DELFT UNIVERSITY OF TECHNOLOGY

REPORT 15-06

Preconditioning Optimal In-Domain Control of Navier-Stokes Equation Using Multilevel Sequentially Semiseparable Matrix Computations

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 2015

Preconditioning Optimal In-Domain Control of Navier-Stokes Equation Using Multilevel Sequentially Semiseparable Matrix Computations

Yue Qiu[a,*], Martin B. van Gijzen[a], Jan-Willem van Wingerden[b], Michel Verhaegen[b], Cornelis Vuik[a]

[a] *Delft Institute of Applied Mathematics, Delft University of Technology*
[b] *Delft Center for Systems and Control, Delft University of Technology*

Abstract

In this manuscript, we study preconditioning techniques for optimal in-domain control of the Navier-Stokes equation, where the control only acts on a few parts of the domain. Optimization and linearization of the optimal in-domain control problem results in a generalized linear saddle-point system. The Schur complement for the generalized saddle-point system is very difficult or even impossible to approximate. This prohibits satisfactory performance of the standard block preconditioners. We apply the multilevel sequentially semiseparable (MSSS) preconditioner to the underlying system. Compared with standard block preconditioning techniques, the MSSS preconditioner computes an approximate factorization of the global generalized saddle-point matrix up to a prescribed accuracy in linear computational complexity. This in turn gives parameter independent convergence for MSSS preconditioned Krylov solvers. We use a simple wind farm control example to illustrate the performance of the MSSS preconditioner. We also compare with the performance of the state-of-the-art preconditioning techniques. Our results show the superiority of the MSSS preconditioning techniques to standard block preconditioning techniques for optimal in-domain control of the Navier-Stokes equation.

Keywords: in-domain flow control, MSSS preconditioners, generalized saddle-point system, Navier-Stokes control

1. Introduction

Nowadays, optimal control problems in practice are mostly solved with nonlinear programming (NLP) methods based on some discretization strategies of the original continuous problems in the functional space [1]. Once the optimization problem is discretized, the optimization variable is reduced to a finite-dimensional space. This results in a parameter optimization problem [2]. Simultaneous strategies, explicitly perform a discretization of the partial differential equations (PDEs) that prescribe the dynamics

*Corresponding author
Email address: `y.qiu@gmx.us` (Yue Qiu)

of the system as well as the cost function, i.e., PDE simulation and the optimization procedure proceed simultaneously, cf. [3, 4, 5]. Sequential strategies, on the other hand, just parameterize the control input and employ numerical schemes in a black-box manner by utilizing the implicit function theorem (IFT) [2, 6, 7]. This approach turns out to be very efficient when the dimension of the control variable is much smaller than the system states that are described by the partial differential equations [2, 7], where optimal shape design [8], boundary control of fluid dynamics [9, 10] are applications for this type of optimization approaches. For the simultaneous approach, its high potential in terms of efficiency and robustness turns out to be very difficult to be realized when the sizes of the states and inputs are very large. This yields a very large optimization problem, where the equality constraints by PDEs are appended to the augmented cost function. The Lagrangian multipliers, the number of which is the same as the state variables of PDEs, make the size of the optimization problem even bigger.

Just like many problems in science and engineering, the most time-consuming part for the optimal control of PDEs is to solve a series of linear systems arising from the simulation of PDEs [3]. With increasing improvement of computational resources and the advancement of numerical techniques, very-large problem can be taken into consideration [11]. An important building block for the optimal control of PDEs is the preconditioning techniques to accelerate the simulation of PDEs. Many efforts have been dedicated to the development of efficient and robust preconditioning techniques for this types of problems [12, 13, 14, 15, 16]. These research projects are devoted to preconditioning control problems of the tracking type where the control is distributed throughout the domain. For the case of the in-domain control where the control only acts on a few parts of the domain where the actuators are placed, or the boundary control case that only the boundary condition can be controlled, preconditioning techniques for this in-domain control problems do not give satisfactory performance. This is because the developed preconditioning techniques highly depend on an efficient approximation of the Schur complement of the block linear system arising from the discretized optimality condition, cf. conclusion parts of [15, 17] for a discussion. Some research for optimal in-domain control problems focuses on developing novel computational techniques for the specific objective [1, 18, 19] without considering efficient preconditioning techniques.

In this manuscript, we focus on designing efficient and robust preconditioning techniques for optimal in-domain control of the Navier-Stokes equation and use a simple wind farm control example to illustrate the performance of our preconditioning technique. Our contributions include, (1) By formulating the optimal in-domain control of the Navier-Stokes problem as a generalized saddle-point problem using the implicit function theorem (IFT), we can reuse the preconditioners for the linearized Navier-Stokes equation to solve the adjoint equations for the computations of the gradient and Hessian matrix. This reudces the computational cost significantly; (2) We study the multilevel sequentially semiseparable (MSSS) preconditioner for the generalized saddle-point system. In contrast to the standard block preconditioners that require to approximate the Schur complement of the block linear system, the MSSS preconditioner computes an approximate factorization of the global system matrix up to a prescribed accuracy in linear computational complexity. This is a big advantage over the standard block preconditioners; (3) We evaluate the performance of the MSSS preconditoner using incompressible flow and fast iterative solver (IFISS) [20][1] and compare with the state-of-the-art preconditioning techniques; (4) Our analysis show that the computational cost can be further reduced by applying block Krylov methods to solve a linear system with multiple left-hand sides and multiple right-hand sides for the computations of the gradient and Hessian matrix.

The structure of this manuscript is organized as follows. In Section 2, we use a simple wind farm control example to formulate an optimal in-domain Navier-Stokes control problem. Applying numerical approaches to solve this optimization problem, we obtain a generalized saddle-point system in Section 3. To preconditioning this generalized saddle-point system, we study the MSSS preconditioning technique in Section 4. In Section 5, we perform numerical experiments to illustrate the performance of the MSSS preconditioning techniques for this type of problems. We also study the state-of-the-art preconditioning techniques in Section 5 as a comparison. Conclusions are drawn in Section 6.

2. Problem Formulation

In this section, we use wind farm control as an example to formulate the in-domain Navier-Stokes control problem. Many research activities illustrate that operating all the turbines in a wind farm at their own optimal state gives suboptimal performance of the overall wind farm [21, 22]. Wind farm control aims to optimize the total power captured from the wind by taking coordinating control strategies to the turbines in the wind farm. By appropriately choosing the computational domain for the flow, the wind farm control can be formulated as an optimal flow control problem where the dynamics of the flow are described by the incompressible Navier-Stokes equation. For the wind farm control, the control only acts on a few parts of the domain where the turbines are located. This in turn gives an optimal in-domain flow control problem. In the next part, we aim to build a simple wind farm model and use this model to formulate the in-domain Navier-Stokes control problem.

2.1. Fluid Dynamics

Some efforts have been devoted to develop a suitable model to simulate the wake effect in the wind farm, cf. [23, 24] for a general survey and [21, 25] for recent developments. In general there exist two approaches for modeling of the wake. One is the heuristic approach that does not solve a flow equation but uses some rules of thumb [21, 26], a typical example is the Park model [21]. The second approach is solving an incompressible Navier-Stokes or Euler equation, cf. [11, 27]. In this manuscript, we use the second approach to model the flow in the wind farm. Moreover, some recent research try to take the boundary layer and some physical behavior into account. This in turn requires a more complicated turbulent flow model for the wind farm simulation study [11, 24, 28]. However, these research activities do not take efficient preconditioning techniques into account but focus on physical properties of the flow for the wind farm simulation. In this manuscript, we focus on designing efficient and robust preconditioning techniques

[1]IFISS is a computational laboratory for experimenting with state-of-the-art preconditioned iterative solvers for the discrete linear equation systems that arise in incompressible flow modeling

and we evaluate the performance of our preconditioning techniques by IFISS. We only consider flow problems that can be addressed within the framework of IFISS.

Consider the stationary incompressible Navier-Stokes equation in $\Omega \in \mathbb{R}^2$ that is given by

$$\begin{aligned} -\nu\Delta\overrightarrow{u} + (\overrightarrow{u}\cdot\nabla)\overrightarrow{u} + \nabla p &= \overrightarrow{f} \\ \nabla\cdot\overrightarrow{u} &= 0 \end{aligned} \tag{1}$$

where ν is the kinematic viscosity, $\overrightarrow{u}$ is the velocity field, p denotes the pressure, $\overrightarrow{f}$ is a source term. Here Ω is a bounded domain with its boundary given by $\Gamma = \partial\Omega = \partial\Omega_D \cup \partial\Omega_N$, where $\partial\Omega_D$ denotes the boundary where Dirichlet boundary conditions are prescribed and $\partial\Omega_N$ represents the boundary where Neumann boundary conditions are imposed. The Reynolds number $Re \in \mathbb{R}^+$ describes the ratio of the inertial and viscous forces within the fluid [29] and is defined by

$$Re \triangleq \frac{u_r L_r}{\nu}, \tag{2}$$

where $u_r \in \mathbb{R}^+$ is the reference velocity, $L_r \in \mathbb{R}^+$ is the reference distance that the flow travels. The Reynolds number plays an important role in the flow equation that describes whether the flow under consideration is laminar or turbulent. In many engineering problems, turbulent flow happens when the Reynolds number $Re > 2000$ [29]. In the case of flow through a straight pipe with a circular cross-section, at a Reynolds number below a critical value of approximately 2040, fluid motion will ultimately be laminar, whereas at larger Reynolds numbers, the flow can be turbulent [30]. Since we focus on efficient preconditioning techniques for optimal in-domain flow control using IFISS in this manuscript and no turbulent flow model is included in IFISS, we consider a flow with Reynolds number $Re = 2000$, although this does not correspond to practical flow for wind farm control.

To study the aerodynamics of the wind farm, we cannot set an infinite dimensional computational domain. We can prescribe suitable boundary conditions for the flow that in turn gives a finite domain. We set a standard reference domain $\Omega = [-1, 1] \times [-1, 1]$ for the wind farm simulation study in Figure 1. The reference velocity u_r is set to 1.

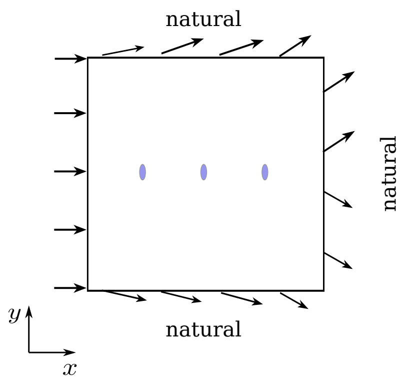


Figure 1: Computational domain for wind farm simulation

The turbines in the wind farm are located in a line that follows the stream direction, and the leftmost turbine is placed at the center of the domain. Such configurations are widely used in the wind farm simulation studies [11, 21, 27]. Here the diameter of the turbine is set to be $^1/_{64}$ of the reference domain in Figure 1. The distance between turbines is set to be 5 times of the diameter of the turbines [27]. Constant inflow is imposed on the left boundary and is given by

$$u_x = u_c, \quad u_y = 0, \tag{3}$$

or equivalently

$$\overrightarrow{u} \cdot \overrightarrow{n} = -u_c. \tag{4}$$

Here, $\overrightarrow{n}$ is the unit normal vector of the boundary that points outwards. For top, right, and bottom boundary that are far away from the turbines, the flow is considered to be free stream and the natural boundary outflow condition given by (5) is prescribed,

$$\nu \frac{\partial \overrightarrow{u}}{\partial \overrightarrow{n}} - p\overrightarrow{n} = \mathbf{0}. \tag{5}$$

Here, $\frac{\partial \overrightarrow{u}}{\partial \overrightarrow{n}}$ is the Gâteaux derivative at $\partial\Omega_N$ in the direction of $\overrightarrow{n}$. This boundary condition states that the flow can move freely across the boundary by resolving the Navier-Stokes equation (1). Associated with the prescribed boundary condition (4) (5), the resolved velocity without outer source for the Navier-Stokes equation (1) is shown in Figure 2.

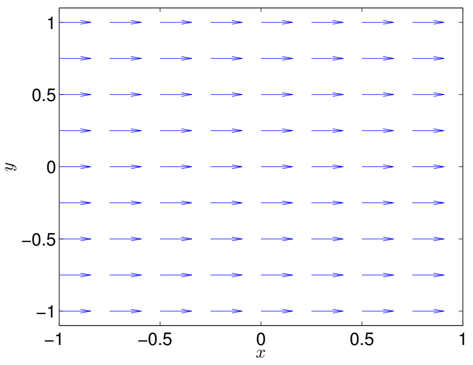


Figure 2: Resolved velocity without outer source

2.2. Wind Turbines

Wind turbines can be modeled as outer sources that interact with the flow. Two widely used methods for wind turbine modelling are the actuator disk method (ADM) [31] and the actuator line method [32]. In this manuscript, we use the ADM method for wind turbines, where the thrust force is denoted by,

$$f = -C_T \rho A \hat{u}_d^2. \tag{6}$$

Here $\hat{u}_d$ is the average axial flow velocity at the turbine disk, C_T is the disk based thrust coefficient, ρ is the density of the flow and A is the area that is swept by the turbine blades.

2.3. Objective Function

The wind farm control aims to maximize the total power captured by all the wind turbines in the wind farm, which can be represented as

$$P_t = \sum_{j=1}^{N_t} -f_j \hat{u}_j = \rho A \sum_{j=1}^{N_t} C_{T_j} \hat{u}_j^3, \tag{7}$$

where N_t is the number of turbines in the wind farm, $\hat{u}_j$ is the uniform disk averaged axial flow speed of the j-th turbine.

To summarize, the in-domain Navier-Stokes control problem can be formulated as the following optimization problem,

$$\begin{aligned} \min_{C_T, \overrightarrow{u}} \quad & -\sum_{j=1}^{N_t} C_{T_j} \hat{u}_j^3 \\ \text{s.t.} \quad & -\nu \Delta \overrightarrow{u} + (\overrightarrow{u} \cdot \nabla) \overrightarrow{u} + \nabla p = \overrightarrow{f}(C_T, \overrightarrow{u}), \\ & \nabla \cdot \overrightarrow{u} = 0, \\ & 0 \leq C_{T_j} \leq 1, \ (j = 1, \ldots, N_t). \end{aligned} \tag{8}$$

Here C_{T_j} varies from 0 to 1 based on individual pitch control and yaw control of the turbine j, $\overrightarrow{f}(C_T, \overrightarrow{u})$ is a nonlinear function and it is of value (6) at the position where turbines are placed and 0 elsewhere, and $C_T = \begin{bmatrix} C_{T_1} & C_{T_2} & \cdots & C_{T_{N_t}} \end{bmatrix}^T$.

3. Reduced Nonlinear Programming

In the previous section, we formulated an in-domain control problem (8) by using a wind farm control example. The size of the control variable N_t, which is the number of turbines, is much smaller than the size of the state variables (number of velocity and pressure grid unknowns). Therefore, we use the sequential approach that is based on the implicit function theorem to solve a reduced optimization problem.

3.1. Reduced Optimization Problem

Denote the equality constraints in (8) for the flow equation by $h(C_T, \phi) = 0$ where $\phi = (\overrightarrow{u}, p)$, and the objective function by $g(C_T, \phi)$, then the optimization problem (8) can be written as

$$\begin{aligned} \min_{C_T, \phi} \quad & g(C_T, \phi) \\ \text{s.t.} \quad & h(C_T, \phi) = 0, \\ & 0 \leq C_T \leq 1. \end{aligned} \tag{9}$$

The equality constraint in (8) implies that the velocity and pressure ϕ is a function of $\overrightarrow{f}$. For the existence of this function, cf. [7] for a proof. Since $\overrightarrow{f}$ is an explicit function of C_T, ϕ is a function of C_T and denote this function by $\phi = s(C_T)$. The function $s(C_T)$ is not explicitly computed but is obtained implicitly by solving the flow equation (1) using given C_T.

By using the implicit function theorem, we can re-write the optimization problem in a reduced form,

$$\begin{aligned} \min_{C_T} \quad & g(C_T, s(C_T)) \\ \text{s.t.} \quad & 0 \leq C_T \leq 1. \end{aligned} \tag{10}$$

Newton-type methods, which are second order methods, are well-suited to solve this type of nonlinear programming (NLP) problems. An alternative approach to solve this type of problem is the (reduced) sequentially quadratic programming ((R)SQP) [33]. For the reduced optimization problem (10), these two types of methods are quite similar and we refer to [7] for a detailed discussion.

In this section, we apply Newton's method to solve the reduced NLP problem (10). The reason to choose the Newton's method is that the Hessian matrix for this problem is of small size and can be computed explicitly. Moreover, we can reuse the information from the last Newton step of solving the nonlinear flow equation to compute the gradient and the Hessian matrix, which makes Newton's method computationally competitive for this optimization problem. This will be explained in the following part. The Newton's method for this problem is described by Algorithm 1.

Algorithm 1 Reduced Newton's algorithm for (9)

```
 1: procedure OPT(Wind)                 ▷ Optimization procedure for wind farm control
 2:     Input: Initial guess $C_T^{(0)}$, maximal optimization steps $\texttt{it}_{\max}$, stop tolerance $\varepsilon_0$
 3:     while $\|\nabla g_k\| > \varepsilon_0$ && $k \leq \texttt{it}_{\max}$ do          ▷ outer iteration
 4:         solve $h(C_T^{(k)}, \phi) = 0$ to compute $\phi^{(k)}$                 ▷ inner iteration
 5:         evaluate the gradient $\nabla g_k$ at $(C_T^{(k)}, \phi^{(k)})$
 6:         evaluate the Hessian matrix $H_k$ at $(C_T^{(k)}, \phi^{(k)})$
 7:         compute the update $\Delta C_T^{(k)} = \arg\min \Delta C_T^T H_k \Delta C_T + \nabla g_k^T \Delta C_T$
 8:         $C_T^{(k+1)} \leftarrow C_T^{(k)} + \Delta C_T^{(k)}$
 9:         Check inequality constraints by projection
10:         if $C_{T_j} > 1$ then
11:             $C_{T_j} = 1$                                    ▷ project on boundary
12:         else if $C_{T_j} < 0$ then
13:             $C_{T_j} = 0$                                    ▷ project on boundary
14:         end if
15:         $k \leftarrow k + 1$
16:     end while
17:     Output: Optimal control variable $C_T$ and corresponding solution of $u$
18: end procedure
```

In Algorithm 1, we denote the optimization step as the outer iteration, and at each outer iteration, we need to solve a Navier-Stokes equation with nonlinear right-hand side. This step is denoted by the inner iteration in line 4 of the algorithm. From the description of Algorithm 1, it is clear that the most time-consuming part for this optimization problem is the solution of the nonlinear flow equation and the computations of the gradient and Hessian matrix. Therefore, efficient numerical methods need to be deployed.

3.2. Computations of the Flow Equation

At each outer iteration of Algorithm 1, we need to solve a nonlinear flow equation that has a nonlinear right-hand side at step 4. We explicitly write this equation in decomposed form as

$$
\begin{aligned}
-\nu\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right)u_x + \overbrace{\left(u_x\frac{\partial}{\partial x}+u_y\frac{\partial}{\partial y}\right)u_x}^{\text{velocity convection}} + \frac{\partial}{\partial x}p &= f_x(C_T,u_x,u_y),\\
-\nu\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right)u_y + \underbrace{\left(u_x\frac{\partial}{\partial x}+u_y\frac{\partial}{\partial y}\right)u_y}_{\text{velocity convection}} + \frac{\partial}{\partial y}p &= f_y(C_T,u_x,u_y),\\
\frac{\partial}{\partial x}u_x + \frac{\partial}{\partial y}u_y &= 0.
\end{aligned}
\tag{11}
$$

Equation (11) is a nonlinear equation where the nonlinearity is caused by both the velocity convection operator and the nonlinear right-hand side. To solve such a nonlinear equation (11), we apply Newton's method. At each Newton iteration of step 4 in Algorithm 1, we need to solve a linear system of the following type,

$$
\begin{bmatrix}
\nu K + N + J_{xx}^n + J_{xx}^f & J_{xy}^n + J_{xy}^f & B_x^T\\
J_{yx}^n + J_{yx}^f & \nu K + N + J_{yy}^n + J_{yy}^f & B_y^T\\
B_x & B_y & 0
\end{bmatrix}
\begin{bmatrix}\Delta u_x\\ \Delta u_y\\ \Delta p\end{bmatrix}
=
\begin{bmatrix}a\\ b\\ c\end{bmatrix},
\tag{12}
$$

after discretizing the nonlinear partial differential equation (11) using mixed finite element method. Here N is the convection matrix, $J_{(\cdot)}^n$ denote the Jacobian matrices from the nonlinear velocity convection term, and $J_{(\cdot)}^f$ represent the Jacobian matrices from the nonlinear right-hand side. Since only a small part of the domain is controlled, f_x and f_y are zero almost everywhere in the domain except in the vicinity where the turbines are placed. This in turn gives singular Jacobian matrices $J_{(\cdot)}^f$.

Comparing system (12) with the standard linearized Navier-Stokes equation by the Newton's method given by

$$
\begin{bmatrix}
\nu K + N + J_{xx}^n & J_{xy}^n & B_x^T\\
J_{yx}^n & \nu K + N + J_{yy}^n & B_y^T\\
B_x & B_y & 0
\end{bmatrix}
\begin{bmatrix}\Delta u_x\\ \Delta u_y\\ \Delta p\end{bmatrix}
=
\begin{bmatrix}a\\ b\\ c\end{bmatrix},
\tag{13}
$$

we see that the linearized flow equation (12) is a perturbed linearized Navier-Stokes equation with singular perturbation in the matrix blocks that correspond to the velocity. Re-write equation (13) in a compact form as

$$
\begin{bmatrix}A & B^T\\ B & 0\end{bmatrix}
\begin{bmatrix}\Delta u\\ \Delta p\end{bmatrix}
=
\begin{bmatrix}f\\ g\end{bmatrix},
\tag{14}
$$

and the equation (12) is given by a perturbed form

$$\begin{bmatrix} A + J^f & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \Delta u \\ \Delta p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}. \tag{15}$$

The linear system (15) is large, sparse, and highly indefinite. It belongs to the type of generalized saddle-point system. Efficient preconditioning techniques are needed to solve such systems using Krylov solvers. Standard preconditioning techniques for such 2-by-2 system highly depend on efficient approximation of the Schur complement, or computing a spectrally equivalent Schur complement. For the linearized Navier-Stokes problem (14), there exits some well-established preconditioning techniques such as the SIMPLE method [34], augmented Lagrangian preconditioner [35], pressure convection-diffusion (PCD) preconditioner [36], et al. However, the aforementioned preconditioners do not perform well to solve the perturbed linear system (15) because the Schur complement for the perturbed linear system is even more difficult to approximate. Numerical experiments in Section 5 illustrate the performance of block preconditioners for (15).

As we will see in Section 4, all the blocks of the matrix in (12) have an MSSS structure, it is therefore natural to permute the matrix with MSSS blocks into a global MSSS matrix. With this permutation, we can compute an approximate factorization. This factorization gives a global MSSS preconditioner for the system (12). This global preconditioner does not require to approximate the Schur complement of the generalized saddle-point system (15). Details of this preconditioning technique will be introduced in Section 4.

3.3. Computations of Partial Derivatives

Denote the reduced gradient of the cost function by

$$\nabla g = \begin{bmatrix} \frac{\partial}{\partial C_{T_1}} g & \frac{\partial}{\partial C_{T_2}} g & \cdots & \frac{\partial}{\partial C_{T_{N_t}}} g \end{bmatrix}^T. \tag{16}$$

The gradient can be computed using

$$\frac{\partial}{\partial C_{T_j}} g = \frac{\partial g}{\partial C_{T_j}} + \frac{\partial g}{\partial u_x} \frac{\partial u_x}{\partial C_{T_j}} + \frac{\partial g}{\partial u_y} \frac{\partial u_y}{\partial C_{T_j}}, \quad (j = 1, 2, \ldots, N_t). \tag{17}$$

Since the cost function g is an analytic function of C_T, u_x and u_y, the partial derivatives $\frac{\partial g}{\partial C_{T_j}}$, $\frac{\partial g}{\partial u_x}$, and $\frac{\partial g}{\partial u_y}$ are trivial to compute. Next, we show how to compute the partial derivatives $\frac{\partial u_x}{\partial C_{T_j}}$, and $\frac{\partial u_y}{\partial C_{T_j}}$.

Assume that u_x, u_y, and p are twice differentiable with respect to $C_{T_j}\,(j = 1, 2, \ldots, N_t)$, and that the first and second order derivatives have continuous second order derivative in Ω, i.e.,

$$\frac{\partial u_x}{\partial C_{T_j}}, \quad \frac{\partial u_y}{\partial C_{T_j}}, \quad \frac{\partial p}{\partial C_{T_j}} \in C^2(\Omega),$$

and

$$\frac{\partial^2 u_x}{\partial C_{T_i} \partial C_{T_j}}, \quad \frac{\partial^2 u_y}{\partial C_{T_i} \partial C_{T_j}}, \quad \frac{\partial^2 p}{\partial C_{T_i} \partial C_{T_j}} \in C^2(\Omega),$$

for $(i,\ j = 1, 2, \ldots, N_t)$.

According to the flow equation (11), we have the first order derivative given by

$$
\begin{aligned}
-\nu \overbrace{\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right)}^{\text{diffusion operator}}\left(\frac{\partial u_x}{\partial C_{T_j}}\right)+\overbrace{\left(u_x\frac{\partial}{\partial x}+u_y\frac{\partial}{\partial y}\right)}^{\text{convection operator}}\left(\frac{\partial u_x}{\partial C_{T_j}}\right)+\overbrace{\left(\frac{\partial u_x}{\partial x}\frac{\partial u_x}{\partial C_{T_j}}+\frac{\partial u_x}{\partial y}\frac{\partial u_y}{\partial C_{T_j}}\right)}^{\text{linear term}}& \\
+\frac{\partial}{\partial x}\left(\frac{\partial p}{\partial C_{T_j}}\right)=\frac{\partial}{\partial C_{T_j}}f_x(C_T,u_x,u_y)&, \\
-\nu \underbrace{\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right)}_{\text{diffusion operator}}\left(\frac{\partial u_y}{\partial C_{T_j}}\right)+\underbrace{\left(u_x\frac{\partial}{\partial x}+u_y\frac{\partial}{\partial y}\right)}_{\text{convection operator}}\left(\frac{\partial u_y}{\partial C_{T_j}}\right)+\underbrace{\left(\frac{\partial u_y}{\partial x}\frac{\partial u_x}{\partial C_{T_j}}+\frac{\partial u_y}{\partial y}\frac{\partial u_y}{\partial C_{T_j}}\right)}_{\text{linear term}}& \\
+\frac{\partial}{\partial y}\left(\frac{\partial p}{\partial C_{T_j}}\right)=\frac{\partial}{\partial C_{T_j}}f_y(C_T,u_x,u_y)&, \\
\frac{\partial}{\partial x}\left(\frac{\partial u_x}{\partial C_{T_j}}\right)+\frac{\partial}{\partial y}\left(\frac{\partial u_y}{\partial C_{T_j}}\right)=0.&
\end{aligned}
\tag{18}
$$

Here the partial differential equation (18) is obtained by computing the first order derivative with respect to $C_{T_j}(j=1,2,\ldots,N_t)$ for the flow equation (11) and making use of the linearity of the diffusion operator and the divergence operator.

Note that equation (18) is a linear partial differential equation. It is often called the adjoint equation of (11). Homogeneous Dirichlet boundary conditions are imposed on the constant inflow boundary and natural boundary conditions are prescribed for the outflow boundary for the adjoint equation (18). We use a mixed-finite element method to discretize (18) and use the following notations

$$
\chi_j=\begin{bmatrix}\left(\frac{\partial u_x}{\partial C_{T_j}}\right)^T & \left(\frac{\partial u_y}{\partial C_{T_j}}\right)^T & \left(\frac{\partial p}{\partial C_{T_j}}\right)^T\end{bmatrix}^T.
$$

We reuse the same notation to denote its discrete analog to get a linear system given by

$$
\underbrace{\begin{bmatrix}\nu K+N_l+J_{xx}^{n_l}+J_{xx}^f & J_{xy}^{n_l}+J_{xy}^f & B_x^T\\ J_{yx}^{n_l}+J_{yx}^f & \nu K+N_l+J_{yy}^{n_l}+J_{yy}^f & B_y^T\\ B_x & B_y & 0\end{bmatrix}}_{\mathcal{A}}\chi_j=\zeta_j, \tag{19}
$$

where N_l is the convection matrix, $J_{xx}^{n_l}$, $J_{xy}^{n_l}$, $J_{yx}^{n_l}$, and $J_{yy}^{n_l}$ are the corresponding Jacobian matrices by linearizing the nonlinear velocity convection operator, respectively. J_{xx}^f, J_{xy}^f, J_{yx}^f, and J_{yy}^f are the associating Jacobian matrices that linearize the nonlinear right-hand side f with respect to C_{T_j}. They are all from the last Newton step of the linearized flow equation (12). The right-hand side vector ζ_j is obtained by discretizing known variables in (18). Therefore, the matrix $\mathcal{A}$ is nothing but the linear system matrix from the last Newton step for solving the flow equation (12). The computed preconditioner for the last Newton step to solve the flow equation can be reused. Therefore, the computational work is much reduced.

By stacking χ_j, and ζ_j ($j=1,\ 2,\ \ldots,\ N_t$) as

$$
\chi=\begin{bmatrix}\chi_1 & \chi_2 & \ldots & \chi_{N_t}\end{bmatrix},\ \text{and}\ \zeta=\begin{bmatrix}\zeta_1 & \zeta_2 & \ldots & \zeta_{N_t}\end{bmatrix},
$$

we obtain the following linear equations with multiple left-hand sides and multiple right-hand sides

$$\mathcal{A}\chi = \zeta. \tag{20}$$

Here the size of unknowns N_t, is much smaller than the problem size. Block Krylov subspace methods, such as block IDR(s) [37, 38], are well-suited to solve all the unknowns simultaneously for this type of problems.

Next, we use the same idea as above to compute the Hessian matrix H, which is given by

$$H = \begin{bmatrix} \frac{\partial^2}{\partial C_{T_1}^2} g & \cdots & \cdots & \frac{\partial^2}{\partial C_{T_1}\partial C_{T_{N_t}}} g \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ \frac{\partial^2}{\partial C_{T_{N_t}}\partial C_{T_1}} g & \cdots & \cdots & \frac{\partial^2}{\partial C_{T_{N_t}}^2} g \end{bmatrix}.$$

According to (17), we have

$$\begin{aligned} \frac{\partial^2}{\partial C_{T_j}\partial C_{T_k}} g = \frac{\partial^2 g}{\partial C_{T_j}\partial C_{T_k}} + \frac{\partial^2 g}{\partial u_x^2}\frac{\partial u_x}{\partial C_{T_k}}\frac{\partial u_x}{\partial C_{T_j}} + \frac{\partial g}{\partial u_x}\frac{\partial^2 u_x}{\partial C_{T_j}\partial C_{T_k}} \\ + \frac{\partial^2 g}{\partial u_y^2}\frac{\partial u_y}{\partial C_{T_k}}\frac{\partial u_y}{\partial C_{T_j}} + \frac{\partial g}{\partial u_y}\frac{\partial^2 u_y}{\partial C_{T_j}\partial C_{T_k}}. \end{aligned} \tag{21}$$

Since g is an analytic function of C_T, u_x, u_y and we have already computed the first order derivative $\frac{\partial u_x}{\partial C_{T_i}}$, and $\frac{\partial u_y}{\partial C_{T_i}}$ in the previous step, we just need to compute the second order derivatives $\frac{\partial^2 u_x}{\partial C_{T_j}\partial C_{T_k}}$, and $\frac{\partial^2 u_y}{\partial C_{T_j}\partial C_{T_k}}$ for the computations of the Hessian matrix. Here we use the same idea as we used in the previous part to compute the first order derivatives.

Compute the partial derivative with respect to C_{T_k} using the adjoint equation (18), we get

$$\begin{aligned} & -\nu \overbrace{\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)}^{\text{diffusion operator}} \left(\frac{\partial^2 u_x}{\partial C_{T_j}\partial C_{T_k}}\right) + \overbrace{\left(u_x\frac{\partial}{\partial x} + u_y\frac{\partial}{\partial y}\right)}^{\text{convection operator}} \left(\frac{\partial^2 u_x}{\partial C_{T_j}\partial C_{T_k}}\right) \\ & + \overbrace{\left(\frac{\partial u_x}{\partial x}\frac{\partial^2 u_x}{\partial C_{T_j}\partial C_{T_k}} + \frac{\partial u_x}{\partial y}\frac{\partial^2 u_y}{\partial C_{T_j}\partial C_{T_k}}\right)}^{\text{linear term}} + \frac{\partial}{\partial x}\left(\frac{\partial^2 p}{\partial C_{T_j}\partial C_{T_k}}\right) \\ & + \overbrace{\left(\frac{\partial u_x}{\partial C_{T_k}}\frac{\partial}{\partial x}\left(\frac{\partial u_x}{\partial C_{T_j}}\right) + \frac{\partial u_y}{\partial C_{T_k}}\frac{\partial}{\partial y}\left(\frac{\partial u_x}{\partial C_{T_j}}\right)\right)}^{\text{known}} \\ & + \overbrace{\left(\frac{\partial u_x}{\partial C_{T_j}}\frac{\partial}{\partial x}\left(\frac{\partial u_x}{\partial C_{T_k}}\right) + \frac{\partial u_y}{\partial C_{T_j}}\frac{\partial}{\partial y}\left(\frac{\partial u_x}{\partial C_{T_k}}\right)\right)}^{\text{known}} = \frac{\partial^2}{\partial C_{T_j}\partial C_{T_k}} f_x(C_T, u_x, u_y), \end{aligned} \tag{22}$$

$$
\begin{aligned}
&-\nu\underbrace{\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right)}_{\text{diffusion operator}}\left(\frac{\partial^2 u_y}{\partial C_{T_j}\partial C_{T_k}}\right)+\underbrace{\left(u_x\frac{\partial}{\partial x}+u_y\frac{\partial}{\partial y}\right)}_{\text{convection operator}}\left(\frac{\partial^2 u_y}{\partial C_{T_j}\partial C_{T_k}}\right)\\
&+\underbrace{\left(\frac{\partial u_y}{\partial x}\frac{\partial^2 u_x}{\partial C_{T_j}\partial C_{T_k}}+\frac{\partial u_y}{\partial y}\frac{\partial^2 u_y}{\partial C_{T_j}\partial C_{T_k}}\right)}_{\text{linear term}}+\frac{\partial}{\partial x}\left(\frac{\partial^2 p}{\partial C_{T_j}\partial C_{T_k}}\right)\\
&+\underbrace{\left(\frac{\partial u_x}{\partial C_{T_k}}\frac{\partial}{\partial x}\left(\frac{\partial u_y}{\partial C_{T_j}}\right)+\frac{\partial u_y}{\partial C_{T_k}}\frac{\partial}{\partial y}\left(\frac{\partial u_y}{\partial C_{T_j}}\right)\right)}_{\text{known}} \qquad (23)\\
&+\underbrace{\left(\frac{\partial u_x}{\partial C_{T_j}}\frac{\partial}{\partial x}\left(\frac{\partial u_y}{\partial C_{T_k}}\right)+\frac{\partial u_y}{\partial C_{T_j}}\frac{\partial}{\partial y}\left(\frac{\partial u_y}{\partial C_{T_k}}\right)\right)}_{\text{known}}=\frac{\partial^2}{\partial C_{T_j}\partial C_{T_k}}f_y(C_T,u_x,u_y),\\
&\frac{\partial}{\partial x}\left(\frac{\partial^2 u_x}{\partial C_{T_j}\partial C_{T_k}}\right)+\frac{\partial}{\partial y}\left(\frac{\partial^2 u_y}{\partial C_{T_j}\partial C_{T_k}}\right)=0.
\end{aligned}
$$

We see that (22) (23) is a linear partial differential equation and it is also an adjoint equation of the flow equation (11). Boundary conditions for the adjoint equation (22) are set by homogeneous Dirichlet boundary conditions on the constant inflow boundary and natural boundary conditions for the outflow boundary. By using mixed finite element method to discretize the equation (22) (23), we have

$$
\underbrace{\begin{bmatrix}\nu K+N_l+J_{xx}^{n_l}+J_{xx}^{f'} & J_{xy}^{n_l}+J_{xy}^{f'} & B_x^T\\ J_{yx}^{n_l}+J_{yx}^{f'} & \nu K+N_l+J_{yy}^{n_l}+J_{yy}^{f'} & B_y^T\\ B_x & B_y & 0\end{bmatrix}}_{\mathcal{A}'}\gamma_{jk}=\xi_{jk}, \qquad (24)
$$

Here we reuse $\frac{\partial^2}{\partial C_{T_j}\partial C_{T_k}}u_x$, $\frac{\partial^2}{\partial C_{T_j}\partial C_{T_k}}u_y$, $\frac{\partial^2}{\partial C_{T_j}\partial C_{T_k}}p$ to represent their discrete analog, respectively, and $\gamma_{jk}=\left[\frac{\partial^2}{\partial C_{T_j}\partial C_{T_k}}u_x^T \quad \frac{\partial^2}{\partial C_{T_j}\partial C_{T_k}}u_y^T \quad \frac{\partial^2}{\partial C_{T_j}\partial C_{T_k}}p^T\right]^T$.

For the linear system (24), $J_{xx}^{n_l}$, $J_{xy}^{n_l}$, $J_{yx}^{n_l}$, and $J_{yy}^{n_l}$ are the corresponding Jacobian matrices as introduced in (19). The Jacobian matrices $J_{(\cdot)}^{f'}$ is obtained by computing the second order derivatives of f_x, f_y with respect to C_T and using partial differentiation rules. It is not difficult to see that $J_{(\cdot)}^{f'}$ are identical to $J_{(\cdot)}^{f}$ in (19). Therefore, the linear system (24) has the same system matrix as the linear system (19), i.e., $\mathcal{A}'=\mathcal{A}$. Moreover, $\mathcal{A}$ is just the system matrix from the last Newton step to solve the nonlinear flow equation (11). The preconditioners computed for the last Newton step can also be reused here.

The right-hand side vector ξ_{jk} is obtained by discretizing known variables in (22) (23). By stacking the unknowns in the following way,

$$
\gamma=\begin{bmatrix}\gamma_{11} & \gamma_{12} & \dots & \gamma_{N_tN_t}\end{bmatrix} \quad \text{and} \quad \xi=\begin{bmatrix}\xi_{11} & \xi_{12} & \dots & \xi_{N_tN_t}\end{bmatrix},
$$

We get the following linear system with multiple left-hand and right-hand sides,

$$\mathcal{A}\gamma = \xi. \tag{25}$$

Here the size of unknowns N_t^2 is also much more smaller than the problem size, block Krylov methods are still well-suited to this type of problems.

4. Multilevel Sequentially Semiseparable Preconditioners

The multilevel sequentially semiseparable (MSSS) preconditioning technique was first studied for the PDE-constrained optimization problems in [39] and later extended to the computational fluid dynamics problems in [40]. The global MSSS preconditioner computes an approximate factorization of the global (generalized) saddle-point matrix up to a prescribed accuracy in linear computational complexity using MSSS matrix computations. To start with, we first introduce the sequentially semiseparable matrices and the block matrix definition for such matrices is given by Definition 4.1.

Definition 4.1 ([41]). *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*

$$A_{ij} = \begin{cases} U_i W_{i+1} \cdots W_{j-1} V_j, & \text{if} \quad i < j; \\ D_i, & \text{if} \quad i = j; \\ P_i R_{i-1} \cdots R_{j+1} Q_j, & \text{if} \quad i > j. \end{cases} \tag{26}$$

The matrices U_i, W_i, V_i, D_i, P_i, R_i, Q_i are matrices whose sizes are compatible for matrix-matrix product when their sizes are not mentioned. They are called generators for an SSS matrix.

Basic operations such as addition, multiplication and inversion are closed under the SSS matrix structure and can be performed in linear computational complexity. Multilevel sequentially semiseparable matrices generalize the SSS matrices to the multi-dimensional case. Similar to Definition 4.1 for SSS matrices, the generators representation for MSSS matrices, specifically the k-level SSS matrices, is defined in Definition 4.2.

Definition 4.2 ([39]). *The matrix A is said to be a k-level SSS matrix if all its generators are $(k-1)$-level SSS matrices. The k-level SSS matrix is the SSS matrix that satisfies Definition 4.1.*

The MSSS matrix structure can be inferred directly from the discretization of PDEs, which is studied in [39, 40]. For block linear system (27) from discretization of coupled PDEs, all the blocks F, B, and D have an MSSS structure [40].

$$\begin{bmatrix} F & B^T \\ B & D \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}. \tag{27}$$

Therefore, the permutation operation introduced in [39] can be applied to permute the system matrix of (27) into a global MSSS matrix, which gives

$$\underbrace{\Pi \begin{bmatrix} F & B^T \\ B & D \end{bmatrix} \Pi^T}_{\bar{A}} \underbrace{\Pi \begin{bmatrix} u \\ p \end{bmatrix}}_{\bar{x}} = \underbrace{\Pi \begin{bmatrix} f \\ g \end{bmatrix}}_{\bar{b}}, \tag{28}$$

where Π is a permutation matrix that is also introduced in [39]. After permutation, the matrix $\bar{A}$ is an MSSS matrix and an approximate factorization can be computed in $\mathcal{O}(r^3 N_s)$ complexity where r is the upper bound of the rank for the low-rank approximation used to approximate the off-diagonal blocks of intermediate dense matrices in this factorization and N_s is the problem size. If r is bounded by a constant small enough compared with the problem size, this factorization can be computed in linear computational complexity. Moreover, we have the following theorem that gives the bound of the MSSS preconditioned spectrum for 2D problems.

Theorem 4.1. *For a nonsingular 2-level SSS matrix $\bar{A}$, we can compute an approximate block factorization which is given by $\tilde{\bar{A}} = \tilde{L}\tilde{D}\tilde{U}$ that satisfies*

$$\left\|\bar{A} - \tilde{\bar{A}}\right\|_2 \leq \varepsilon, \quad \textit{and} \quad \left\|I - \tilde{\bar{A}}^{-1}\bar{A}\right\|_2 \leq \frac{\varepsilon}{\varepsilon_0 - \varepsilon},$$

where ε_0 is the smallest singular value of $\bar{A}$, I denotes the identity matrix, and ε satisfies

$$\varepsilon \leq 2\sqrt{n}(n-1)\tau.$$

Here n is the number of 1-level SSS blocks and n is smaller than the number of grid points in one dimension, τ is a freely chosen parameter to compute the MSSS preconditioner.

Proof: For the proof of this theorem, we refer to Lemma 3.2, Theorem 3.6, and Theorem 5.3 of [42]. □

Theorem 4.1 states that the MSSS preconditioner can be computed up to arbitrary accuracy, this in turn makes the MSSS preconditioned spectrum cluster around $(1, 0)$ and the radius of the circle that contains the preconditioned spectrum can be made arbitrarily small. This results a robust preconditioner and makes the convergence of the MSSS preconditioned Krylov solver independent of the mesh size and Reynolds number for the generalized saddle-point system (15).

In practice, ε is usually of the same order with τ, which is much smaller than the bound given by Theorem 4.1. This in turn yields a small r that is bounded by a constant independent of N_s and $r \ll N_s$, which means that the MSSS preconditioner can be computed in linear computational complexity. Numerical results in Section 5 verify this statement.

5. Numerical Experiments

In the previous sections, we formulate an optimal in-domain control problem. It is shown that this in-domain control problem can be solved by the reduced Newton method described by Algorithm 1. The biggest computational issue is to solve a nonlinear flow equation (11) by using the Newton's method and two adjoint equations (19) (24) to compute the gradient and the Hessian matrix. At each optimization step, we solve the nonlinear flow equation (11) with inner iterations. A preconditioned Krylov solver is performed at each inner iteration.

In Section 3, we showed that the linearized flow equation (12) is a perturbed linearized Navier-Stokes equation (13) with singular perturbation on the $(1, 1)$ block of the linearized Navier-Stokes system matrix. Standard preconditioning techniques for the

Navier-Stokes equation, which highly depend on efficient approximation of the Schur complement, fail to give satisfactory performance for this problem due to this perturbation. This will be illustrated by numerical results later. In this section, we evaluate the performance of the MSSS preconditioning techniques for the optimal in-domain control problem and compare with the pressure convection diffusion (PCD) preconditioner [43] that is implemented in IFISS. All the numerical experiments are implemented in MATLAB 2015a on a desktop of Intel Core i5 CPU of 3.10 GHz and 16 Gb memory with the Debian GNU/Linux 8.0 system.

5.1. Preconditioning Techniques

In this part, we report the performance of the MSSS preconditioner and the PCD preconditioner for the second inner iteration of the first outer iteration of Algorithm 1. We use the `IDR(s)` method [44] to solve the preconditioned system. The preconditioned `IDR(s)` solver is stopped if the the 2-norm of the residual at step k, which is denoted by $\|r_k\|_2$, satisfies $\|r_k\|_2 \leq 10^{-4}\, \|r_0\|_2$.

The PCD preconditioner $\mathcal{P}_p$ for the linear system (15) is chosen as,

$$\mathcal{P}_p = \begin{bmatrix} A + J^f & B^T \\ & -S_p \end{bmatrix}, \tag{29}$$

where $S_p = L_p A_p^{-1} M_p$ is the approximatation of the Schur complement $B(A+J^f)^{-1}B^T$. Here, A_p and L_p are the convection-diffusion operator and Laplace operator in the finite dimensional solution space of the pressure with prescribed boundary conditions, M_p is the pressure mass matrix. For this PCD preconditioner, both $A + J^f$ and S_p are approximated by the algebraic multigrid (AMG) method that is implemented in IFISS.

We set the Reynolds number $Re = 2000$ as discussed in the previous section. We report the performance of both preconditioners in Table 1-2. Here the column “precon.” represents the time to compute the MSSS preconditioner or the time for the setup of the AMG method, and the column “IDR(4)” denotes the time of the preconditioned `IDR(4)` solver to compute the solution up to prescribed accuracy. Both time are measured in seconds.

Table 1 shows the time to compute the MSSS preconditioner scales linearly with the problem size. The number of iterations remains virtually constant with the refinement of the mesh. The time of the preconditioned `IDR(4)` solver also scales linearly with the problem size. For PCD preconditioner, the preconditioned `IDR(4)` solver fails to converge to the solution of prescribed accuracy within 400 iterations for relatively bigger mesh size. For a very fine mesh, the preconditioned `IDR(4)` solver computes the solution up to the prescribed accuracy within 300 iterations. This is because the entries of perturbation by the matrix J^f in (29), which is introduced by the nonlinear right-hand side, is of $\mathcal{O}(h^2)$. As $h \to 0$, this perturbation by the Jacobian matrix becomes smaller compared with A in the $(1, 1)$ block of (29). Therefore, S_p approximates the perturbed Schur complement well.

The computational results by the MSSS preconditioner in Table 2 show that the time for the setup of AMG does not scale linearly with the problem size. The reason may be that the AMG implemented in IFISS is not optimized, or the AMG algorithm in IFISS does not have linear computational complexity.

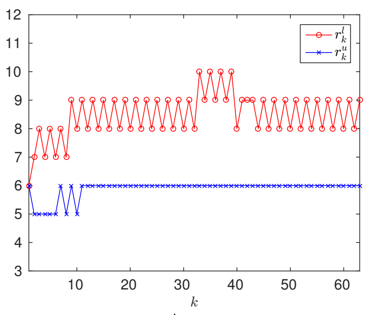
Table 1: Computational results of the MSSS preconditioner for $Re = 2000$

grid	problem size	# iter.	precon. (sec.)	IDR(4) (sec.)	total (sec.)
64×64	$1.25e + 04$	2	4.36	0.64	5.00
128×128	$4.97e + 04$	3	18.43	2.53	20.96
256×256	$1.98e + 05$	5	65.09	9.25	74.34
512×512	$7.88e + 05$	3	272.63	24.62	297.25

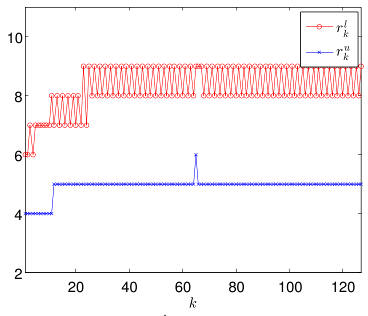
Table 2: Computational results of the PCD preconditioner for $Re = 2000$

grid	problem size	# iter.	precon. (sec.)	IDR(4) (sec.)	total (sec.)
64×64	$1.25e + 04$	400	8.56	no convergence	-
128×128	$4.97e + 04$	400	70.74	no convergence	-
256×256	$1.98e + 05$	266	237.68	42.93	280.61
512×512	$7.88e + 05$	203	1386.98	101.72	1488.70

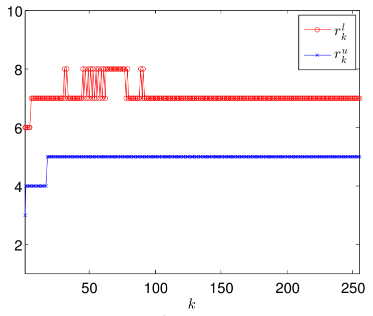
To compute the MSSS preconditioner, we set $\tau = 10^{-5}$ for the 512×512 grid and 10^{-4} for the rest grids. Here τ is the upper bound of the discarded singular values for the model order reduction that is performed to compute the approximate factorization. The adaptive rank for the low-rank approximation of the upper-triangular part dennoted by r_k^u and for approximation of the lower-triangular part denoted by r_k^l for this set up are plotted in Figure 3. For details of this approximate factorization, we refer to [42].



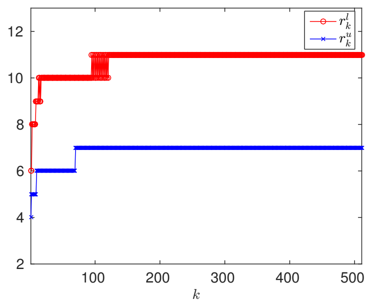
(a) $\tau = 10^{-4}$, 64×64 grid



(b) $\tau = 10^{-4}$, 128×128 grid



(c) $\tau = 10^{-4}$, 256×256 grid



(d) $\tau = 10^{-5}$, 512×512 grid

Figure 3: Adaptive semiseparable order to compute the MSSS preconditioner for $Re = 2000$

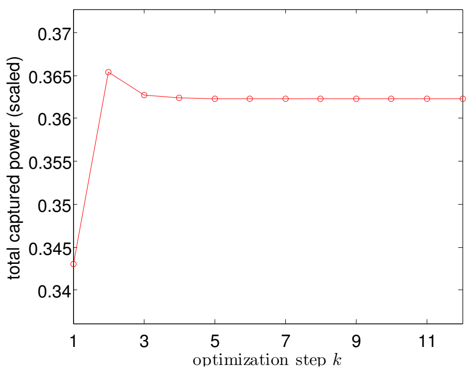
The adaptive rank in Figure 3 is bounded by a small constant around 10 for all the computations of MSSS preconditioners. This in turn gives the linear computational complexity of the MSSS preconditioning techniques, which is illustrated by Table 1-2.

5.2. Optimization Algorithm

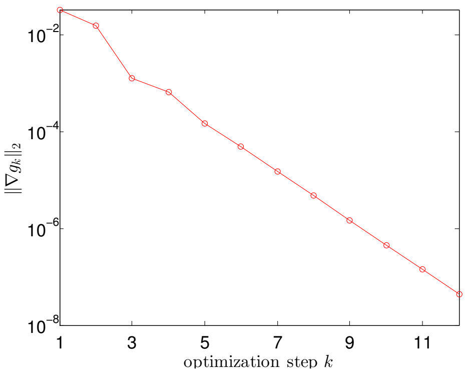
We test Algorithm 1 for the optimization problem (9) by using a 256×256 grid. At each outer iteration, we need to solve a series of linear equations of size 197634×197634 to compute the solution of the nonlinear flow equation (11) by the Newton's method. We also need to solve two adjoint equations (19) and (24) of the same size to compute the gradient and the Hessian matrix.

We use the wind farm configuration as introduced in Section 2.1. With this problem settings, the rightmost turbine just operates in the maximum power tracking mode, i.e., $C_{T_3} = 1$. Therefore, we just need to optimize C_{T_1} and C_{T_2} for the first two turbines. Without optimization, the turbines are operated in the mode that corresponds to the maximal power extraction for a single turbine, i.e., $C_T = 1$. We start with $C_{T_1}^{(0)} = C_{T_2}^{(0)} = 1$ as an initial guess for this optimization problem. Then the (scaled) total extracted power by the wind farm at each optimization step is given in Figure 4(a), while the 2-norm of the gradient at each optimization step is shown by Figure 4(b). Results in Figure 4(a) show that the total power is increased by around 5.5% when applying optimal control scheme.

Note that there is an overshoot after the second optimization step as illustrated by Figure 4(a), which makes the optimization problem to converge to a local optimum. This is primarily because of the non-convexity and highly nonlinearity of this optimization problem. The convexity of this optimization problem is still an open problem. Another reason that contributes to this behavior is the sensitivity of this optimization problem. Here we measure the sensitivity of the change of the velocity in the flow direction with respect to C_{T_j} by $\frac{\partial u_x}{\partial C_{T_j}}$. We plot the magnitude of $\frac{\partial u_x}{\partial C_{T_1}}$ and $\frac{\partial u_x}{\partial C_{T_2}}$ at the local optimum of C_{T_1} and C_{T_2} in Figure 5.

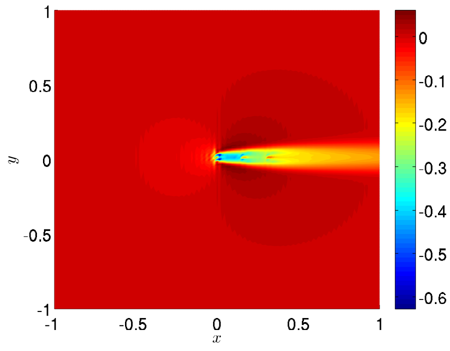


(a) Total power

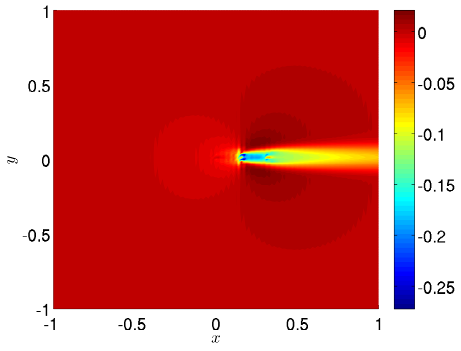


(b) 2-norm of the gradient

Figure 4: Total power and 2-norm of the gradient



(a) $\frac{\partial u_x}{\partial C_{T_1}}$



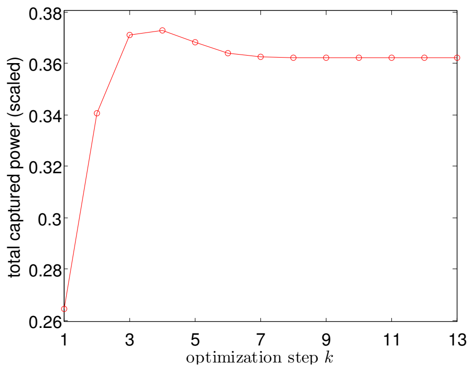
(b) $\frac{\partial u_x}{\partial C_{T_2}}$

Figure 5: $\frac{\partial u_x}{\partial C_{T_1}}$ and $\frac{\partial u_x}{\partial C_{T_2}}$ in magnitude

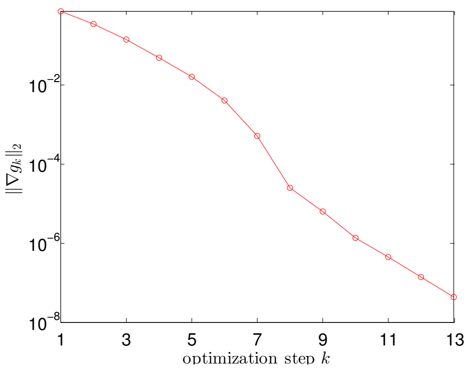
Figure 5 show that there is a big gradient in the vicinity of the places where the turbines are located. This in turn tells us that the velocity in the vicinity of the turbines is very sensitive to the changes of C_{T_j}. This makes this optimization problem very sensitive.

Another reason may be the robustness and the efficiency of the optimization method around the optimum. Since we focus on preconditioning, we leave this for the discussions and recommendations in the next chapter.

We also solve the optimization problem with an initial guess $C_{T_1}^{(0)} = C_{T_2}^{(0)} = 0$, although this corresponds to an impractical operation status. The scaled total power and the gradient at each optimization step are given in Figure 6.



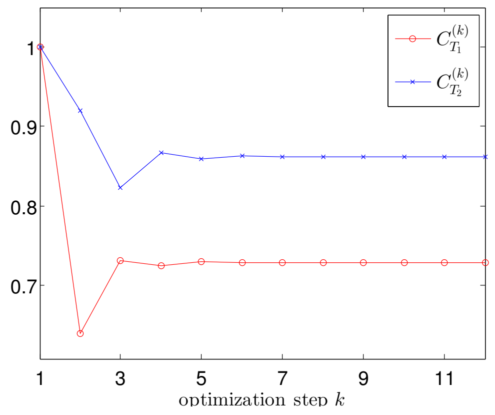
(a) Total power



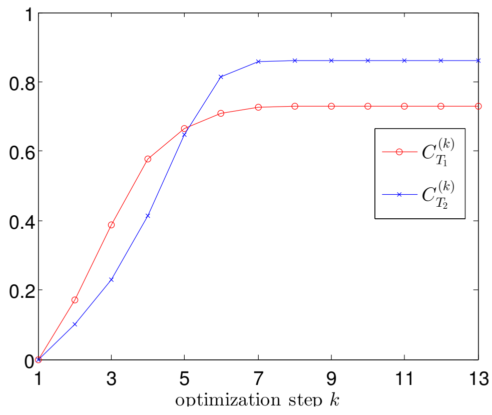
(b) 2-norm of the gradient

Figure 6: Total power and 2-norm of the gradient

For those two cases with different initial guesses, the corresponding optimized variables C_{T_1} and C_{T_2} at each optimization step are given in Figure 7.



(a) $C_{T_1}^{(0)} = C_{T_2}^{(0)} = 1$



(b) $C_{T_1}^{(0)} = C_{T_2}^{(0)} = 0$

Figure 7: Optimized C_{T_1}, C_{T_2} with different initial guesses

Figure 7(a) and 7(b) show that with different initial guesses, the optimized variables $(C_{T_1},\ C_{T_2})$ converge to the same point $(0.729,\ 0.862)$, which corresponds to a local optimum of the optimization problem.

6. Conclusions

In this manuscript, we studied preconditioning optimal in-domain Navier-Stokes control problem using multilevel sequentially semiseparable (MSSS) matrix computations. For the in-domain Navier-Stokes control problem, the control input only acts on part of the domain, which results in a problem even more difficult to solve. The optimization problem was solved by a reduced Newton's method while the most time consuming part for this problem was solving a nonlinear flow equation and computations of the gradient and the Hessian matrix. We showed that the gradient and the Hessian matrix can be computed by solving an adjoint equation such that the preconditioner for the solution of the flow equation can be reused. This in turn reduces the computational complexity dramatically. We evaluated the performance of the MSSS preconditioning technique by using IFISS. Numerical results show the superiority of the MSSS preconditioning technique to the standard preconditioning technique.

Since we focused on preconditioning optimal in-domain Navier-Stokes control problem, we used a laminar flow model in IFISS to study the performance of the MSSS preconditioning technique. The next step to extend this research shall focus on applying the turbulent flow model for the real-world wind farm control applications.

References

[1] D. B. Leineweber, I. Bauer, H. G. Bock, J. P. Schlöder, An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization. Part I: theoretical aspects, Computers & Chemical Engineering 27 (2) (2003) 157 – 166.

[2] A. Borzì, V. Schulz, Multigrid methods for PDE optimization, SIAM Review 51 (2) (2009) 361–395.

[3] L. T. Biegler, O. Ghattas, M. Heinkenschloss, B. van Bloemen Waanders, Large-scale PDE-constrained optimization, Vol. 30, Springer Science & Business Media, 2012.

[4] P. E. Gill, W. Murray, D. B. Ponceleón, M. A. Saunders, Preconditioners for indefinite systems arising in optimization, SIAM Journal on Matrix Analysis and Applications 13 (1) (1992) 292–311.

[5] J. Schöberl, W. Zulehner, Symmetric indefinite preconditioners for saddle point problems with applications to PDE-constrained optimization problems, SIAM Journal on Matrix Analysis and Applications 29 (3) (2007) 752–773.

[6] H. Badreddine, S. Vandewalle, J. Meyers, Sequential quadratic programming (SQP) for optimal control in direct numerical simulation of turbulent flow, Journal of Computational Physics 256 (2014) 1–16.

[7] M. Hinze, K. Kunisch, Second order methods for optimal control of time-dependent fluid flow, SIAM Journal on Control and Optimization 40 (3) (2001) 925–946.

[8] M. Laumen, Newton's method for a class of optimal shape design problems, SIAM Journal on Optimization 10 (2) (2000) 503–533.

[9] M. Berggren, Numerical solution of a flow-control problem: Vorticity reduction by dynamic boundary action, SIAM Journal on Scientific Computing 19 (3) (1998) 829–860.

[10] A. V. Fursikov, M. D. Gunzburger, L. S. Hou, Boundary value problems and optimal boundary control for the Navier-Stokes system: the two-dimensional case, SIAM Journal on Control and Optimization 36 (3) (1998) 852–894.

[11] J. P. Goit, J. Meyers, Optimal control of energy extraction in wind-farm boundary layers, Journal of Fluid Mechanics 768 (2015) 5–50.

[12] S. S. Adavani, G. Biros, Multigrid algorithms for inverse problems with linear parabolic PDE constraints, SIAM Journal on Scientific Computing 31 (1) (2008) 369–397.

[13] J. W. Pearson, M. Stoll, Fast iterative solution of reaction-diffusion control problems arising from chemical processes, SIAM Journal on Scientific Computing 35 (5) (2013) B987–B1009.

[14] A. Potschka, M. S. Mommer, J. P. Schlöder, H. G. Bock, Newton-picard-based preconditioning for linear-quadratic optimization problems with time-periodic parabolic pde constraints, SIAM Journal on Scientific Computing 34 (2) (2012) A1214–A1239.

[15] T. Rees, A. J. Wathen, Preconditioning iterative methods for the optimal control of the stokes equations, SIAM Journal on Scientific Computing 33 (5) (2011) 2903–2926.

[16] M. Stoll, T. Breiten, A low-rank in time approach to PDE-constrained optimization, SIAM Journal on Scientific Computing 37 (1) (2015) B1–B29.

[17] J. W. Pearson, Preconditioned iterative methods for Navier-Stokes control problems, Journal of Computational Physics 292 (0) (2015) 194 – 207.

[18] E. Bänsch, P. Benner, J. Saak, H. K. Weichelt, Riccati-based boundary feedback stabilization of incompressible Navier-Stokes flows, SIAM Journal on Scientific Computing 37 (2) (2015) A832–A858.

[19] J. Meyers, C. Meneveau, Optimal turbine spacing in fully developed wind farm boundary layers, Wind Energy 15 (2) (2012) 305–317.

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

[21] J. Annoni, P. Seiler, K. Johnson, P. Fleming, P. Gebraad, Evaluating wake models for wind farm control, in: Proceedings of American Control Conference, 2014, pp. 2517–2523.

[22] J. R. Marden, S. D. Ruben, L. Y. Pao, A model-free approach to wind farm control using game theoretic methods, IEEE Transactions on Control Systems Technology 21 (4) (2013) 1207–1214.

[23] A. Crespo, F. J. R. Hernández, S. Frandsen, Survey of modelling methods for wind turbine wakes and wind farms, Wind energy 2 (1) (1999) 1–24.

[24] B. Sanderse, S. P. van der Pijl, B. Koren, Review of computational fluid dynamics for wind turbine wake aerodynamics, Wind Energy 14 (7) (2011) 799–819.

[25] P. M. O. Gebraad, Data-driven wind plant control, Ph.D. thesis, Delft University of Technology (2014).

[26] O. Rathmann, S. T. Frandsen, R. J. Barthelmie, Wake modelling for intermediate and large wind farms, in: 2007 European Wind Energy Conference and Exhibition.

[27] P. Torres, J.-W. Van Wingerden, M. Verhaegen, Modeling of the flow in wind farms for total power optimization, in: Proceedings of the 9th IEEE International Conference on Control and Automation, 2011, pp. 963–968.

[28] M. Abkar, F. Porté-Agel, The effect of free-atmosphere stratification on boundary-layer flow and power output from very large wind farms, Energies 6 (5) (2013) 2338–2361.

[29] F. M. White, I. Corfield, Viscous fluid flow, Vol. 3, McGraw-Hill, New York, 2006.

[30] K. Avila, D. Moxey, A. de Lozar, M. Avila, D. Barkley, B. Hof, The onset of turbulence in pipe flow, Science 333 (6039) (2011) 192–196.

[31] R. Mikkelsen, Actuator disc methods applied to wind turbines, Ph.D. thesis, Technical University of Denmark (2003).

[32] N. Troldborg, Actuator line modeling of wind turbine wakes, Ph.D. thesis, Technical University of Denmar (2008).

[33] J. Nocedal, S. Wright, Numerical optimization, Springer Science & Business Media, New York, 2006.

[34] C. Li, C. Vuik, Eigenvalue analysis of the SIMPLE preconditioning for incompressible flow, Numerical Linear Algebra with Applications 11 (5-6) (2004) 511–523.

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

[36] D. Kay, D. Loghin, A. Wathen, A preconditioner for the steady-state Navier-Stokes equations, SIAM Journal on Scientific Computing 24 (1) (2002) 237–256.

[37] K. Abe, G. L. Sleijpen, Hybrid Bi-CG methods with a Bi-CG formulation closer to the IDR approach, Applied Mathematics and Computation 218 (22) (2012) 10889 – 10899.

[38] L. Du, T. Sogabe, B. Yu, Y. Yamamoto, S. Zhang, A block IDR(s) method for nonsymmetric linear systems with multiple right-hand sides, Journal of Computational and Applied Mathematics 235 (14) (2011) 4095 – 4106.

[39] Y. Qiu, M. B. van Gijzen, J.-W. van Wingerden, M. Verhaegen, C. Vuik, Efficient preconditioners for PDE-constrained optimization problems with a multilevel sequentially semiseparable matrix structure, Electronic Transactions on Numerical Analysis 44 (2015) 367–400.

[40] Y. Qiu, M. B. van Gijzen, J.-W. van Wingerden, M. Verhaegen, C. Vuik, Evaluation of multilevel sequentially semiseparable preconditioners on CFD benchmark problems using incompressible flow and iterative solver software, Mathematical Methods in the Applied Sciences 38. doi:10.1002/mma.3416.

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

[42] Y. Qiu, M. B. van Gijzen, J.-W. van Wingerden, M. Verhaegen, C. Vuik, Convergence analysis of the multilevel sequentially semiseparable preconditioners, Tech. Rep. 15-01, Delft Institution of Applied Mathematics, Delft University of Technology, available at http://ta.twi.tudelft.nl/nw/users/yueqiu/publications.html (2015).

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

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