \min_{1\leq k\leq N^2} \left(|(d_n/\tau) - (L_h)_{k,k}| - \sum_{j\neq k} |(L_h)_{j,k}| \right). \tag{18}$$

Since $\mathsf{Re}(d_n) > 0$ and $-(L_h)_{k,k} > 0$, there obviously holds

$$|(d_n/\tau) - (L_h)_{k,k}| \geq \mathsf{Re}(d_n/\tau) + |(L_h)_{k,k}|,$$

and $|(L_h)_{k,k}| \geq \sum_{j\neq k} |(L_h)_{k,j}|$ due to the row weakly diagonal dominance of $(-L_h)$, which leads to

$$\beta_R \geq \min_k \left(\mathsf{Re}(d_n/\tau) + |(L_h)_{k,k}| - \sum_{j\neq k} |(L_h)_{k,j}| \right) \geq \mathsf{Re}(d_n)/\tau. \tag{19}$$

Similarly there holds $\beta_C \geq \mathsf{Re}(d_n)/\tau$ because of the column weakly diagonal dominance of $(-L_h)$. By a lower bound of the smallest singular value [60, Cor. 2], there holds

$$\|A_n^{-1}\|_2 \leq \frac{1}{\sqrt{\beta_R \beta_C}} \leq \frac{\tau}{\mathsf{Re}(d_n)} \leq \frac{\tau}{1-\alpha^{1/K}} = \frac{\tau}{1-\alpha^{\tau/T}} \leq \frac{T}{1-\alpha}, \tag{20}$$

where we used the fact that $\phi(\tau) := \frac{\tau}{1-\alpha^{\tau/T}}$ is an increasing function of $\tau \in (0,T]$. In fact, by the inequality $\ln(1+z) < z$ for $z > 0$, a simple calculation can verify $\phi'(\tau) > 0$, that is

$$\phi'(\tau) = \frac{1 - \alpha^{\tau/T}(1 + \ln(\alpha^{-\tau/T}))}{(1-\alpha^{\tau/T})^2} > \frac{1 - \alpha^{\tau/T}(1 + (\alpha^{-\tau/T} - 1))}{(1-\alpha^{\tau/T})^2} = 0.$$

The bound (20) shows that some A_n may become more ill-conditioned (or as ill-conditioned as L_h) as the chosen fixed parameter α gets closer to 1. The second inequality also explains why the linear systems with $\mathsf{Re}(d_n) \approx 0$ (or $\cos(\theta_n) \approx 1$) are more difficult to approximate since their condition numbers are larger. We mention that the limiting case $\alpha = 1$ may lead to much slower convergence rate of preconditioned GMRES (see Example 2 in [35]) and hence not further considered here (see below Remark 1 for discussion of the pure diffusion situation with an upper bound covers the case $\alpha = 1$).

Denote the reduced approximation error by $e_n = x_n - \widetilde{x}_n$, and define the residual vector by $r_n = b_n - A_n\widetilde{x}_n$. The error representation reads:

$$A_n e_n = A_n(x_n - \widetilde{x}_n) = A_n x_n - A_n\widetilde{x}_n = b_n - A_n\widetilde{x}_n = r_n. \tag{21}$$

which implies

$$\|e_n\|_2 = \|A_n^{-1} r_n\|_2 \le \|A_n^{-1}\|_2 \|r_n\|_2 \le \frac{\tau}{\mathsf{Re}(d_n)} \|r_n\|_2 \le \frac{T}{1-\alpha} \|r_n\|_2. \tag{22}$$

The right-hand-side term provides an upper bound of the error norm and the denominator $(1-\alpha)$ suggests that choosing a smaller α close to 0 leads to tighter estimate. Numerically, we find that $\alpha = 10^{-2}$ works very well for all tested examples. Although it is possible to directly take the above upper bounds (e.g. $\frac{\tau}{\mathsf{Re}(d_n)}\|r_n\|_2$) as an absolute error indicator, to be compatible with the used outer FGMRES stopping criterion, we will use the following relative residual norm

$$\mathscr{E}_n := \frac{\|r_n\|_2}{\|b_n\|_2} \tag{23}$$

as an error indicator for estimating the errors and guiding the greedy algorithm for searching reduced basis vectors. During the process, we also obtained the approximate solutions to Step-(b) simultaneously.

Remark 3.1. *The above uniform upper bound (20) may be improvable, since it does not explicitly depend on the ellipticity constant $a_0 > 0$ yet. For example, consider the pure diffusion equation with $\mathbf{c}(\mathbf{x}) = (0,0)$ and constant coefficient $a(x,y) = a_0 > 0$, $(-L_h)$ is symmetric positive definite and there holds [26, Thm 4.34]*

$$\lambda_{\min}(-L_h) = \|(-L_h)^{-1}\|_2^{-1} \ge 16 a_0,$$

where $\lambda_{\min}(-L_h)$ denotes the smallest eigenvalue of $(-L_h)$. Then from $A_n e_n = r_n$ with $A_n = (d_n/\tau) I_h - L_h$, we can get

$$(d_n/\tau) e_n^* e_n + e_n^*(-L_h) e_n = e_n^* A_n e_n = e_n^* r_n,$$

and since $\mathsf{Re}(d_n) > 0$ and $e_n^*(-L_h)e_n \ge \lambda_{\min}(-L_h)\|e_n\|_2^2 \ge 16 a_0 \|e_n\|_2^2 > 0$ there holds

$$(\mathsf{Re}(d_n/\tau) + 16a_0)\|e_n\|_2^2 \le \mathsf{Re}(d_n/\tau) e_n^* e_n + e_n^*(-L_h) e_n \le |e_n^* A_n e_n| = |e_n^* r_n| \le \|e_n\|_2 \|r_n\|_2.$$

This leads to the following slightly improved error estimate

$$\|e_n\|_2 \le \frac{1}{\mathsf{Re}(d_n/\tau) + 16a_0} \|r_n\|_2 \le \frac{\tau}{(1-\alpha^{\tau/T}) + 16 a_0 \tau} \|r_n\|_2 \le \frac{T}{(1-\alpha) + 16 a_0 T} \|r_n\|_2 \le \frac{1}{16 a_0} \|r_n\|_2,$$

which is tighter than the bound (22) when $a_0 \gg 0$ and T is large and is also applicable to the limiting case $\alpha = 1$. For a general convection-diffusion equation, the spatial discretization matrix $(-L_h)$ is not symmetric any more and the above Rayleigh quotient-based arguments utilizing a known lower bound of the smallest eigenvalue would not be valid in general.

A greedy algorithm for basis generation and its complexity

The reduced basis generation strategy is summarized in Algorithm 1. At each iteration, the algorithm identifies a problem that would yield the worst *a posteriori* error, and generates a new basis from the associated FOM solution. As the reduced space gets richer, the reduced approximation would become more accurate. When the accuracy meets the user-defined tolerance $\mathsf{tol}_{\mathsf{ROM}}$, the process would terminate and return the approximation solutions to (13). Note that Step (b) is performed inside the FGMRES iteration, for which we only need to provide a reasonable approximate solution. In our implementation, $\mathsf{tol}_{\mathsf{ROM}}$ is set to be square root of the tolerance of FGMRES iterations. We refer to [21, 57] for further analysis on how the inexact solving of preconditioned systems affects the convergence of FGMRES or relaxed GMRES.

The major computation inside the **for** loop lies in steps 1, 2 and 6. At the m-th iteration, the complexity of these three steps contains $\mathscr{O}(m^3 K + N^2 + N^2 K)$ operations for assembling the matrices and solving the ROMs with direct solver; $\mathscr{O}(mN^2K)$ operations for estimating the residual norm; and $\mathscr{O}(N^2)$

Algorithm 1: Greedy basis generation

Input: $\mathsf{tol}_{\mathsf{ROM}}$, K and $r = 1$

Output: reduced basis matrix Φ_r and solutions to (13) $\{x_1, \ldots, x_K\}$

choose $n_1 \in \Theta = \{1, \ldots, K\}$ such that $n_1 = \arg\max_{n \in \Theta} \|b_n\|$, initialize $\Theta_{in} = \{n_1\}$;

compute x_{n_1} in (13), orthogonalize it as basis w_1, and initialize $\Phi_r = [w_1]$;

for $r = 1, \ldots, r_{\max}$ **do**

1. compute $\widehat{x}_n$ of (15) for all $n \in \Theta \backslash \Theta_{in}$; // $\mathscr{O}(m^3K + N^2 + N^2K)$ operations at m-th iter.
2. evaluate the error estimator $\mathscr{E}_n$ for all $n \in \Theta \backslash \Theta_{in}$; // $\mathscr{O}(mN^2K)$ operations at m-th iter.
3. choose $n_{r+1} = \arg\max_{n \in \Theta \backslash \Theta_{in}} \mathscr{E}_n$;
4. **if** $\mathscr{E}_{n_{r+1}} < \mathsf{tol}_{\mathsf{ROM}}$ **then**
   calculate $x_n = \Phi_r \widehat{x}_n$ for all $n \in \Theta \backslash \Theta_{in}$;
   break;

   end
5. update $\Theta_{in} = \{\Theta_{in}, n_{r+1}\}$;
6. compute $x_{n_{r+1}}$ to (13) at $n = n_{r+1}$, orthogonalize it as basis w_{r+1}, and update $\Phi_r = [\Phi_r, w_{r+1}]$;
   // $\mathscr{O}(N^2)$ operations at m-th iter.

end

operations for solving the FOM with multigrid V-cycles and obtaining a new basis vector. Therefore, *for each FGMRES iteration,* the computation complexity for the Algorithm 1 with ROM dimension $r \ll N$ is $\mathscr{O}(r^4K + 2rN^2 + rN^2K + r^2N^2K) = \mathscr{O}(r^2N^2K)$ operations, which is theoretically comparable to the multigrid-based FOM solver with the $\mathscr{O}(N^2K)$ operations, provided r is bounded by a mesh-independent constant. Hence, the ROM-based solver requires the underlying dynamics to have a low-dimensional solution manifold, to be computationally competitive with the multigrid-based FOM solver.

In order to further improve the practical computational efficiency of the above greedy basis generation algorithm, various heuristic approaches have been proposed in the past decade, see e.g. in [30, 53]. For our greedy basis generation algorithm, we introduce the following algorithmic modifications to reduce its computational complexity.

(i) Update reduced system matrices recursively. The reduced systems can be recursively assembled by making use of the basis structure $\Phi_r = [\Phi_{r-1}, w_r]$. Define $A_n^{(r)} = \Phi_r^* A_n \Phi_r$, $B_r = \Phi_r^* A_n$, $C_r = A_n \Phi_r$, and $F_r = \Phi_r^* S_1$, then we recursively construct them (with $r \geq 2$) according to

$$A_n^{(r)} = \begin{bmatrix} A_n^{(r-1)} & B_r w_r \\ w_r^* C_r & w_r^* A_n w_r \end{bmatrix}, \; B_r = \begin{bmatrix} B_{r-1} \\ w_r^* A_n \end{bmatrix}, \; C_r = \begin{bmatrix} C_{r-1} & A_n w_r \end{bmatrix}, \; F_r = \begin{bmatrix} F_{r-1} & w_r^* S_1 \end{bmatrix},$$

starting with the initialization: $A_n^{(1)} = w_1^\intercal A_n w_1$, $B_1 = w_1^* A_n$, $C_1 = A_n w_1$, $F_1 = w_1^* S_1$.

(ii) Sample $\Theta \backslash \Theta_{in}$ randomly. In steps 1 and 2 at each **for** loop, instead of testing the entire set $\Theta \backslash \Theta_{in}$ for estimating the ROM approximation errors, we randomly choose P components from it. Define $P = r_p K$ with $r_p \in (0, 1]$. Although one usually fixes a randomly sampled training parameter set in RBM, we vary the sample set during the training process in order to achieve generalization. We refer to [11] for further analysis on RBM using random training sets. Based on our numerical experiments, taking only 10 percent (i.e. $r_p = 0.1$) of all the indices in $\Theta \backslash \Theta_{in}$ for training is enough to obtain accurate approximation results.

(iii) Evaluate residual norm on a coarser mesh. In step 2, we compute the residual vectors at N_s^2 points instead of all the N^2 spatial grid points, with $N_s < N$, in order to reduce the complexity of evaluating the residual norm. To define these points, one way is to sample spatial coordinates at random. However, it could become unreliable if the distribution of residuals is far more uneven. In that case, structured random

embeddings could be considered [3, 40]. Here, we propose to select the points from a coarser spatial mesh. Based on our numerical experiments, it is sufficient to use a four times coarser mesh size (i.e. $N_s = N/4$ or $N_s^2 = N^2/16$) for obtaining reliable and accurate results cheaply.

With these modifications (i.e. $P = r_p K$ and $N_s = N/4$), the complexity of Step-(b) *at each FGMRES iteration* now becomes $\mathcal{O}(r^4 P + 2rN^2 + rN^2 P + r^2 N_s^2 P) = \mathcal{O}(r_p r^4 K + 2rN^2 + r_p rN^2 K + (r_p/16) r^2 N^2 K)$ operations, which scales as $\mathcal{O}((r/16+1) r_p rN^2 K)$ if $r \ll N$ and hence improves the original complexity of Algorithm 1 since $r_p \in (0, 1]$. Therefore, it has a great potential to become more efficient than the multigrid-based FOM solver in practice, especially when r is relatively small. Such an improvement is conditional but not surprising because the multigrid algorithm delivers the 'optimal' complexity for solving elliptic systems.

We also note that the number of outer FGMRES iterations might change when different solvers are used in Step-(b). Based on our numerical tests in next section, the ROM-based solvers in Step-(b) always lead to less number of outer FGMRES iterations than the multigrid-based FOM solver while achieving the same approximation accuracy. Finally, we remark that steps 1-2 are parallelizable as individual problems are independent, the involved matrix-vector products can also be implemented in a parallel fashion. One can further divide all the K systems into disjoint subsets and then run the Algorithm 1 within each subset. Some discussions on parallel computing in the RBM setting can be found in [32].

3.1. The motivating example revisited

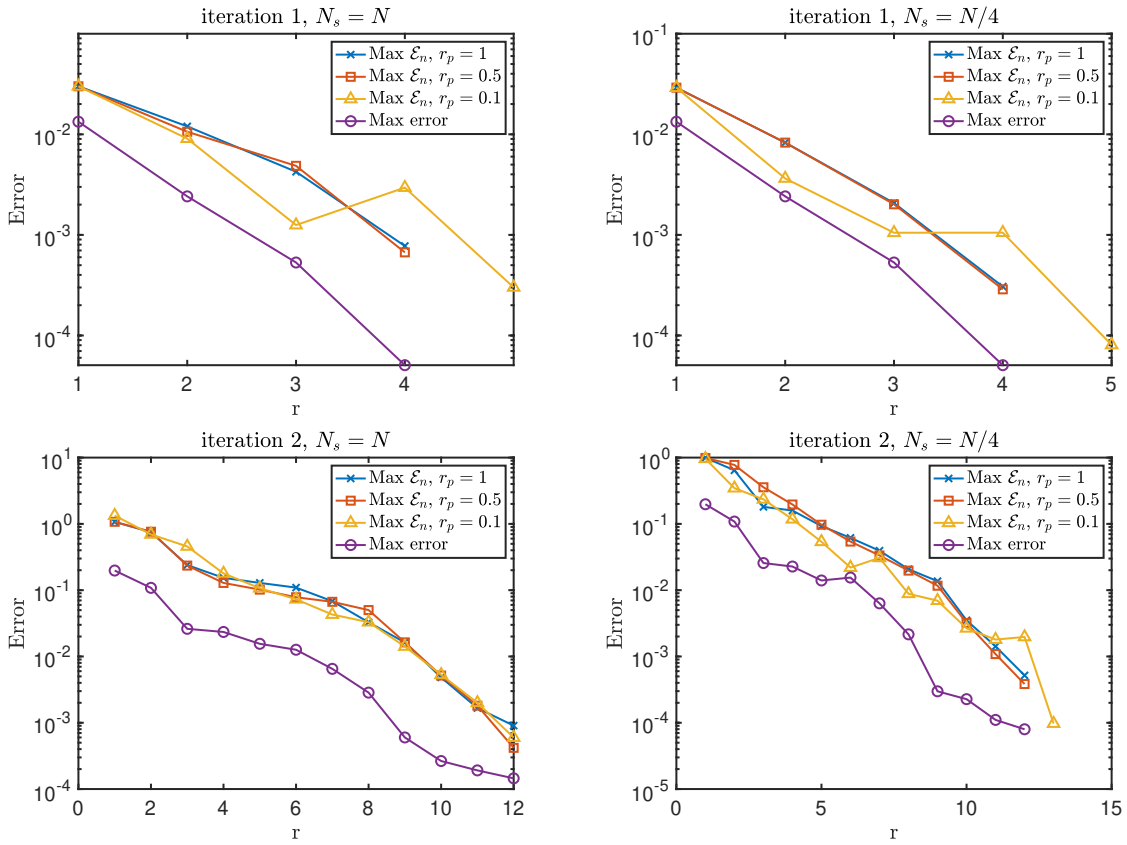

Figure 2: Exact errors and estimated errors of Step-(b) at iteration 1 (first row) and iteration 2 (second row) while varying the value of $r_p = 1, 0.5, 0.1$: (left) $N_s = N$ and (right) $N_s = N/4$.

We illustrate the error indicator $\mathscr{E}_n$ of our proposed ROM-based solver in Step-(b) by investigating the same motivating example again. The aforementioned modifications for Algorithm 1 involve the use of random training sets and evaluations of residual norms on a coarser spatial mesh. First, under the same discretization as in Section 2.1, we keep $N_s = N$ but vary r_p from 1, 0.5 to 0.1. In these cases, the preconditioned FGMRES converges in 2 iterations.

Fig. 2 (left column) shows the evolution of estimated errors with respect to the increasing number of reduced basis vectors during the greedy search algorithm, together with the exact errors. It is seen that: (1) the error indicator gives a good approximation of the actual approximation error; (2) with the same stopping criterion, the dimension of resulting reduced basis stays almost the same when different values of r_p (or equivalently P) are used. Therefore, as r_p gets smaller, the computational efficiency improves but the overall accuracy of the algorithm does not deteriorate. Using the same computational setting but decreasing N_s from N to $N/4$, we observe similar numerical behaviors as shown in Fig. 2 (right column). This indicates the used heuristic modifications for Algorithm 1 are practicable and effective.

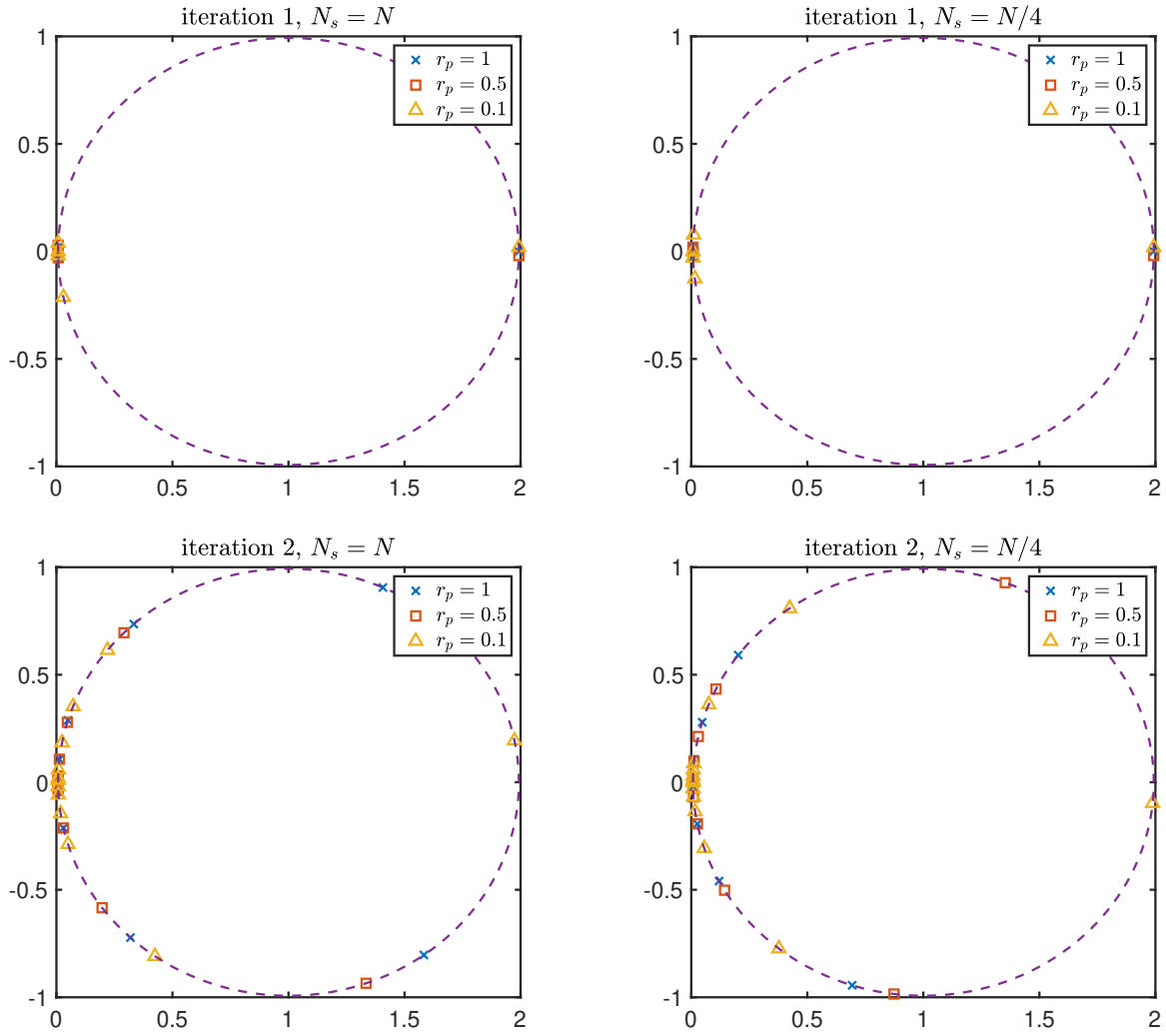

Figure 3: Distribution of d_n for $n \in \Theta_{in}$ in Step-(b) at iteration 1 (first row) and iteration 2 (second row) while varying the value of $r_p = 1, 0.5, 0.1$: (left) $N_s = N$ and (right) $N_s = N/4$.

We plot the complex number d_n's associated to the selected basis indices $n \in \Theta_{in}$ as points in Fig. 3. The points distribute on the circle centered at $1+0\mathrm{i}$ with radius $\alpha^{\frac{1}{K}} \approx 0.99$ but concentrate near $\theta_n = 0$ or 2π. It matches our observations in Fig. 1 where more structures appear near $n = 1$ and $n = K$. This also implies ROMs generated from evenly distributed snapshots (as commonly used in POD-based ROMs) may not be effective enough for approximating such systems.

Compared with the MG-based FOM solver with 4 FGMRES iterations, the proposed ROM-based solver, when $r_p = 0.1$ and $N_s = N/4$, achieves the same accuracy 5.7×10^{-5} with only 2 FGMRES iterations. As expected, the total CPU time decreases about 9 times from 6.2 s to 0.7 s, in which 0.4 s (about 57%) is spent on solving Step-(b). The reduced number of preconditioned FGMRES iterations also contributes to the substantial saving of overall CPU times.

To further investigate the influence of the algorithmic parameters α, N_s, and r_p on the ParaDIAG-ROM preconditioner, we fix the mesh size ($N = 128$ and $K = 1280$) and check the numerical performance while varying the parameter values. The convergence results are summarized in Table 2. Here 'Iter' means the preconditioned FGMRES iteration numbers, 'CPU' represents the total CPU times (in seconds), '$\bar{r}$' denotes the average reduced basis dimension, and 'Vc' stands for the average multigrid V-cycles used for solving the selected full-order models. The approximation errors are not reported as they stay unchanged in all cases. This is because the preconditioned FGMRES solver attains the same stopping tolerance and different ROM reduced basis only affects the preconditioning step. Based on Table 2, we can make the following remarks:

- With a fixed (N_s, r_p), the preconditioned FGMRES iteration numbers and reduced basis dimension $\bar{r}$ are insensitive to the choices of α. A smaller α shows slightly faster convergence rate, but α should not be taken too small since the round-off errors due to diagonalization will quickly dominate the overall approximation errors; we refer to [8] for relevant technical discussion on the adaptive choice of α in the framework of preconditioned Richardson iterations for faster convergence rates.

- With a fixed (N_s, α), the choice of $r_p = 0.05$ or $r_p = 0.1$ often leads to the lowest CPU times, while a too small $r_p = 0.01$ tends to generate inaccurate reduced basis approximations and hence results in more preconditioned FGMRES iteration numbers. Finding the optimal $r_p \in (0, 1]$ seems to be difficult and problem-dependent, but a potential criterion is to steadily decrease the value of r_p until the preconditioned FGMRES iteration number starts to increase.

- With a fixed (r_p, α), there is no significant change in the performance of the preconditioner when N_s decays from N to $N/8$. However, for 3D problems, in which N is usually large, the use of a small N_s could yield more obvious savings in CPU times (considering $N_s^3 = N^3/64$ when $N_s = N/4$).

These numerical observations provide some evidence to support our empirical (though not optimal) parameter choice: $\alpha = 0.01, r_p = 0.1$ and $N_s = N/4$, to be used in all the following numerical examples.

4. Numerical examples

In this section, we present several 1D and 2D PDE examples to illustrate the effectiveness of our proposed ROM-based preconditioning techniques. All simulations are implemented using MATLAB on a Dell Precision 7520 Laptop with Intel(R) Core(TM) i7-7700HQ CPU@2.80GHz and 48GB RAM. The CPU time (in seconds) for solving the whole system and all the sub-systems in Step-(b) are estimated separately by using the timing functions `tic`/`toc` with our serial implementation. We employ the right-preconditioned FGMRES [52] solver (without restarts) provided in the Tensor Train (TT) Toolbox [2] and choose a zero initial guess with a stopping tolerance $\mathrm{tol} = 10^{-6}$ based on the reduction in relative residual norms. To avoid

[2] https://github.com/oseledets/TT-Toolbox

Table 2: Numerical behavior of the ParaDIAG-ROM preconditioner with different α, N_s, and r_p for Example 2a (with a fixed mesh size $N = 128$ and $K = 1280$). The lowest CPU time is 4.4 seconds (in bold) when $\alpha = 0.01, N_s = N/4$, and $r_p = 0.05$.

		$\alpha = 1$			$\alpha = 0.1$			$\alpha = 0.01$			$\alpha = 0.001$		
N_s	r_p	Iter	CPU	$\bar{r}$ (Vc)	Iter	CPU	$\bar{r}$ (Vc)	Iter	CPU	$\bar{r}$ (Vc)	Iter	CPU	$\bar{r}$ (Vc)
N	1.00	3	23.1	12 (5)	2	12.8	9 (5)	2	12.2	9 (5)	2	12.1	9 (5)
	0.50	3	15.9	12 (5)	2	8.6	9 (5)	2	8.4	9 (5)	2	9.1	9 (5)
	0.10	3	8.8	11 (5)	3	8.1	11 (5)	2	5.4	10 (5)	2	5.3	10 (5)
	0.05	3	8.9	11 (5)	2	5.2	9 (5)	2	5.1	10 (5)	2	5.1	10 (5)
	0.01	4	9.1	9 (4)	4	9.8	12 (5)	4	9.6	12 (4)	3	7.2	10 (4)
$N/2$	1.00	3	17.5	11 (5)	2	9.4	9 (5)	2	9.4	9 (5)	2	9.7	9 (5)
	0.50	3	12.4	11 (5)	2	6.8	9 (5)	2	7.1	9 (5)	2	7.1	9 (5)
	0.10	3	8.2	10 (5)	3	7.6	11 (5)	2	5.2	10 (5)	2	6.4	10 (5)
	0.05	3	7.4	11 (5)	2	5.1	9 (5)	2	5.0	10 (5)	2	4.9	9 (5)
	0.01	5	11.0	8 (4)	4	9.5	11 (4)	4	9.5	11 (4)	4	9.9	12 (4)
$N/4$	1.00	3	12.5	10 (5)	2	6.8	8 (5)	2	7.4	9 (5)	2	7.7	9 (5)
	0.50	3	10.1	10 (5)	2	5.8	8 (5)	2	6.0	9 (5)	2	6.2	9 (5)
	0.10	3	7.4	10 (5)	3	7.4	11 (5)	2	4.9	10 (5)	2	4.9	10 (5)
	0.05	3	7.1	11 (5)	2	4.6	9 (5)	2	**4.4**	9 (5)	2	4.5	9 (5)
	0.01	4	8.4	8 (4)	4	9.2	10 (4)	4	9.0	10 (4)	3	6.7	10 (4)
$N/8$	1.00	3	12.6	10 (5)	3	14.1	11 (5)	2	7.8	9 (5)	2	8.2	9 (5)
	0.50	3	10.3	10 (5)	2	6.2	8 (5)	2	6.6	9 (5)	2	6.6	9 (5)
	0.10	3	7.5	8 (5)	3	7.3	10 (5)	3	7.6	10 (5)	3	7.6	11 (5)
	0.05	3	7.3	10 (5)	3	7.6	11 (5)	2	4.9	9 (5)	2	4.9	9 (5)
	0.01	4	9.0	8 (4)	4	9.8	11 (4)	3	7.0	9 (4)	3	7.2	10 (4)

introducing possible round-off errors due to α being too small (as theoretically preferred), we will choose the preconditioner P_α with a fixed small $\alpha = 10^{-2}$, which leads to only a small number of preconditioned FGMRES iterations in all tested examples. With other moderately small α it leads to very similar results.

For the ROM-based solver, we choose the relative residual tolerance to be $\mathsf{tol}_{\mathsf{ROM}} = \sqrt{\mathsf{tol}}$, which works well for all the tested examples. In particular, we numerically observed that using a costly sparse direct solver in Step-(b) yields the same outer FGMRES iteration numbers as our ROM-based preconditioner (ParaDIAG-ROM). We highlight that a smaller ROM residual tolerance would lead to higher reduced basis dimension and computational costs, which however will not improve the overall FGMRES accuracy. Within the Algorithm 1, we will solve the selected full-order systems (13) by the backslash sparse direct solver for 1D examples (with tridiagonal systems), and the geometric multigrid V-cycles with ILU smoother and the same stopping tolerance $\mathsf{tol} = 10^{-6}$ for 2D examples. We set the random number generator using the function `rng(0,'v5uniform')`, with a seed of 0 and the uniform generator `'v5uniform'`.

For the compared multigrid-based ParaDIAG (ParaDIAG-MG) preconditioner, only one V-cycle (based on the IFISS package [56]) is used to approximately solve inner linear systems in Step-(b), which gives somewhat rough approximation and therefore requires more outer FGMRES iterations than the ROM-based preconditioner with our chosen tolerance. In ParaDIAG-MG preconditioner, for a general comparison purpose we will use the Gauss-Seidel (GS) smoother and incomplete LU(ILU) smoother for 1D and 2D examples, respectively. For 2D examples, the ILU smoother usually provides more robust and faster convergence rate than the GS smoother. We have tried to optimize/vectorize the majority of our MATLAB codes whenever possible, but we acknowledge the reported CPU times are mainly for illustration purpose.

In all the numerical experiments, we compare the errors and CPU time between the ParaDIAG-MG and ParaDIAG-ROM. The approximation errors are evaluated in discrete L^2 norm in both space and time. The

order of accuracy is estimated by the logarithmic ratio of the approximation errors between two successive refined (halved) meshes, which should be close to 1 for the numerical scheme if solutions are sufficiently smooth. The CPU time measures the efficiency of the two approaches in serial computing. Since the only difference between them is the solver used in Step-(b), we also list the separate CPU time for this particular step (displayed inside parentheses of the column 'CPU' in Tables 3-8).

4.1. 1D Examples.

Example 1a. Heat equation with constant diffusion coefficients. We first consider a linear heat equation with constant coefficients defined on the domain $\Omega = (0, \pi)$:

$$\begin{cases} u_t = \varepsilon u_{xx} + f, & \text{in } \Omega \times (0,T), \\ u = 0, & \text{on } \partial\Omega \times (0,T). \end{cases} \tag{24}$$

We choose $f = 0$, and a non-smooth triangle shaped initial condition

$$u(x,0) = \begin{cases} 2x, & 0 \le x \le \pi/2, \\ 2(\pi - x), & \pi/2 < x \le \pi, \end{cases}$$

with the exact solution is given by

$$u(x,t) = \frac{8}{\pi} \sum_{n=0}^{\infty} \frac{\cos((2n+1)(2x-\pi)/2)}{(2n+1)^2} e^{-\varepsilon(2n+1)^2 t}.$$

The errors and convergence results are reported in Table 3, where the total CPU time of our proposed ROM-based ParaDIAG (ParaDIAG-ROM) preconditioner is over 10 times faster than the multigrid-based ParaDIAG (ParaDIAG-MG) preconditioner with the point-wise Gauss-Seidel smoother. The speed up ratio in CPU time for computing Step-(b) is even higher than 20 times. Due to the non-smoothness of the initial condition, the order of accuracy is expected to be slightly lower than one as the mesh refines. Since one multigrid V-cycle very approximately solve the linear systems in Step-(b), which leads to more outer FGMRES iterations than our ParaDIAG-ROM preconditioner. The average dimension of the ROM reduced basis, denoted by column '$\bar{r}$', shows very mild growth as the mesh is refined, which is reasonable considering the full order models' dimension increases by four times. Fig. 4 compares the exact solution and the numerical solution computed by our ParaDIAG-ROM preconditioner, where the non-smooth initial condition is smoothed out quickly due to the diffusion.

Table 3: Results of preconditioned FGMRES for Example 1a: 1D heat equation ($\varepsilon = 0.1$, $T = 10$)

	ParaDIAG-MG(GS) Preconditioner				ParaDIAG-ROM Preconditioner				
(N,K)	Error	Order	Iter	CPU	Error	Order	Iter	CPU	$\bar{r}$
(256,2560)	8.5E-04	1.2	6	4.1 (4.0)	8.5E-04	1.2	2	0.2 (0.1)	10
(512,5120)	4.1E-04	1.1	6	12.5 (11.6)	4.1E-04	1.1	2	0.9 (0.5)	12
(1024,10240)	2.2E-04	0.9	6	42.3 (38.1)	2.2E-04	0.9	2	3.3 (1.8)	13

Example 1b. Convection-diffusion (C-D) equation. We also consider another convection-diffusion equation defined on the domain $\Omega = (0,1)$:

$$\begin{cases} u_t = \varepsilon u_{xx} - c u_x, & \text{in } \Omega \times (0,T), \\ u(0,t) = 1, \quad u(1,t) = 0, & t \in (0,T), \\ u(x,0) = 0, & x \in \Omega, \end{cases} \tag{25}$$

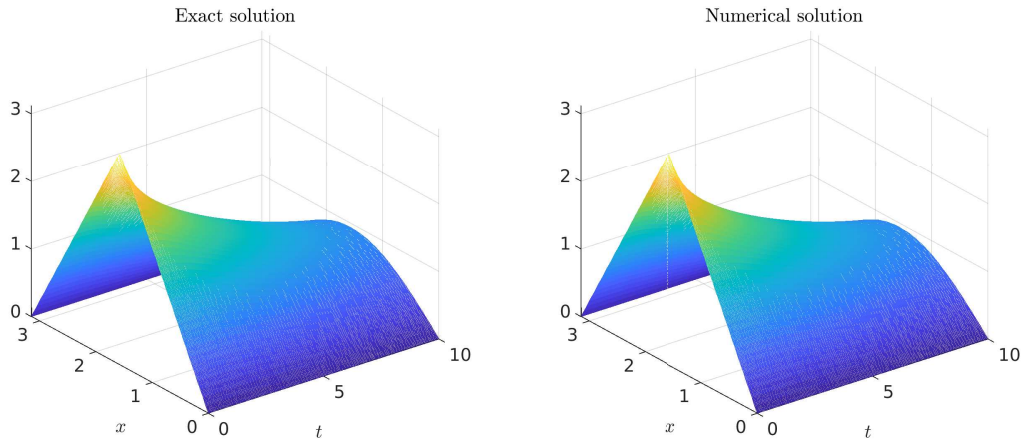

Figure 4: The exact solution and computed numerical solution by ParaDIAG-ROM preconditioner in Example 1a.

where $P_e = c/\varepsilon$ is the Péclet number that represents the ratio of the convection rate over the diffusion rate. The exact solution is explicitly known [42] in terms of infinity series, which is used to measuring the errors. The errors and convergence results are reported in Table 4, where our ParaDIAG-ROM preconditioner is over 10 times faster than the ParaDIAG-MG preconditioner. Fig. 5 compares the exact solution and the numerical solution computed by our ParaDIAG-ROM preconditioner. Due to a discontinuity at the corner $(0,0)$, our used finite difference scheme show a slightly degraded order of accuracy. Again, the boundary layer leads to slightly larger $\bar{r}$ for accurate approximation than the previous example.

Table 4: Results of preconditioned FGMRES for Example 1b: 1D C-D equation with upwind scheme and ($\varepsilon = 0.1, c = 0.2, T = 10$)

	ParaDIAG-MG(GS) Preconditioner				ParaDIAG-ROM Preconditioner				
(N,K)	Error	Order	Iter	CPU	Error	Order	Iter	CPU	$\bar{r}$
(256,2560)	6.1E-03	0.7	5	3.5 (3.4)	6.1E-03	0.7	2	0.2 (0.1)	13
(512,5120)	3.7E-03	0.7	5	10.1 (9.4)	3.7E-03	0.7	2	0.7 (0.4)	14
(1024,10240)	2.3E-03	0.7	5	35.5 (32.1)	2.2E-03	0.7	2	2.9 (1.5)	16

4.2. 2D Examples.

Example 2a. Heat equation with constant diffusion coefficients. We consider the motivating example used in Section 2.1 again. The errors and convergence results are reported in Table 5, where our ParaDIAG-ROM preconditioner is about 8 times faster than the ParaDIAG-MG preconditioner. In the last column '$\bar{r}$ (Vc)', the first number $\bar{r}$ indicates the average reduced basis dimension and Vc stands for the average multigrid V-cycles used for solving all the chosen full-order models. The ParaDIAG-MG preconditioner shows very fast and robust convergence rate due to the used ILU smoother, but the overall computational cost is still high since it requires total K V-cycles to approximately solve all the K linear systems at each iteration. In contrast, the ParaDIAG-ROM preconditioner only used in average $\bar{r} \times (\text{Vc}) \ll K$ full-order model V-cycles to find the reduced basis and then solved K reduced linear systems of much smaller sizes cheaply. Fig. 6 compares the exact solution and the numerical solution computed by our ParaDIAG-ROM preconditioner at the final time.

Example 2b. Heat equation with spatially variable diffusion coefficients. We also consider linear

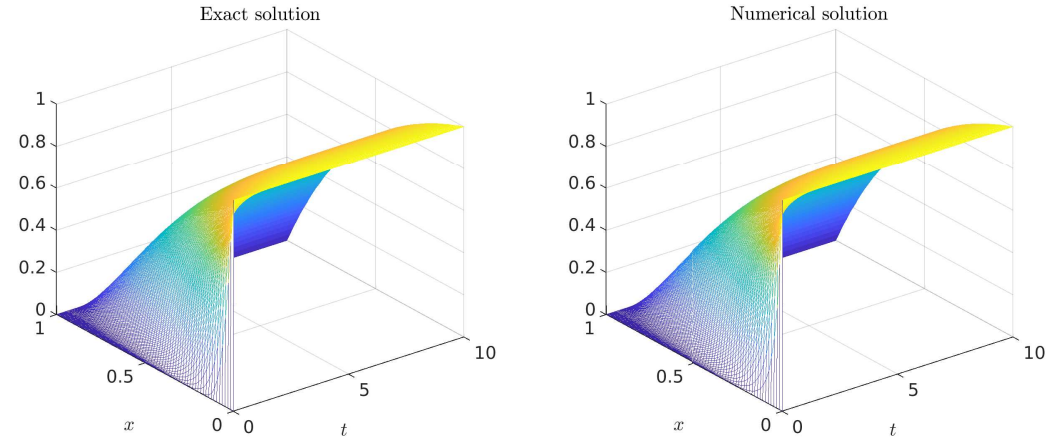

Figure 5: The exact solution and computed numerical solution by ParaDIAG-ROM preconditioner in Example 1b.

Table 5: Results of preconditioned FGMRES for Example 2a: 2D heat equation ($\varepsilon = 0.01$, $T = 10$)

	ParaDIAG-MG(ILU) Preconditioner				ParaDIAG-ROM Preconditioner				
(N^2, K)	Error	Order	Iter	CPU	Error	Order	Iter	CPU	$\bar{r}$ (Vc)
(64^2,640)	5.7E-05	1.2	4	6.2(5.5)	5.7E-05	1.2	2	0.7 (0.4)	8 (4)
(128^2,1280)	2.7E-05	1.1	4	48.3(40.7)	2.7E-05	1.1	2	6.1 (2.6)	9 (5)
(256^2,2560)	1.3E-05	1.0	4	378.5(301.5)	1.3E-05	1.0	2	43.1 (20.3)	10 (5)

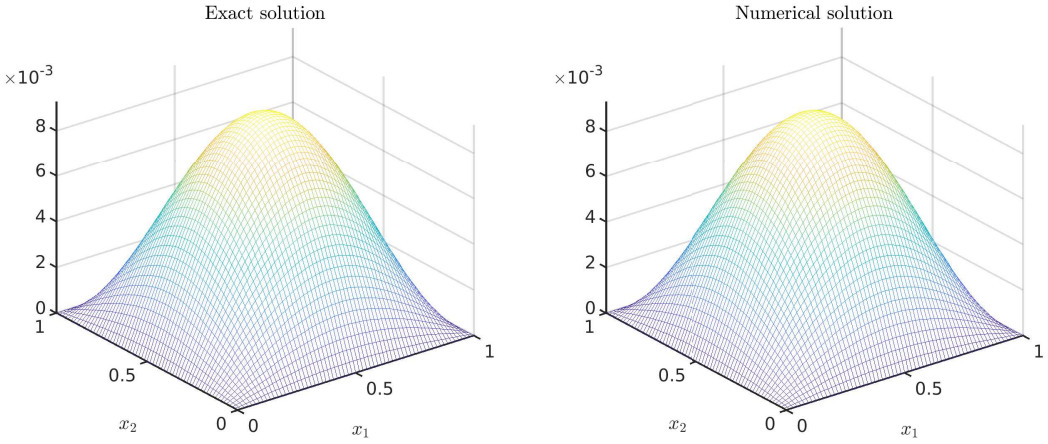

Figure 6: The exact solution and computed numerical solution by ParaDIAG-ROM preconditioner at the final time $T = 10$ in Example 2a.

heat equation with variable diffusion coefficients (in divergence form) on $\Omega = (0,1)^2$:

$$\begin{cases} u_t = \nabla \cdot (a \nabla u) + f, & \text{in } \Omega \times (0,T), \\ u = g, & \text{on } \partial\Omega \times (0,T). \end{cases} \tag{26}$$

We choose $a(x,y) = 10^{-5}\sin(\pi xy)$, the Dirichlet boundary condition g, the initial condition u_0 and a suit-

able f such that the exact solution is given as $u(x,y,t) = e^{-t/10}e^{\cos(2\pi x)+\sin(3\pi y)}$. The errors and convergence results are reported in Table 6, where our ParaDIAG-ROM preconditioner is about 10 times faster than the ParaDIAG-MG preconditioner. In particular, it takes only two reduced basis vectors to accurately approximate the full-order model solutions in Step-(b). The variable diffusion coefficients does not affect the convergence rate of both preconditioners. Our ParaDIAG-ROM preconditioner show very high computational efficiency, although the ParaDIAG-MG preconditioner based on ILU smoother indeed converges faster than the 1D case based on the GS smoother. Fig. 7 compares the exact solution and the numerical solution computed by our ParaDIAG-ROM preconditioner at the final time.

Table 6: Results of preconditioned FGMRES for Example 2b: 2D heat equation with variable coefficient ($T = 10$)

	ParaDIAG-MG(ILU) Preconditioner				ParaDIAG-ROM Preconditioner				
(N^2,K)	Error	Order	Iter	CPU	Error	Order	Iter	CPU	$\bar{r}$(Vc)
(64^2,640)	2.5E-03	1.0	3	4.4(3.8)	2.5E-03	1.0	2	0.5 (0.1)	2 (2)
(128^2,1280)	1.3E-03	1.0	3	35.9(29.7)	1.3E-03	1.0	2	3.7 (0.8)	2 (2)
(256^2,2560)	6.5E-04	1.0	3	289.7(223.2)	6.3E-04	1.0	2	27.0 (6.4)	2 (2)

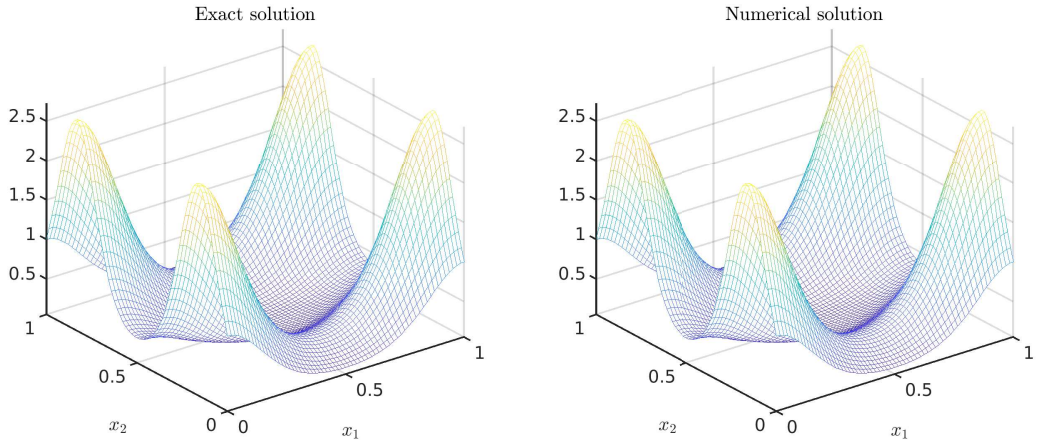

Figure 7: The exact solution and computed numerical solution by ParaDIAG-ROM preconditioner at the final time $T = 10$ in Example 2b.

Example 2c. Convection-diffusion (C-D) equation with internal layer. We consider another convection-diffusion equation (adapted from [31]) defined on the domain $\Omega = (0,1)^2$:

$$\begin{cases} u_t = \varepsilon\Delta u - \boldsymbol{c}\cdot\nabla u + f, & \text{in } \Omega\times(0,T), \\ u(x,y,t) = 0, & \text{on } \partial\Omega\times(0,T) \\ u(x,y,0) = u_0(x,y), & \text{in } \Omega, \end{cases} \tag{27}$$

where $\boldsymbol{c} = (2,3)$, $u_0 = u(x,y,0)$, and f is chosen such that the exact solution reads

$$u = 16e^{-t}x(1-x)y(1-y)\left(1/2+\arctan(2\sqrt{1/\varepsilon}(0.25^2-(x-0.5)^2-(y-0.5)^2))/\pi\right).$$

In this problem, there is a hump changing its height over time and the steepness (or thickness) of the circular internal layer depends on the diffusion coefficient ε (or $\sqrt{\varepsilon}$). The errors and convergence results are reported in Table 7, where our ParaDIAG-ROM preconditioner is about 3 times faster than the ParaDIAG-MG

preconditioner. Fig. 8 compares the exact solution and the numerical solution computed by our ParaDIAG-ROM preconditioner at the final time. As also observed in [31], our used finite difference upwind scheme indeed leads to spurious oscillations behind the hump in the direction of the convection, which may be further suppressed by other specially designed schemes. In this example, the ParaDIAG-MG preconditioner based on ILU smoother converges very fast in only 2-3 iterations, while our ParaDIAG-ROM preconditioner shows slightly increasing $\bar{r}$(V-cycles), which indicates the advantage of our ParaDIAG-ROM preconditioner may become slightly less significant. This drawback of mildly increasing average reduced basis dimension $\bar{r}$ in our ParaDIAG-ROM preconditioner for convection-dominated problems deserves further investigation.

Table 7: Results of preconditioned FGMRES for Example 2c: 2D C-D equation with upwind scheme ($\varepsilon = 10^{-4}$, $T = 10$)

	ParaDIAG-MG(ILU) Preconditioner				ParaDIAG-ROM Preconditioner				
(N^2, K)	Error	Order	Iter	CPU	Error	Order	Iter	CPU	$\bar{r}$ (Vc)
(64^2,640)	1.0E-01	0.8	2	3.0(2.6)	1.0E-01	0.8	2	0.8 (0.4)	13 (2)
(128^2,1280)	6.1E-02	0.7	2	24.8(19.9)	6.1E-02	0.7	2	5.8 (2.8)	17 (2)
(256^2,2560)	3.6E-02	0.8	3	285.9(221.9)	3.6E-02	0.8	2	46.9 (24.8)	21 (3)

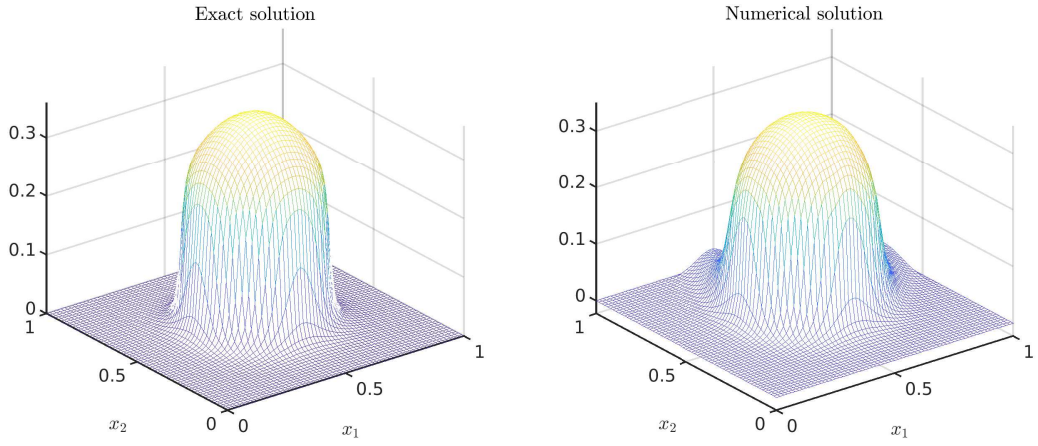

Figure 8: The exact solution and computed numerical solution by ParaDIAG-ROM preconditioner at the final time $T = 10$ in Example 2c.

Example 2d. Convection-diffusion (C-D) equation with boundary layer. In the last example, we consider the prototype convection-diffusion equation [13, 35] defined on the domain $\Omega = (0,1)^2$:

$$\begin{cases} u_t = \varepsilon \Delta u - \boldsymbol{c} \cdot \nabla u, & \text{in } \Omega \times (0,T), \\ u(x,y,t) = (1 - e^{-10t}) \mathbb{1}_{\{x=1\}}, & \text{on } \partial\Omega \times (0,T) \\ u(x,y,0) = 0 & \text{in } \Omega. \end{cases} \tag{28}$$

where $\boldsymbol{c} = (2y(1-x^2), -2x(1-y^2))$ is the circulating wind and the indicator function $\mathbb{1}_{\{x=1\}}$ denotes one-side hot wall boundary condition at $x = 1$. Since the exact solution is unknown, we will only report the relative residual norms of the computed solutions. The relative residual norms and convergence results are reported in Table 8, where our ParaDIAG-ROM preconditioner is about 8 times faster than the ParaDIAG-MG preconditioner. Fig. 9 compares the exact solution and the numerical solution computed by

our ParaDIAG-ROM preconditioner at the final time, where the boundary layer was accurately approximated without any obvious spurious oscillations. Due to the boundary layer, our ParaDIAG-ROM preconditioner requires even larger dimension $\bar{r}$ (and slightly more V-cycles) for accurate approximation, which still deliver much better computational efficiency than the ParaDIAG-MG preconditioner. We highlight that the mesh size $(N^2 = 256^2, K = 2560)$ gives over 167 million unknowns, which costs only about 2 mins.

Table 8: Results of preconditioned FGMRES for Example 2d: 2D C-D equation with upwind difference ($\varepsilon = 1/200, T = 10$)

	ParaDIAG-MG(ILU) Preconditioner			ParaDIAG-ROM Preconditioner			
(N^2, K)	Rel. Res.	Iter	CPU	Rel. Res.	Iter	CPU	$\bar{r}$ (Vc)
(64^2,640)	2.1E-07	9	13.5(11.9)	1.2E-08	4	4.1 (3.4)	34 (7)
(128^2,1280)	2.8E-07	10	116.9(99.0)	1.1E-08	4	26.4 (20.3)	41 (9)
(256^2,2560)	6.0E-07	10	900.6(742.4)	3.7E-07	3	124.4 (91.7)	42 (10)

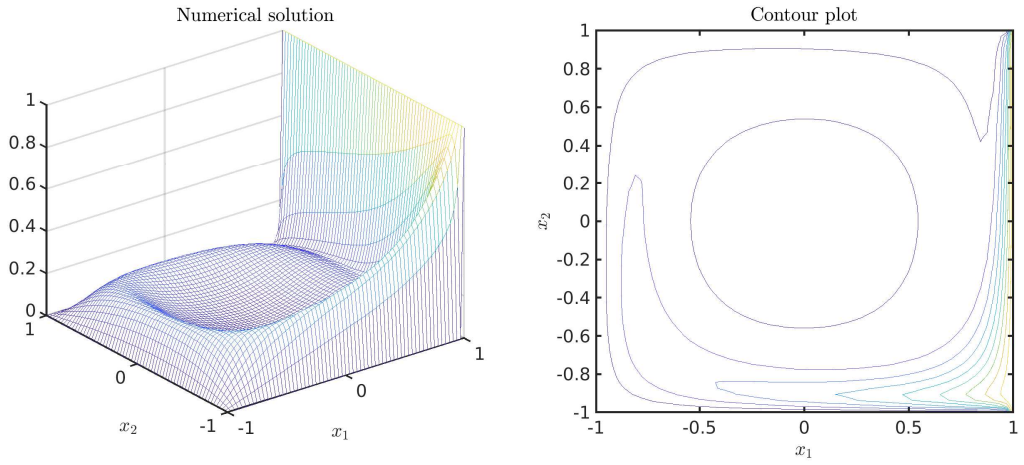

Figure 9: The computed numerical solution by ParaDIAG-ROM preconditioner and its contour plot with $N = 64$ at the final time $T = 10$ in Example 2d.

5. Conclusion

In this work, we developed a ROM-accelerated parallel-in-time preconditioner for solving all-at-once systems from evolutionary PDEs. The ROM is used to reduce the computational cost for solving the sequence of complex-shifted linear systems arising from Step-(b) of the ParaDIAG algorithm. The reduced basis method is applied to build the ROM online, in which a greedy basis selection algorithm with several algorithmic improvements is used for finding the reduced basis efficiently. A variety of numerical examples are tested that illustrate the efficacy of the proposed ROM-accelerated ParaDIAG preconditioner. Compared with the state-of-the-art multigrid-based ParaDIAG preconditioner, the proposed approach gains more than an order of magnitude speed-up in CPU times, although the former formally has the "optimal" complexity. The dimension of the reduced basis seems to be moderately problem dependent, where the convection-dominated cases require higher ROM dimensions than the diffusion-dominated ones. The practical performance and scalability of our proposed ParaDIAG-ROM preconditioner in parallel computing setting deserve further investigation, since our given greedy basis generation algorithm is sequential.

We plan to investigate more efficient RBM methods such as the reduced collocation methods [10] at the next step. We have mainly focused on the backward Euler scheme in time, which has the first-order

accuracy, but the generalization of our proposed approach to the second-order or higher-order time schemes (i.e., BDF2 scheme [61]) is straightforward. The application of our current approach to the more difficult hyperbolic wave equation [12, 17, 37] and its optimal control problem [62] is another ongoing work, where the indefiniteness of complex-shifted linear systems in Step-(b) leads to dramatically enlarged reduced basis dimensions and hence results in less efficiency. This is not surprising since the multigrid solvers would also have convergence issues when solving such indefinite systems. It is also valuable to generalize our approach to treat similar complex-shifted linear systems with varying right-hand-sides arising in the Laplace transform-based parallelizable contour integral method (see e.g., [25, 29, 46, 54, 55]).

Acknowledgments

Z.W. gratefully acknowledges support from U.S. National Science Foundation through grant DMS-1913073.

References

[1] Antoulas, A.C.: Approximation of large-scale dynamical systems. SIAM (2005)

[2] Antoulas, A.C., Beattie, C.A., Güğercin, S.: Interpolatory methods for model reduction. SIAM (2020)

[3] Balabanov, O., Nouy, A.: Randomized linear algebra for model reduction. Part i: Galerkin methods and error estimation. Advances in Computational Mathematics **45**(5-6), 2969–3019 (2019)

[4] Barrault, M., Maday, Y., Nguyen, N.C., Patera, A.T.: An 'empirical interpolation' method: application to efficient reduced-basis discretization of partial differential equations. Comptes Rendus Mathematique **339**(9), 667–672 (2004)

[5] Binev, P., Cohen, A., Dahmen, W., DeVore, R., Petrova, G., Wojtaszczyk, P.: Convergence rates for greedy algorithms in reduced basis methods. SIAM journal on mathematical analysis **43**(3), 1457–1472 (2011)

[6] Bini, D.A., Latouche, G., Meini, B.: Numerical Methods for Structured Markov Chains. Oxford University Press (2005)

[7] Brunton, S.L., Kutz, J.N.: Data-driven science and engineering: Machine learning, dynamical systems, and control. Cambridge University Press (2019)

[8] Caklovic, G., Speck, R., Frank, M.: A parallel implementation of a diagonalization-based parallel-in-time integrator. arXiv preprint arXiv:2103.12571 (2021)

[9] Chen, F., Hesthaven, J.S., Zhu, X.: On the use of reduced basis methods to accelerate and stabilize the parareal method. In: Reduced Order Methods for modeling and computational reduction, pp. 187–214. Springer (2014)

[10] Chen, Y., Gottlieb, S.: Reduced collocation methods: Reduced basis methods in the collocation framework. Journal of Scientific Computing **55**(3), 718–737 (2013)

[11] Cohen, A., Dahmen, W., DeVore, R., Nichols, J.: Reduced basis greedy selection using random training sets. ESAIM: Mathematical Modelling and Numerical Analysis **54**(5), 1509–1524 (2020)

[12] De Sterck, H., Friedhoff, S., Howse, A.J., MacLachlan, S.P.: Convergence analysis for parallel-in-time solution of hyperbolic systems. Numerical Linear Algebra with Applications **27**(1), e2271 (2020)

[13] Elman, H.C., Silvester, D.J., Wathen, A.J.: Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics. Oxford University Press, USA (2014)

[14] Falgout, R.D., Friedhoff, S., Kolev, T.V., MacLachlan, S.P., Schroder, J.B.: Parallel time integration with multigrid. SIAM Journal on Scientific Computing **36**(6), C635–C661 (2014)

[15] Farhat, C., Cortial, J., Dastillung, C., Bavestrello, H.: Time-parallel implicit integrators for the near-real-time prediction of linear structural dynamic responses. International journal for numerical methods in engineering **67**(5), 697–724 (2006)

[16] Gander, M.J.: 50 years of time parallel time integration. In: Multiple shooting and time domain decomposition methods, pp. 69–113. Springer (2015)

[17] Gander, M.J., Halpern, L., Rannou, J., Ryan, J.: A direct time parallel solver by diagonalization for the wave equation. SIAM Journal on Scientific Computing **41**(1), A220–A245 (2019)

[18] Gander, M.J., Liu, J., Wu, S.L., Yue, X., Zhou, T.: Paradiag: Parallel-in-time algorithms based on the diagonalization technique. arXiv preprint arXiv:2005.09158 (2020)

[19] Gander, M.J., Neumuller, M.: Analysis of a new space-time parallel multigrid algorithm for parabolic problems. SIAM Journal on Scientific Computing **38**(4), A2173–A2208 (2016)

[20] Gander, M.J., Zhang, H.: A class of iterative solvers for the Helmholtz equation: Factorizations, sweeping preconditioners, source transfer, single layer potentials, polarized traces, and optimized schwarz methods. Siam Review **61**(1), 3–76 (2019)

[21] Giraud, L., Gratton, S., Langou, J.: Convergence in backward error of relaxed GMRES. SIAM Journal on Scientific Computing **29**(2), 710–728 (2007)

[22] Goddard, A., Wathen, A.: A note on parallel preconditioning for all-at-once evolutionary pdes. Electronic Transactions on Numerical Analysis **51**, 135–150 (2019)

[23] Golub, G., Van Loan, C.: Matrix Computations. Matrix Computations. Johns Hopkins University Press (2012)

[24] Grepl, M.A., Maday, Y., Nguyen, N.C., Patera, A.T.: Efficient reduced-basis treatment of nonaffine and nonlinear partial differential equations. ESAIM: Mathematical Modelling and Numerical Analysis **41**(3), 575–605 (2007)

[25] Guglielmi, N., López-Fernández, M., Nino, G.: Numerical inverse Laplace transform for convection-diffusion equations. Mathematics of Computation **89**(323), 1161–1191 (2020)

[26] Hackbusch, W.: Elliptic Differential Equations: Theory and Numerical Treatment. Springer Series in Computational Mathematics. Springer Berlin Heidelberg (2017)

[27] Hesthaven, J.S., Rozza, G., Stamm, B.: Certified Reduced Basis Methods for Parametrized Partial Differential Equations. Springer (2015)

[28] Holmes, P., Lumley, J.L., Berkooz, G.: Turbulence, Coherent Structures, Dynamical Systems and Symmetry. Cambridge University Press (1996)

[29] In't Hout, K., Weideman, J.: A contour integral method for the Black–Scholes and Heston equations. SIAM Journal on Scientific Computing **33**(2), 763–785 (2011)

[30] Jiang, J., Chen, Y.: Adaptive greedy algorithms based on parameter-domain decomposition and reconstruction for the reduced basis method. International Journal for Numerical Methods in Engineering **121**(23), 5426–5445 (2020)

[31] John, V., Schmeyer, E.: Finite element methods for time-dependent convection–diffusion–reaction equations with small diffusion. Computer methods in applied mechanics and engineering **198**(3-4), 475–494 (2008)

[32] Knezevic, D.J., Peterson, J.W.: A high-performance parallel implementation of the certified reduced basis method. Computer Methods in Applied Mechanics and Engineering **200**(13-16), 1455–1466 (2011)

[33] Leng, W., Ju, L.: An additive overlapping domain decomposition method for the Helmholtz equation. SIAM Journal on Scientific Computing **41**(2), A1252–A1277 (2019)

[34] LeVeque, R.: Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems. SIAM, Philadelphia, PA (2007)

[35] Lin, X.L., Ng, M.: An all-at-once preconditioner for evolutionary partial differential equations. arXiv preprint arXiv:2002.01108 (2020)

[36] Lions, J.L., Maday, Y., Turinici, G.: A "parareal" in time discretization of PDE's. Comptes Rendus de l'Académie des Sciences - Series I - Mathematics **332**, 661–668 (2001)

[37] Liu, J., Wu, S.L.: A fast block α-circulant preconditoner for all-at-once systems from wave equations. SIAM Journal on Matrix Analysis and Applications **41**(4), 1912–1943 (2020)

[38] Maday, Y., Mula, O.: An adaptive parareal algorithm. Journal of computational and applied mathematics **377**, 112915 (2020)

[39] Markovinović, R., Jansen, J.: Accelerating iterative solution methods using reduced-order models as solution predictors. International journal for numerical methods in engineering **68**(5), 525–541 (2006)

[40] Martinsson, P.G., Tropp, J.: Randomized numerical linear algebra: Foundations & algorithms. arXiv preprint arXiv:2002.01387 (2020)

[41] McDonald, E., Pestana, J., Wathen, A.: Preconditioning and iterative solution of all-at-once systems for evolutionary partial differential equations. SIAM Journal on Scientific Computing **40**(2), A1012–A1033 (2018)

[42] Mohsen, M.F.N., Baluch, M.H.: An analytical solution of the diffusion-convection equation over a finite domain. Applied mathematical modelling **7**(4), 285–287 (1983)

[43] Morton, K.W.: Numerical solution of convection-diffusion problems. Chapman & Hall (1996)

[44] Nguyen, N.C., Chen, Y.: Reduced-basis method for the iterative solution of parametrized symmetric positive-definite linear systems. arXiv preprint arXiv:1804.06363 (2018)

[45] Ong, B.W., Schroder, J.B.: Applications of time parallelization. Computing and Visualization in Science **23**(1), 1–15 (2020)

[46] Pang, H.K., Sun, H.W.: Fast numerical contour integral method for fractional diffusion equations. Journal of Scientific Computing **66**(1), 41–66 (2016)

[47] Pasetto, D., Ferronato, M., Putti, M.: A reduced order model-based preconditioner for the efficient solution of transient diffusion equations. International Journal for Numerical Methods in Engineering **109**(8), 1159–1179 (2017)

[48] Pinkus, A.: N-widths in Approximation Theory, vol. 7. Springer (2012)

[49] Quarteroni, A., Manzoni, A., Negri, F.: Reduced Basis Methods for Partial Differential Equations: An Introduction, vol. 92. Springer (2015)

[50] Quarteroni, A., Valli, A.: Numerical approximation of partial differential equations, vol. 23. Springer Science & Business Media (2008)

[51] Roos, H.G., Stynes, M., Tobiska, L.: Robust numerical methods for singularly perturbed differential equations: convection-diffusion-reaction and flow problems, vol. 24. Springer Science & Business Media (2008)

[52] Saad, Y.: A flexible inner-outer preconditioned gmres algorithm. SIAM Journal on Scientific Computing **14**(2), 461–469 (1993)

[53] Sen, S.: Reduced-basis approximation and a posteriori error estimation for many-parameter heat conduction problems. Numerical Heat Transfer, Part B: Fundamentals **54**(5), 369–389 (2008)

[54] Sheen, D., Sloan, I., ThomÃŠe, V.: A parallel method for time-discretization of parabolic problems based on contour integral representation and quadrature. Mathematics of Computation **69**(229), 177–195 (2000)

[55] Sheen, D., Sloan, I.H., Thomée, V.: A parallel method for time discretization of parabolic equations based on laplace transformation and quadrature. IMA Journal of Numerical Analysis **23**(2), 269–299 (2003)

[56] Silvester, D., Elman, H., Ramage, A.: Incompressible Flow and Iterative Solver Software (IFISS) version 3.6 (2020). `http://www.manchester.ac.uk/ifiss/`

[57] Simoncini, V., Szyld, D.B.: Flexible inner-outer Krylov subspace methods. SIAM Journal on Numerical Analysis **40**(6), 2219–2239 (2002)

[58] Simoncini, V., Szyld, D.B.: Theory of inexact Krylov subspace methods and applications to scientific computing. SIAM Journal on Scientific Computing **25**(2), 454–477 (2003)

[59] Taus, M., Zepeda-Núñez, L., Hewett, R.J., Demanet, L.: L-sweeps: A scalable, parallel preconditioner for the high-frequency helmholtz equation. Journal of Computational Physics **420**, 109706 (2020)

[60] Varah, J.M.: A lower bound for the smallest singular value of a matrix. Linear Algebra and its Applications **11**(1), 3–5 (1975)

[61] Wu, S., Zhou, Z.: Parallel-in-time high-order BDF schemes for diffusion and subdiffusion equations. arXiv preprint arXiv:2007.13125 (2020)

[62] Wu, S.L., Liu, J.: A parallel-in-time block-circulant preconditioner for optimal control of wave equations. SIAM Journal on Scientific Computing **42**(3), A1510–A1540 (2020)

[63] Wu, S.L., Zhou, T.: Diagonalization-based parallel-in-time algorithms for parabolic pde-constrained optimization problems. ESAIM: Control, Optimisation and Calculus of Variations **26**, 88 (2020)