SIAM J. SCI. COMPUT.
Vol. 39, No. 1, pp. A255–A280

LOW-RANK SOLUTION TO AN OPTIMIZATION PROBLEM CONSTRAINED BY THE NAVIER–STOKES EQUATIONS*

SERGEY DOLGOV† AND MARTIN STOLL‡

Abstract. The numerical solution of PDE-constrained optimization problems subject to the nonstationary Navier–Stokes equation is a challenging task. While space-time approaches often show favorable convergence properties, they often suffer from storage problems. Here we propose to approximate the solution to the optimization problem in a low-rank form, which is similar to the model order reduction (MOR) approach. However, in contrast to classical MOR schemes we do not compress the full solution at the end of the algorithm but start our algorithm with low-rank data and maintain this form throughout the iteration. Numerical experiments indicate that this approach reduces the computational costs by two orders of magnitude.

Key words. PDE-constrained optimization, low-rank methods, iterative solvers, model reduction, preconditioning, alternating solvers

AMS subject classifications. Primary, 65F10, 65N22, 65F50; Secondary, 76D07

DOI. 10.1137/15M1040414

1. Introduction.

Optimization subject to constraints given by PDEs is an active field of research [57, 28]. Much progress has been made in recent years concerning the analysis of problems and the development of efficient numerical schemes to solve the linear and nonlinear optimization problems. In almost all scenarios one arrives at the solution of a large-scale linear system that either represents the first order (KKT) conditions [43] or is part of some nonlinear scheme such as an SQP or interior point approach [25, 58].

The development of iterative solvers and especially preconditioners for these linear systems, which often have a saddle-point form, has been a key research area in numerical linear algebra [15, 41, 22]. For parabolic problems, space-time methods have shown great promise [50, 54] regarding robustness with respect to dependence on both mesh- and regularization parameters. Multigrid methods [10] are also used for parabolic problems, as are methods using stationary iterations [29].

Our approach in this paper follows recent work presented in [53], where the space-time system is solved using a low-rank decomposition. This schemes replaces the space-time solution, which can be written as a matrix, by an approximation that only needs minimal information from the space and time domains. We will make this more precise in section 2.1. The goal is to reduce the storage amount to a small multiple of that of the stationary problem. The work in [53] only considered linear problems, and here we present how this approach can be carried over to the case when we consider the optimization subject to the Navier–Stokes equations. The control of the Navier–Stokes equations has been an active research topic in recent years, and we refer the

*Submitted to the journal's Methods and Algorithms for Scientific Computing section September 18, 2015; accepted for publication (in revised form) October 11, 2016; published electronically February 8, 2017.

http://www.siam.org/journals/sisc/39-1/M104041.html

Funding: The first author acknowledges the support of the EPSRC Postdoctoral Fellowship EP/M019004/1.

†Department of Mathematical Sciences, University of Bath, Claverton Down, BA2 7AY Bath, United Kingdom (S.Dolgov@bath.ac.uk).

‡Numerical Linear Algebra for Dynamical Systems, Max Planck Institute for Dynamics of Complex Technical Systems, 39106 Magdeburg, Germany (stollm@mpi-magdeburg.mpg.de).

reader to [23, 18] and the references therein.

Low-rank approximations have become state-of-the-art methods for data-sparse solution of high-dimensional problems [34, 17, 19]. This approach is based on the idea of separation of variables and approximates a large multidimensional array (tensor) by a polylinear combination of smaller tensors. Some of the most powerful of such combinations are the tensor train (TT) [46] and hierarchical Tucker (HT) [21] formats. A tensor can be associated with a discretized operator or solution of a high-dimensional PDE. To solve an equation, one needs an iterative algorithm, since elements of an initial (large) tensor are never accessed directly. One can adapt classical algorithms, such as GMRES [3, 27], or develop tailored methods for low-rank formats, such as the alternating least squares [26].

TT and HT decompositions are based on a recurrent application of the matrix low-rank factorization, for example, the singular value decomposition (SVD). Therefore, their storage efficiency depends on the ranks of the corresponding matricizations of a tensor. If all ranks are bounded by a moderate value for the given data, the storage needed for a low-rank format is logarithmic compared to the cardinality of the initial tensor. In an ultimate scenario, one can reshape any data to a tensor, the dimensions of which are prescribed by the smallest (prime) factors of the number of elements. The TT decomposition applied to such a tensor is called the quantized tensor train (QTT) format [33, 45], and it has demonstrated impressive compression properties for smooth solutions of PDEs, discretized on tensor product grids.

However, if a PDE is posed on a complicated spatial domain, an ultimate tensorization is inapplicable. Nonetheless, there can still be several independent variables, such as an aggregated spatial coordinate and time and auxiliary parameters. In this paper we consider separation of space and time variables only, without further tensorization. Hence, the proposed method can easily be used for unstructured grids in space. In this case, the solution is reshaped to a matrix, and the classical low-rank decomposition is sought. There exist many efficient algorithms for solution of matrix equations in the low-rank form, such as the ADI and Krylov methods for Sylvester equations [59, 4] (including high-dimensional generalizations [36, 40]). However, they often require commutativity of (at least, dominant) low-rank factors of the operator. Alternating low-rank tensor algorithms can be more efficient for problems such as the Navier–Stokes equations, where the operator has a complicated structure.

Development of low-rank methods for nonlinear problems has been performed in different communities in different ways. On one hand, the proper orthogonalized decomposition (POD) is a mature technique in MOR, and it has been used intensively for reducing the Navier–Stokes equations; see, e.g., [11, 2, 42]. However, the POD requires solving the full problem first, which might be extremely computationally demanding. On the other hand, tensor methods compute directly the low-rank factors, but they are applied to only a few nonlinear problems. One can mention the Hartree–Fock equation [31, 51] and some plasma models [13, 35]. More developed are methods for Ricatti equations [6], but they rely explicitly on the form of the operator.

In this paper we generalize the alternating least squares algorithm to the saddle-point structure of the optimality system, arising from the Lagrangian optimization, and adapt it particularly to the Navier–Stokes equations as constraints. We compare it with the traditional space-time optimization with the state-of-the-art preconditioners [50] and show that the new algorithm provides a significant reduction of computational time and storage.

Our paper is structured as follows. We first introduce the problem formulation for both the Navier–Stokes forward problem and the corresponding optimization problem. In section 2.1 we discuss the forward formulation and introduce an appropriate low-

rank formulation. This is followed by section 2.2, where a low-rank formulation for the optimization problem is developed. We also show that this can be used with various time-discretization schemes. In section 3.1 we propose an alternating linear scheme (ALS) for the forward simulation which in section 3.2 is followed by a detailed discussion of such an ALS method for the optimality system that sits at the heart of the outer nonlinear Picard iteration. The particular case of carefully handling the pressure degrees of freedom needed for the ALS method is discussed in section 3.3. We propose efficient solvers for the linear systems in saddle-point form in section 3.4. A discussion about the existence of solutions for the optimization problem is presented in section 4. Our numerical experiments shown in section 5 illustrate that our method performs very robustly. In particular we show that the storage amount needed for the low-rank scheme is typically a fraction of the storage requirement for the full problem. This is combined with a thorough parameter study, where all system parameters are varied over orders of magnitude with only very benign rank growth observed.

2. Problem formulation. We start our discussion by introducing the formulation of the Navier–Stokes equations that we are going to use throughout this paper:

$$\mathbf{y}_t - \nu\Delta\mathbf{y} + (\mathbf{y}\cdot\nabla)\,\mathbf{y} + \nabla p = \mathbf{u}, \tag{2.1}$$

$$\nabla\cdot\mathbf{y} = 0, \tag{2.2}$$

posed on domain $\Omega \in \mathbb{R}^{2,3}$ with appropriate boundary and initial conditions (see [15] and the references mentioned therein for more details). Often one is also interested in solving optimization problems where the Navier–Stokes equations appear as a constraint [8, 9, 23]. For this we consider the following objective function:

$$J(\mathbf{y},\mathbf{u}) = \frac{1}{2}\left\|\mathbf{y}-\mathbf{y}_d\right\|_{Q_0}^2 + \frac{\beta}{2}\left\|\mathbf{u}\right\|_{Q_c}^2, \tag{2.3}$$

where $Q_o = \Omega_o \times [0,T]$ and $Q_c = \Omega_c \times [0,T]$ are space-time cylinders. Here $\Omega_o \subseteq \Omega$ is the observation domain and $\Omega_c \subseteq \Omega$ the control domain. For now we assume that both are equal to Ω. The function $\mathbf{y}_d$ is the desired state. For this case the right-hand side of (2.1) represents the control $\mathbf{u}$, which is computed in such a way that the solution of the Navier–Stokes equation is close to the desired state.

Additionally, we also consider an objective function including a vorticity term [23],

$$J_2(\mathbf{y},\mathbf{u}) = \frac{\alpha_1}{2}\left\|\mathbf{y}-\mathbf{y}_d\right\|_{Q_0}^2 + \frac{\alpha_2}{2}\left\|\mathrm{curl}(\mathbf{y})\right\|_{Q_c}^2 + \frac{\beta}{2}\left\|\mathbf{u}\right\|_{Q_c}^2. \tag{2.4}$$

Many researchers have studied the numerical solution of the Navier–Stokes equations and how they can be used as constraints in an optimization problem. Our goal here is to discuss the possibility of extending the framework recently introduced for PDE optimization problems [53] with linear constraints. This framework utilizes a low-rank structure of the solution and hence enables efficient solvers for the optimization problem.

Before discussing the Navier–Stokes case, we briefly want to introduce the idea using the Stokes equations as an example. For this we consider the Stokes equations

$$\mathbf{y}_t - \nu\Delta\mathbf{y} + \bigtriangledown p = \mathbf{u}, \tag{2.5}$$

$$\bigtriangledown\cdot\mathbf{y} = 0 \tag{2.6}$$

equipped with appropriate initial and boundary conditions. Employing a finite element discretization in space, an implicit Euler discretization for the temporal discretization of the PDE, and a trapezoidal rule for numerical integration of the objective function leads to a discretized optimization problem [54, 24]. The first order

conditions using a Lagrangian with Lagrange multiplier $\boldsymbol{\lambda}$ then lead to the following system:

$$\begin{bmatrix} \mathcal{M}_1 & 0 & \mathcal{K}^T \\ 0 & \mathbf{M}_2 & -\mathcal{M}_3^T \\ \mathcal{K} & -\mathcal{M}_3 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{y}_h \\ p_h \\ \mathbf{u}_h \\ \boldsymbol{\lambda}_h \\ \xi_h \end{bmatrix} = \begin{bmatrix} \mathcal{M}_1 \mathbf{y}_d \\ 0 \\ d \end{bmatrix}, \tag{2.7}$$

which we want to write in Kronecker notation utilizing the fact that the Stokes equations are discretized as

$$\mathcal{K} \begin{bmatrix} \mathbf{y}_h \\ p_h \end{bmatrix} - \mathcal{M}_3 \mathbf{u}_h = d, \tag{2.8}$$

where

$$\mathcal{K} = (I_n \otimes \mathcal{L} + C \otimes \mathcal{M}), \quad \mathcal{M}_3 = I_n \otimes \begin{bmatrix} M \\ 0 \end{bmatrix}, \quad d = e_1 \otimes \begin{bmatrix} \mathbf{M}\mathbf{y}_0 \\ 0 \end{bmatrix} + f,$$

where $e_1 \in \mathbb{R}^n$ is the first unit vector (this term accounts for the initial state), and f agglomerates the boundary conditions (this will be written in detail later). The number n denotes the number of time steps, and $C \in \mathbb{R}^{n,n}$ is given by

$$C = \frac{1}{\tau} \begin{bmatrix} 1 & & & \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{bmatrix},$$

where τ is the time step. As for the spatial matrices,

$$\mathcal{L} = \begin{bmatrix} \mathbf{L} & B^T \\ B & 0 \end{bmatrix}$$

represents an instance of a time-dependent Stokes problem with B the discrete divergence and $\mathbf{L}$ the Laplacian (including viscosity ν); $\mathbf{M}$ is the mass matrix, associated with the velocity space; and $\mathcal{M} = \left[\begin{smallmatrix} \mathbf{M} & 0 \\ 0 & 0 \end{smallmatrix}\right]$ is the mass matrix for the velocity-pressure space. The matrices $\mathcal{M}_1 = \Theta \otimes \mathcal{M}$ and $\mathbf{M}_2 = \beta\Theta \otimes \mathbf{M}$ with $\Theta = \tau \cdot \mathrm{diag}(\frac{1}{2}, 1, \ldots, 1, \frac{1}{2})$ denote the mass matrices coming from the discretization of the functional (2.3).

The goal then is to use the fact that the right-hand side of the optimality system can be written in low-rank form, and this can be carried trough an iterative solver like MINRES [48] without a substantial increase in the rank of the solution [53].

2.1. Low-rank approximation of the Navier–Stokes forward problem. The situation for the Navier–Stokes equations is more complex as the nonlinear convection term does not allow for such an easy description of the problem. Typically the Navier–Stokes equations are discretized in space followed by a discretization in time. One then has to solve a nonlinear problem at every time step, for which both Newton as well as Picard iterations have shown to be good candidates [15]. Here we focus on the Picard iteration and follow the description in [15, Chapter 8.2] which establishes that

$$\int \mathbf{y}_t \mathbf{v} - \int \nu \nabla \mathbf{y} : \nabla \mathbf{v} + c(\bar{\mathbf{y}}, \mathbf{y}, \mathbf{v}) - \int p(\nabla \cdot \mathbf{v}) = \int \mathbf{u}\mathbf{v} \qquad \forall \mathbf{v} \in H^1(\Omega), \tag{2.9}$$

$$\int q(\nabla \cdot \mathbf{y}) = 0 \qquad \forall q \in L_2(\Omega) \tag{2.10}$$

with the trilinear form

$$c(\bar{\mathbf{y}}, \mathbf{y}, \mathbf{v}) := \int (\bar{\mathbf{y}} \cdot \nabla \mathbf{y}) \cdot \mathbf{v},$$

where $\bar{\mathbf{y}}$ denotes the previous iterate of the nonlinear Picard solver. Note that this formulation is typically known as the Oseen equations and will be at the heart of this paper.

The basis for the low-rank solution in the Stokes case is based on the fact that the right-hand side of the linear system is of low rank or can be well approximated by a low-rank function. We start by considering the forward problem with the right-hand side $\mathbf{u}$. We now assume that $\mathbf{u}$ is either given or approximated by

$$\mathbf{u} = \sum_{i=1}^{r_{\mathbf{u}}} v_{i,\mathbf{u}}(t) w_{i,\mathbf{u}}(x),$$

which in discretized form is written as

$$\mathbf{u}_h = \sum_{i=1}^{r_{\mathbf{u}_h}} v_{i,\mathbf{u}_h} \otimes w_{i,\mathbf{u}_h}.$$

Using this for the first step of the Picard iteration, an implicit Euler discretization in time and finite elements in space, we obtain the following discretized system:

$$(C \otimes \mathcal{M} + I_n \otimes \mathcal{L})\, \mathbf{y}_h = \sum_{i=1}^{r_{\mathbf{u}_h}} v_{i,\mathbf{u}_h} \otimes w_{i,\mathbf{u}_h} + d. \tag{2.11}$$

In our case, the spatial discretization is based on the well-known *Taylor–Hood* $\mathbf{Q}_2/\mathbf{Q}_1$ finite elements [15], which results in a stable discretization of the Oseen equations. Note that our approach is also valid for other discretizations such as stabilized ones but would typically require adjustments in the construction of the preconditioners [15]. We now assume that at a step ℓ of our Picard iteration, the previous solution is given by

$$\bar{\mathbf{y}}_h = \sum_{i=1}^{r_{\bar{\mathbf{y}}_h}} v_{i,\bar{\mathbf{y}}_h} \otimes w_{i,\bar{\mathbf{y}}_h}, \tag{2.12}$$

and we want to compute the next (i.e., $(\ell+1)$th) Picard iteration. Notice that the trilinear form in (2.9) is linear in $\bar{\mathbf{y}}$ and hence preserves the low-rank form of $\bar{\mathbf{y}}_h$. Let us assume that finite elements $\{\boldsymbol{\phi}_1(x), \ldots, \boldsymbol{\phi}_m(x)\}$ are used for the discretization of the velocity in space. Then $\bar{\mathbf{y}}(t_l, x)$ is constructed from $\bar{\mathbf{y}}_h$ by interpolation:

$$\bar{\mathbf{y}}(t_l, x) = \sum_{k=1}^{m} \bar{\mathbf{y}}_{h,k}(t_l) \boldsymbol{\phi}_k(x) = \sum_{k=1}^{m} \sum_{i=1}^{r_{\bar{\mathbf{y}}_h}} v_{i,\bar{\mathbf{y}}_h,l} \otimes w_{i,\bar{\mathbf{y}}_h,k} \boldsymbol{\phi}_k(x),$$

where $l = 1, \ldots, n$ is the time step. Plugging this into $c(\bar{\mathbf{y}}, \mathbf{y}, \mathbf{v})$, we obtain

$$c(\bar{\mathbf{y}}(t_l), \boldsymbol{\phi}_{j'}, \boldsymbol{\phi}_j) = \sum_{i=1}^{r_{\bar{\mathbf{y}}_h}} v_{i,\bar{\mathbf{y}}_h,l} \otimes (\mathbf{N}_i)_{j,j'}, \quad j, j' = 1, \ldots, m, \tag{2.13}$$

where $\mathbf{N}_i \equiv \mathbf{N}(w_{i,\bar{\mathbf{y}}_h}) \in \mathbb{R}^{m \times m}$ is defined by its elements:

$$(\mathbf{N}_i)_{j,j'} = \int \boldsymbol{\phi}_j \left(\sum_{k=1}^{m} w_{i,\bar{\mathbf{y}}_h,k} \boldsymbol{\phi}_k \right) \nabla \boldsymbol{\phi}_{j'}. \tag{2.14}$$

Since $\boldsymbol{\phi}_k(x)$ are finitely supported, most of the triple products of $\boldsymbol{\phi}$ above are zeros, and $\mathbf{N}_i$ can be assembled in $\mathcal{O}(m)$ operations.

Now, for the fully discretized problem, we have

$$\left(C \otimes \mathcal{M} + I_n \otimes \mathcal{L} + \sum_{i=1}^{r_{\bar{\mathbf{y}}_h}} D_i \otimes \mathcal{N}_i\right) \mathbf{y}_h = \sum_{i=1}^{r_{\mathbf{u}_h}} v_{i,\mathbf{u}_h} \otimes w_{i,\mathbf{u}_h} + \bar{d}, \tag{2.15}$$

where $\mathcal{N}_i = \text{blkdiag}(\mathbf{N}_i, 0)$ and $D_i = \text{diag}(v_{i,\bar{\mathbf{y}}_h})$. Note that $\bar{d}$ consists of the contributions coming from the boundary conditions at the previous step due to the changing matrix $\mathbf{N}_i$. The Picard iteration is now continued until convergence. The main advantage of the nonlinear solver, i.e., Picard iteration in this case, as the outer iteration is that we can reduce the storage amount for the inner space-time problem. This is true if the ranks $r_{\mathbf{y}_h}$ are kept small and hence the amount of storage is kept small and only a few matrices $\mathbf{N}_i$ have to be assembled.

Although we employ the implicit Euler scheme in practical computations, our method can also be used with other temporal discretizations [15],

$$\frac{1}{\tau}\left(\mathbf{y}^{(l+1)} - \mathbf{y}^{(l)}\right) + (\bar{\mathbf{y}}^* \cdot \nabla)\,\mathbf{y}^{(l+\frac{1}{2})} - \nu\Delta\mathbf{y}^{(l+\frac{1}{2})} + \nabla p^{(l+\frac{1}{2})} = \mathbf{u}^{(l+\frac{1}{2})}, \tag{2.16}$$

$$\nabla \cdot \mathbf{y}^{(l+\frac{1}{2})} = 0, \tag{2.17}$$

where $\mathbf{y}^{(l+\frac{1}{2})} := \frac{1}{2}(\mathbf{y}^{(l+1)} + \mathbf{y}^{(l)})$ and similarly for $\mathbf{u}$ and p. The choice $\bar{\mathbf{y}}^* = \mathbf{y}^{(l+\frac{1}{2})}$ represents the Crank–Nicolson scheme and $\bar{\mathbf{y}}^* = \frac{3}{2}\mathbf{y}^{(l)} - \frac{1}{2}\mathbf{y}^{(l-1)}$ the Simo–Amero scheme (cf. [15] for more details and further references). Note that as we approximate the space-time solution in a low-rank form, we do not proceed sequentially in time and these schemes need to be rewritten for our purposes. Hence we consider an all-at-once semidiscretized system for which the state can then be written as

$$\mathbf{y} := \begin{bmatrix} \mathbf{y}^{(1)} \\ \mathbf{y}^{(2)} \\ \vdots \\ \mathbf{y}^{(n)} \end{bmatrix}. \tag{2.18}$$

For this we need the two matrices

$$C = \frac{1}{\tau}\begin{bmatrix} 1 & & & \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \end{bmatrix} \quad \text{and} \quad \tilde{C} = \frac{1}{2}\begin{bmatrix} 1 & & & \\ 1 & 1 & & \\ & \ddots & \ddots & \\ & & 1 & 1 \end{bmatrix}, \tag{2.19}$$

$$C\mathbf{y} - \nu\Delta\tilde{C}\mathbf{y} + \mathbf{N}(\bar{\mathbf{y}})\tilde{C}\mathbf{y} + \nabla\tilde{C}p = \tilde{C}\mathbf{u}, \tag{2.20}$$

$$\nabla \cdot \tilde{C}\mathbf{y} = 0, \tag{2.21}$$

where for the Crank–Nicolson scheme the semidiscretized part is given by

$$\mathbf{N}(\bar{\mathbf{y}})\tilde{C}\mathbf{y} = \left(\hat{C}\bar{\mathbf{y}}\right)^T \text{blkdiag}(\cdot\nabla, \ldots, \cdot\nabla)\tilde{C}\mathbf{y}, \tag{2.22}$$

where $\hat{C} = \tilde{C}$ represents a Crank–Nicolson scheme or the Simo–Amero scheme via

$$\hat{C} = \frac{1}{2}\begin{bmatrix} 1 & & & & \\ 0 & 1 & & & \\ -1 & 3 & 0 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 3 & 0 \end{bmatrix}.$$

This is then followed by a spatial discretization where the discretization of most terms in (2.20) is straightforward and we focus on the term $\mathbf{N}(\bar{\mathbf{y}})\tilde{C}\mathbf{y}$, which using (2.14) is discretized as

$$\sum_{i=1}^{r_{\bar{\mathbf{y}}_h}} \operatorname{diag}(\hat{C}v_{i,\bar{\mathbf{y}}_h})\tilde{C} \otimes \mathbf{N}(w_{i,\bar{\mathbf{y}}_h}) = \sum_{i=1}^{r_{\bar{\mathbf{y}}_h}} \tilde{D}_i \otimes \mathbf{N}(w_{i,\bar{\mathbf{y}}_h}),$$

where $\tilde{D}_i = \operatorname{diag}(\hat{C}v_{i,\bar{\mathbf{y}}_h})\tilde{C}$. This leaves us with the overall space-time discretization

$$\left(C \otimes \mathcal{M} + \tilde{C} \otimes \mathcal{L} + \sum_{i=1}^{r_{\bar{\mathbf{y}}_h}} \tilde{D}_i \otimes \mathcal{N}_i\right)\mathbf{y}_h = \sum_{i=1}^{r_{\mathbf{u}_h}} v_{i,\mathbf{u}_h} \otimes w_{i,\mathbf{u}_h} + \bar{d}. \tag{2.23}$$

2.2. Low-rank approximation of the optimization problem. We now consider the case when the Navier–Stokes equations represent a constraint for a misfit functional such as the one given in (2.3). There are now two approaches that one can take to solve the optimization. The first discretizes the objective function first and then optimizes the discrete problem, while the second approach first derives the optimality system and then discretizes the resulting system. As we are only concerned with the efficient solution of a discretized problem, we believe that our approach can be used in both cases, but we focus on the optimize-then-discretize case. This means that we first build an infinite-dimensional Lagrangian and then consider its variation with respect to state, pressure, control, and two Lagrange multipliers that can be identified as the adjoint state and adjoint pressure [57, 28, 23]. Here we simply state the optimality system as derived in [23, Example 3.1],

$$\begin{aligned} \mathbf{y}_t - \Delta\mathbf{y} + (\mathbf{y}\cdot\nabla)\,\mathbf{y} + \bigtriangledown p &= \mathbf{u} && \text{on } [0,T]\times\Omega,\\ \nabla\cdot\mathbf{y} &= 0 && \text{on } [0,T]\times\Omega,\\ \mathbf{y}(0,.) &= \mathbf{y}_0 && \text{on } \Omega,\\ \mathbf{y} &= \mathbf{y}_\Gamma && \text{on } \Gamma, \end{aligned} \tag{2.24}$$

$$\begin{aligned} -\boldsymbol{\lambda}_t - \Delta\boldsymbol{\lambda} - (\mathbf{y}\cdot\nabla)\,\boldsymbol{\lambda} + (\nabla\mathbf{y})^T\boldsymbol{\lambda} + \bigtriangledown\xi &= -(\mathbf{y}-\mathbf{y}_d) && \text{on } [0,T]\times\Omega,\\ \nabla\cdot\boldsymbol{\lambda} &= 0 && \text{on } [0,T]\times\Omega,\\ \boldsymbol{\lambda}(T,.) &= -(\mathbf{y}(T)-\mathbf{y}_d(T)) && \text{on } \Omega,\\ \boldsymbol{\lambda} &= 0 && \text{on } \Gamma, \end{aligned} \tag{2.25}$$

$$\beta\mathbf{u} + \boldsymbol{\lambda} = 0 \qquad \text{on } [0,T]\times\Omega. \tag{2.26}$$

Now it is easily seen that this is a nonlinear problem due to the nonlinearity coming from the Navier–Stokes equation. Additional nonlinearities could come into this

equation if more complicated objective functions were considered. Note that the questions of existence and uniqueness are answered in [23]. Once again this equation has to be treated using a nonlinear solver, and we again propose the use of an Oseen (Picard-type) iteration [49] to give

$$\begin{aligned} \mathbf{y}_t - \Delta \mathbf{y} + (\bar{\mathbf{y}} \cdot \nabla)\, \mathbf{y} + \bigtriangledown p &= \mathbf{u}, \\ \nabla \cdot \mathbf{y} &= 0, \end{aligned}$$

$$\begin{aligned} -\boldsymbol{\lambda}_t - \Delta \boldsymbol{\lambda} - (\bar{\mathbf{y}} \cdot \nabla)\, \boldsymbol{\lambda} + (\nabla \bar{\mathbf{y}})^T \boldsymbol{\lambda} + \bigtriangledown \xi &= -(\mathbf{y} - \mathbf{y}_d), \\ \nabla \cdot \boldsymbol{\lambda} &= 0, \end{aligned}$$

$$\beta \mathbf{u} + \boldsymbol{\lambda} = 0,$$

where for brevity we omitted initial/final and boundary conditions. After that, we update $\bar{\mathbf{y}} = \mathbf{y}$ and proceed with the next iteration. We are now proposing the same solution as in the forward simulation. Additionally, we assume that all quantities are discretized in time and space. This means at each step of the algorithm we assume the states, adjoint states, and control are given by

$$\mathbf{y}_h = \sum_{i=1}^{r_{\mathbf{y}_h}} v_{i,\mathbf{y}_h} \otimes w_{i,\mathbf{y}_h}, \quad p_h = \sum_{i=1}^{r_{\boldsymbol{\lambda}_h}} v_{i,p_h} \otimes w_{i,p_h}, \tag{2.27}$$

$$\boldsymbol{\lambda}_h = \sum_{i=1}^{r_{\boldsymbol{\lambda}_h}} v_{i,\boldsymbol{\lambda}_h} \otimes w_{i,\boldsymbol{\lambda}_h}, \quad \xi_h = \sum_{i=1}^{r_{\xi_h}} v_{i,\xi_h} \otimes w_{i,\xi_h}, \tag{2.28}$$

$$\mathbf{u}_h = \sum_{i=1}^{r_{\mathbf{u}_h}} v_{i,\mathbf{u}_h} \otimes w_{i,\mathbf{u}_h}. \tag{2.29}$$

The Oseen equation that we have to solve is then of the following form:

$$\begin{bmatrix} \Theta \otimes \mathcal{M} & 0 & \mathcal{K}^* \\ 0 & \Theta \otimes \beta \mathbf{M} & -\mathcal{M}_3^\top \\ \mathcal{K} & -\mathcal{M}_3 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{y}_h \\ p_h \\ \mathbf{u}_h \\ \boldsymbol{\lambda}_h \\ \xi_h \end{bmatrix} = b, \tag{2.30}$$

where b represents the right-hand side and $\mathcal{K}$ describes the forward operator for the space-time Navier–Stokes equations,

$$\mathcal{K} = C \otimes \mathcal{M} + I_n \otimes \mathcal{L} + \sum_{i=1}^{r_{\bar{\mathbf{y}}_h}} D_i \otimes \mathcal{N}_i. \tag{2.31}$$

The adjoint PDE represented by $\mathcal{K}^*$ contains more terms than the forward equation due to the terms $(\mathbf{y} \cdot \nabla)\, \boldsymbol{\lambda} + (\nabla \mathbf{y})^T \boldsymbol{\lambda}$. As in the forward problem, we assume (2.12), so that $(\bar{\mathbf{y}} \cdot \nabla)\, \boldsymbol{\lambda}$ is discretized as $\sum_{i=1}^{r_{\bar{\mathbf{y}}_h}} D_i \otimes \mathcal{N}_i$. The term $(\nabla \bar{y})^T \boldsymbol{\lambda}$ now becomes $\sum_{i=1}^{r_{\bar{y}}} D_i \otimes \mathbf{H}_i(\bar{\mathbf{y}}_h)$ with $\mathbf{H}_i(\bar{\mathbf{y}}_h)$ being a matrix of entries $\int \boldsymbol{\phi}_j \cdot \nabla \left(\sum_{k=1}^{m} w_{i,\bar{\mathbf{y}}_h,k} \boldsymbol{\phi}_k\right) \cdot \boldsymbol{\phi}_{j'}$ for $j, j' = 1, \ldots, m$. The adjoint matrix is then given by

$$\mathcal{K}^* = C^\top \otimes \mathcal{M} + I_n \otimes \mathcal{L} - \sum_{i=1}^{r_{\bar{\mathbf{y}}_h}} D_i \otimes \mathcal{N}_i + \sum_{i=1}^{r_{\bar{\mathbf{y}}_h}} D_i \otimes \mathcal{H}_i,$$

Algorithm 1. Picard iteration for the Navier-Stokes optimization problem.

Require: Desired state $\mathbf{y}_d$, initial state $\mathbf{y}_0$, initial guess $\bar{\mathbf{y}}$.

for $\ell = 1$ to $\mathrm{L}_{\max}$ **do**
  Compute $\mathcal{D}_i, \mathcal{N}_i, \mathcal{H}_i$ for $i = 1, \ldots, r_{\bar{\mathbf{y}}_h}$
  Update the right-hand side $\bar{d}$
  Solve the system (2.33)
  if Error $\|\mathbf{y}_h - \bar{\mathbf{y}}_h\|$ is small **then**
    Stop
  else
    Replace $\bar{\mathbf{y}}_h = \mathbf{y}_h$ and continue
  end if
end for
return $[\mathbf{y}_h, \mathbf{u}_h, \boldsymbol{\lambda}_h]$

where $\mathcal{H}_i = \mathrm{blkdiag}(\mathbf{H}_i, 0)$.

We have now seen that we can perform an outer Picard iteration and then proceed in a low-rank fashion with the inner Oseen problem. Algorithm 1 depicts a pseudocode of our proposed scheme. In the following we will discuss the numerical solver for the low-rank solution of the linear system (2.30). We also want to discuss the case when the objective function is changed to include the vorticity term (2.4) following results in a different formulation of the adjoint equation [30, 38],

$$-\boldsymbol{\lambda}_t - \Delta\boldsymbol{\lambda} - (\mathbf{y}\cdot\nabla)\,\boldsymbol{\lambda} + (\nabla\mathbf{y})^T\,\boldsymbol{\lambda} + \bigtriangledown\xi = -\alpha_1(\mathbf{y}-\mathbf{y}_d) - \alpha_2\mathbf{curl}(\mathbf{curl}(\mathbf{y})). \tag{2.32}$$

We now get the following system at every nonlinear step:

(2.33)
$$\begin{bmatrix} \Theta\otimes(\alpha_1\mathcal{M}+\alpha_2\mathcal{L}_0) & 0 & \mathcal{K}^* \\ 0 & \Theta\otimes\beta\mathbf{M} & -\mathcal{M}_3^\top \\ \mathcal{K} & -\mathcal{M}_3 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{y}_h \\ p_h \\ \mathbf{u}_h \\ \boldsymbol{\lambda}_h \\ \xi_h \end{bmatrix} = \begin{bmatrix} (\Theta\otimes\alpha_1\mathbf{M})\mathbf{y}_d \\ 0 \\ 0 \\ \bar{d} \\ 0 \end{bmatrix} = \begin{bmatrix} b_1 \\ 0 \\ b_2 \\ b_3 \\ 0 \end{bmatrix},$$

where $\mathcal{L}_0 = \left[\begin{smallmatrix} \mathrm{L} & 0 \\ 0 & 0 \end{smallmatrix}\right]$ represents the discretized version of $\mathbf{curl}(\mathbf{curl}(\mathbf{y}))$, which is just the Laplacian operator, since the velocity is divergence-free.

3. Solution algorithms. We focus now on the efficient solution of the system (2.30) in low-rank form. Having solved the full KKT system (2.30), the low-rank format of the solution (2.27)–(2.28) can be computed by the SVD. This is called an *offline* stage in model reduction methods [39]. Our goal is to avoid this expensive offline stage and compute the low-rank factors of the solution *directly*. One of the best tools for this task is the alternating iteration.

3.1. Alternating low-rank methods for the forward problem. First, we start from a single linear system $Ay = b$, where $A \in \mathbb{R}^{nm\times nm}$ and $b \in \mathbb{R}^{nm}$ are given, and the solution is sought in the low-rank form

$$A = \sum_{i=1}^{r_A} F_{i,A}\otimes G_{i,A}, \qquad b = \sum_{i=1}^{r_b} v_{i,b}\otimes w_{i,b}, \qquad y = \sum_{i=1}^{r_y} v_{i,y}\otimes w_{i,y}.$$

As a motivating example, consider the case $A = A^\top > 0$. Then solving the linear system is equivalent to the minimization of the energy functional, $y = \arg\min_y J(y)$,

where $J(y) = y^\top Ay - 2y^\top b$. Now, plug the low-rank decomposition of y into J, and optimize it sequentially (or *alternating*, hence the name) over v_y and w_y:

$$v_y = \underset{v_y \in \mathbb{R}^{nr_y}}{\arg\min} J\left(\sum_{i=1}^{r_y} v_{i,y} \otimes w_{i,y}\right), \quad w_y = \underset{w_y \in \mathbb{R}^{mr_y}}{\arg\min} J\left(\sum_{i=1}^{r_y} v_{i,y} \otimes w_{i,y}\right), \tag{3.1}$$

where v_y and w_y denote vertically stacked $v_{i,y}$ and $w_{i,y}$. Differentiating J with respect to the elements of v and w, we can find that they are defined by reduced linear systems. To describe the latter, we introduce $V_y \in \mathbb{R}^{n \times r_y}$ and $W_y \in \mathbb{R}^{m \times r_y}$, which are matrices of horizontally stacked $v_{i,y}$ and $w_{i,y}$, respectively. Then (3.1) is satisfied by solving

$$\begin{aligned} \left[(I_n \otimes W_y)^\top A (I_n \otimes W_y)\right] v_y &= (I_n \otimes W_y)^\top b, \\ \left[(V_y \otimes I_m)^\top A (V_y \otimes I_m)\right] w_y &= (V_y \otimes I_m)^\top b, \end{aligned} \tag{3.2}$$

or, in abbreviated form, $\hat{A} v_y = \hat{b}$, $\check{A} w_y = \check{b}$.

These two systems are solved one after another until convergence. Minimization of (3.1) is equivalent to minimization of the squared A-norm of the error; hence the method is called alternating least squares [37]. Since the restricted optimization is a linear problem (3.2), it is also called alternating linear scheme [26], abbreviated by *ALS* in both cases. Although it is difficult to prove theoretical convergence (it is essentially local [52]), in practice this method often converges rapidly, provided the rank r_y is high enough.

However, it is inconvenient to guess the rank a priori. One can start from a rank-1 initial guess and increase it during the course of the computations until a desired error threshold is reached. Hence one needs a procedure for adding new vectors to the low-rank factors V_y and W_y. It is reasonable to select the new vectors related to the current residual, as it allows one to relate the scheme to gradient descent methods. For example, *ADI* methods [59] solve shifted linear systems and expand, e.g., V_y by $(F_A - sI)^{-1} V_R$, where V_R is a low-rank factor of the residual, and *greedy* methods [56, 1, 44] compute rank-1 factors V_z, W_z of the solution to $Az = b - Ay$ and pad V_y with V_z. However, simply augmenting the solution may give a suboptimal representation. An advantage of the variational formulation (3.1) is that the low-rank factors deliver a locally minimal A-norm error in each step. *Orthogonal* greedy [44] and *alternating minimal energy* [12] algorithms combine the steps that insert new vectors into a low-rank factor of the solution with the steps that update the factor as a whole via the Galerkin system (3.2). In this paper we use the latter technique.

These methods converge relatively well if the matrix is positive definite. However, as was noticed in [5], even with orthogonal factors V and W, the Galerkin projection (3.2) can become degenerate if A is a saddle-point system like (2.33). To avoid this issue, we need to take the saddle-point structure into account explicitly.

3.2. Alternating methods for the inverse problem. Let us consider a block system

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix},$$

where each submatrix of A or subvector of b is presented in its own low-rank form, and the sizes of all blocks coincide.[1] However, the *solution* components will be factorized

[1] In practical computations, y_1, y_2, and y_3 have the meanings of $\mathbf{y}_h$, $\mathbf{u}_h$, and $\boldsymbol{\lambda}_h$, but here we present the scheme in an abstract form; therefore we use more abstract notation than in section 2. Later we will also show that the requirement to have all block sizes equal is not restrictive.

with one of the blocks *shared*: we suppose that either

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \sum_{i=1}^{r_y} \begin{bmatrix} v_{i,y,1} \\ v_{i,y,2} \\ v_{i,y,3} \end{bmatrix} \otimes \hat{w}_{i,y} \quad \text{or} \quad \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \sum_{i=1}^{r_y} \check{v}_{i,y} \otimes \begin{bmatrix} w_{i,y,1} \\ w_{i,y,2} \\ w_{i,y,3} \end{bmatrix},$$

where $\hat{w}_{i,y} \in \mathbb{R}^m$, $\check{v}_{i,y} \in \mathbb{R}^n$, and agglomerated matrices are $\hat{W}_y \in \mathbb{R}^{m \times r_y}$ and $\check{V}_y \in \mathbb{R}^{n \times r_y}$. Now, we can write two ALS steps in the *block* form

$$\begin{aligned} &\begin{bmatrix} \hat{A}_{11} & \hat{A}_{12} & \hat{A}_{13} \\ \hat{A}_{21} & \hat{A}_{22} & \hat{A}_{23} \\ \hat{A}_{31} & \hat{A}_{32} & \hat{A}_{33} \end{bmatrix} \begin{bmatrix} v_{y,1} \\ v_{y,2} \\ v_{y,3} \end{bmatrix} = \begin{bmatrix} \hat{b}_1 \\ \hat{b}_2 \\ \hat{b}_3 \end{bmatrix}, \qquad \begin{aligned} \hat{A}_{kl} &= (I \otimes \hat{W}_y)^\top A_{kl} (I \otimes \hat{W}_y), \\ \hat{b}_k &= (I \otimes \hat{W}_y)^\top b_k, \end{aligned} \\ &\begin{bmatrix} \check{A}_{11} & \check{A}_{12} & \check{A}_{13} \\ \check{A}_{21} & \check{A}_{22} & \check{A}_{23} \\ \check{A}_{31} & \check{A}_{32} & \check{A}_{33} \end{bmatrix} \begin{bmatrix} w_{y,1} \\ w_{y,2} \\ w_{y,3} \end{bmatrix} = \begin{bmatrix} \check{b}_1 \\ \check{b}_2 \\ \check{b}_3 \end{bmatrix}, \qquad \begin{aligned} \check{A}_{kl} &= (\check{V}_y \otimes I)^\top A_{kl} (\check{V}_y \otimes I), \\ \check{b}_k &= (\check{V}_y \otimes I)^\top b_k, \end{aligned} \end{aligned} \tag{3.3}$$

where $k, l = 1, 2, 3$. Note that the blocks $\hat{W}_y$ and $\check{V}_y$ do not contain the enumerator k, i.e., they serve as common bases for the components y_k. To compute this common basis (e.g., $\check{V}_y$), we can use the truncated SVD. Similarly to a single V_y in the previous section, we consider each component as a matrix $V_{y,k} \in \mathbb{R}^{n \times r_y}$. Having computed $V_{y,k}$ in the first step of (3.3), we factorize via SVD:

$$\begin{bmatrix} V_{y,1} & V_{y,1} & V_{y,3} \end{bmatrix} = \check{V}_y S P^\top + \mathcal{E}, \quad \text{such that} \quad \|\mathcal{E}\|_F \le \varepsilon \|S\|_F,$$

where $\check{V}_y^\top \check{V}_y = I_{\hat{r}_y}$, S is a diagonal matrix of $\hat{r}_y$ dominant singular values, and $P \in \mathbb{R}^{3r_y \times \hat{r}_y}$ is a matrix of right singular vectors. Left singular vectors $\check{V}_y \in \mathbb{R}^{n \times \hat{r}_y}$ give the sought common basis. In the same way we derive $\hat{W}_y$ from $W_{y,k}$ after the second step of (3.3).

Notice that the new rank $\hat{r}_y$ can be chosen from the range $1, \dots, 3r_y$. That is, the blocked storage allows one to increase the rank without explicit insertion of additional vectors.[2] This is similar to the *density matrix renormalization group* (DMRG) method [60], developed in quantum physics to solve high-dimensional eigenvalue problems in low-rank formats. However, the DMRG method applied in our two-dimensional case would require us to solve the whole problem without any reduction, whereas the block ALS formulation (3.3) allows us to have both the rank adaptivity and moderate complexity.

The block ALS method requires only submatrices A_{kl} to be positive (semi)definite; the whole matrix A needs only to be invertible. A drawback, however, is that the submatrices should be square. Moreover, with the Navier–Stokes constraints, $A_{31} = \mathcal{K}$ (2.31) and $A_{13} = \mathcal{K}^*$ are themselves saddle-point matrices. To avoid this issue, we will compute the pressures separately.

3.3. Alternating methods for the (Navier–)Stokes equation. Let us start from the forward Navier–Stokes equation, which reads

$$\begin{bmatrix} \mathbf{K} & I_n \otimes B^\top \\ I_n \otimes B & \end{bmatrix} \begin{bmatrix} \mathbf{y}_h \\ p_h \end{bmatrix} = \begin{bmatrix} d \\ 0 \end{bmatrix}, \quad \mathbf{K} = C \otimes \mathbf{M} + I_n \otimes \mathbf{L} + \sum_{i=1}^{r_{\mathbf{y}_h}} D_i \otimes \mathbf{N}_i.$$

[2] A residual-based augmentation is still recommended to improve the convergence and accuracy. This is not a bottleneck, however: even a very low rank approximation to the residual provides sufficiently fast convergence [12].

Suppose $\mathbf{y}_h$ is presented in the low-rank form (2.27). If we are performing the second ALS step in (3.3), the matrix remains invertible: the saddle-point structure is introduced only in the spatial variable, and the ALS projection of the temporal variable does not affect it. Then from the second row we have $BW_{\mathbf{y}_h} = 0$. Returning to the first ALS step, we project the first row by $I_n \otimes W_{\mathbf{y}_h}^\top$, so we have

$$(I_n \otimes W_{\mathbf{y}_h})^\top \mathbf{K}(I_n \otimes W_{\mathbf{y}_h})v_{\mathbf{y}_h} + (I_n \otimes W_{\mathbf{y}_h}^\top B^\top)p_h = (I_n \otimes W_{\mathbf{y}_h})^\top d.$$

However, the second term is zero irrespective of the pressure: $W_{\mathbf{y}_h}^\top B^\top = (BW_{\mathbf{y}_h})^\top = 0$. Therefore, the first ALS step is also well-posed in this scheme.

The problem is that the formulation above is valid only with zero boundary conditions. With nonzero Dirichlet conditions enforced, we have either a different matrix instead of $B^\top$, or a nonzero second component of the right-hand side. We need to shift the velocity by some function, such that the sought solution is zero at the boundary.

At this point, it is reasonable to assume that the boundary values, as any other input data, are given in the low-rank form,

$$\mathbf{y}_h|_\Gamma = \sum_{i=1}^{r_\Gamma} v_{\Gamma,i} \otimes w_{\Gamma,i}, \tag{3.4}$$

where Γ denotes boundary degrees of freedom. This is a reasonable assumption as with a particular look to control applications, a highly varying boundary condition might be unlikely. We look for the solution in the form $\mathbf{y} = \mathbf{q} + \boldsymbol{\mu}$, where $\mathbf{q}|_\Gamma = 0$ and $\boldsymbol{\mu}|_\Gamma = \mathbf{y}|_\Gamma$. We could reformulate the equation for $\mathbf{q}$ if we find a convenient closed-form (and low-rank) expression for $\boldsymbol{\mu}$.

This can be done by solving a few stationary Stokes equations. Let us partition the spatial degrees of freedom, and hence, the matrix elements, as follows:

$$\mathbf{L} = \begin{pmatrix} \mathbf{L}_{\Omega\Omega} & \mathbf{L}_{\Omega\Gamma} \\ \mathbf{L}_{\Gamma\Omega} & \mathbf{L}_{\Gamma\Gamma} \end{pmatrix}, \qquad \mathbf{M} = \begin{pmatrix} \mathbf{M}_{\Omega\Omega} & \mathbf{M}_{\Omega\Gamma} \\ \mathbf{M}_{\Gamma\Omega} & \mathbf{M}_{\Gamma\Gamma} \end{pmatrix}, \qquad B = \begin{pmatrix} B_\Omega & B_\Gamma \end{pmatrix},$$

where Ω corresponds to the inner points, and Γ denotes the boundary points. The Stokes equation with nonzero boundary conditions can be written as follows:

$$\begin{bmatrix} \begin{pmatrix} \mathbf{L}_{\Omega\Omega} & \\ & I \end{pmatrix} & \begin{pmatrix} B_\Omega^\top \\ 0 \end{pmatrix} \\ \begin{pmatrix} B_\Omega & B_\Gamma \end{pmatrix} & \end{bmatrix} \begin{bmatrix} w_{\boldsymbol{\mu}_h,i} \\ p_{h,i} \end{bmatrix} = \begin{bmatrix} \begin{pmatrix} -\mathbf{L}_{\Omega\Gamma} w_{\Gamma,i} \\ w_{\Gamma,i} \end{pmatrix} \\ 0 \end{bmatrix}. \tag{3.5}$$

Since the Stokes equation is linear, it admits a superposition: solving (3.5) for all $i = 1, \ldots, r_\Gamma$, we obtain exactly the low-rank factors for $\boldsymbol{\mu}$, and summing them up in

$$\boldsymbol{\mu}_h = \sum_{i=1}^{r_\Gamma} v_{\Gamma,i} \otimes w_{\boldsymbol{\mu}_h,i},$$

we get the desired correction function.

The Navier–Stokes equation can now be rewritten for $\mathbf{q}_h$ without the boundary as

$$\begin{bmatrix} \mathbf{K}_{\Omega\Omega} & I \otimes B_\Omega^\top \\ I \otimes B_\Omega & \end{bmatrix} \begin{bmatrix} \mathbf{q}_h \\ p_h \end{bmatrix} = \begin{bmatrix} b_\Omega - \mathbf{K}_{\Omega:}\boldsymbol{\mu}_h \\ 0 \end{bmatrix}, \quad \text{where} \tag{3.6}$$

$$\begin{aligned}\mathbf{K}_{\Omega\Omega} &= C \otimes \mathbf{M}_{\Omega\Omega} + I \otimes \mathbf{L}_{\Omega\Omega} + \sum_{i=1}^{r_{\bar{\mathbf{y}}_h}} D_i \otimes \mathbf{N}_{i,\Omega\Omega},\\ \mathbf{K}_{\Omega:} &= C \otimes \begin{bmatrix}\mathbf{M}_{\Omega\Omega} & \mathbf{M}_{\Omega\Gamma}\end{bmatrix} + I \otimes \begin{bmatrix}\mathbf{L}_{\Omega\Omega} & \mathbf{L}_{\Omega\Gamma}\end{bmatrix} + \sum_{i=1}^{r_{\bar{\mathbf{y}}_h}} D_i \otimes \begin{bmatrix}\mathbf{N}_{i,\Omega\Omega} & \mathbf{N}_{i,\Omega\Gamma}\end{bmatrix},\end{aligned} \tag{3.7}$$

and $\mathbf{N} = \mathbf{N}(\bar{\mathbf{y}}_h) = \mathbf{N}(\bar{\mathbf{q}}_h + \boldsymbol{\mu}_h)$ is computed as previously from the last iterate $\bar{\mathbf{y}} = \bar{\mathbf{q}} + \boldsymbol{\mu}$, which *includes* the correction $\boldsymbol{\mu}$ together with the boundary nodes. Here, the fixed right-hand side b carries only the initial state, $b = e_1 \otimes \mathbf{M}\mathbf{y}_0$.

The presented ALS scheme computes only the velocity. To restore the pressure, suppose that the velocity is known. Then the first row in (3.6) gives an equation, which can be resolved by least squares:

$$\left(I_n \otimes B_\Omega B_\Omega^\top\right) p_h = \left(I_n \otimes B_\Omega\right)\left(b_\Omega - \mathbf{K}_{\Omega:}\boldsymbol{\mu}_h - \mathbf{K}_{\Omega\Omega}\mathbf{q}_h\right).$$

The right-hand side is low-rank, since $\mathbf{K}, \boldsymbol{\mu}_h$, and $\mathbf{q}_h$ also are, and the matrix in the left-hand side is a direct product, which can be inverted without changing the rank.

A single step of such a method may give only an approximate solution, so we conduct several iterations of Chorin/Gauss–Seidel type:

$$\begin{bmatrix}\hat{\mathbf{K}}_{\Omega\Omega} & I_{r_{\mathbf{q}_h}} \otimes B_\Omega^\top \\ I_{r_{\mathbf{q}_h}} \otimes B_\Omega & \end{bmatrix} \begin{bmatrix} w_{\mathbf{q}_h} \\ dp_h \end{bmatrix} = \begin{bmatrix} \left(V_{\mathbf{q}_h} \otimes I_m\right)^\top \left(b_\Omega - \mathbf{K}_{\Omega:}\boldsymbol{\mu}_h\right) - \left(V_{\mathbf{q}_h}^\top \otimes B_\Omega^\top\right) p_h \\ 0 \end{bmatrix}, \tag{3.8}$$

$$\check{\mathbf{K}}_{\Omega\Omega} v_{\mathbf{q}_h} = \left(I_n \otimes W_{\mathbf{q}_h}\right)^\top \left(b_\Omega - \mathbf{K}_{\Omega:}\boldsymbol{\mu}_h\right),$$

$$\left(I_n \otimes B_\Omega B_\Omega^\top\right) p_h = \left(I_n \otimes B_\Omega\right)\left(b_\Omega - \mathbf{K}_{\Omega:}\boldsymbol{\mu}_h - \mathbf{K}_{\Omega\Omega}\mathbf{q}_h\right),$$

and so on from the first equation. Here, $\hat{\mathbf{K}}_{\Omega\Omega} = \left(V_{\mathbf{q}_h} \otimes I_m\right)^\top \mathbf{K}_{\Omega\Omega} \left(V_{\mathbf{q}_h} \otimes I_m\right)$, and $\check{\mathbf{K}}_{\Omega\Omega} = \left(I_n \otimes W_{\mathbf{q}_h}\right)^\top \mathbf{K}_{\Omega\Omega} \left(I_n \otimes W_{\mathbf{q}_h}\right)$. Remember that $V_{\mathbf{q}_h}$ and $W_{\mathbf{q}_h}$ are orthogonalized before they are used as projectors. A dummy variable dp_h in (3.8), imposing the divergence-free condition, converges to zero and can be discarded after the calculation. In practice, it is sufficient to merge these inner iterations and the outer Picard iterations: we update $\mathbf{N}$ in every step of (3.8).

The inverse problem is solved in a similar way; the only difference is that there are two "pressure-like" variables. Since we neither control nor observe the boundary, the Lagrange multiplier can have zero boundary condition. So, the final ALS iteration for the inverse Navier–Stokes equation is written as follows.

1. In the first step, we compute all spatial blocks,

$$\begin{bmatrix} \hat{\mathbf{M}}_1 & 0 & 0 & \hat{\mathbf{K}}^*_{\Omega\Omega} & I_r \otimes B_\Omega^\top \\ 0 & 0 & 0 & I_r \otimes B_\Omega & 0 \\ 0 & 0 & \hat{\mathbf{M}}_2 & -\hat{\mathbf{M}}_3^\top & 0 \\ \hat{\mathbf{K}}_{\Omega\Omega} & I_r \otimes B_\Omega^\top & -\hat{\mathbf{M}}_3 & 0 & 0 \\ I_r \otimes B_\Omega & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} w_{\mathbf{q}_h} \\ dp_h \\ w_{\mathbf{u}_h} \\ w_{\boldsymbol{\lambda}_h} \\ d\xi_h \end{bmatrix} \tag{3.9}$$

$$= \begin{bmatrix} \hat{b}_1 - (\check{V}^\top \otimes B_\Omega^\top)\xi_h \\ 0 \\ 0 \\ \hat{b}_3 - (\check{V}^\top \otimes B_\Omega^\top) p_h \\ 0 \end{bmatrix},$$

where $\hat{b}_1 = (\check{V}^\top \Theta \otimes \alpha_1 \mathbf{M}_{\Omega:})\mathbf{y}_d$, $\hat{b}_3 = (\check{V} \otimes I_m)^\top (b_\Omega - \mathbf{K}_{\Omega:}\boldsymbol{\mu}_h)$,

$$
\begin{aligned}
\hat{\mathbf{M}}_1 &= (\check{V}^\top \Theta \check{V}) \otimes (\alpha_1 \mathbf{M}_{\Omega\Omega} + \alpha_2 \mathbf{L}_{\Omega\Omega}),\\
\hat{\mathbf{M}}_2 &= (\check{V}^\top \Theta \check{V}) \otimes \beta \mathbf{M}_{\Omega\Omega},\\
\hat{\mathbf{M}}_3 &= I_r \otimes \mathbf{M}_{\Omega\Omega},\\
\hat{\mathbf{K}}_{\Omega\Omega} &= \check{V}^\top C \check{V} \otimes \mathbf{M}_{\Omega\Omega} + I_r \otimes \mathbf{L}_{\Omega\Omega} + \sum_{i=1}^{r_{\bar{\mathbf{y}}_h}} \check{V}^\top D_i \check{V} \otimes \mathbf{N}_{i,\Omega\Omega},\\
\hat{\mathbf{K}}^*_{\Omega\Omega} &= \hat{\mathbf{K}}^\top_{\Omega\Omega} + \sum_{i=1}^{r_{\bar{\mathbf{y}}_h}} \check{V}^\top D_i \check{V} \otimes (\mathbf{H}_{i,\Omega\Omega} - \mathbf{N}_{i,\Omega\Omega} - \mathbf{N}^\top_{i,\Omega\Omega}).
\end{aligned}
\tag{3.10}
$$

We compute the SVD $\begin{bmatrix} W_{\mathbf{q}_h} & W_{\mathbf{u}_h} & W_{\boldsymbol{\lambda}_h} \end{bmatrix} \approx \hat{W} S P^\top$ to derive the common basis.

2. In the second step, we compute the temporal blocks of the velocities,

$$
\begin{bmatrix} \check{\mathbf{M}}_1 & 0 & \check{\mathbf{K}}^*_{\Omega\Omega} \\ 0 & \check{\mathbf{M}}_2 & -\check{\mathbf{M}}_3^\top \\ \check{\mathbf{K}}_{\Omega\Omega} & -\check{\mathbf{M}}_3 & 0 \end{bmatrix} \begin{bmatrix} v_{\mathbf{q}_h} \\ v_{\mathbf{u}_h} \\ v_{\boldsymbol{\lambda}_h} \end{bmatrix} = \begin{bmatrix} \check{b}_1 \\ 0 \\ \check{b}_3 \end{bmatrix},
\tag{3.11}
$$

where $\check{b}_1 = (\Theta \otimes \alpha_1 \hat{W}^\top \mathbf{M}_{\Omega:})\mathbf{y}_d$, $\check{b}_3 = (I_n \otimes \hat{W})^\top (b_\Omega - \mathbf{K}_{\Omega:}\boldsymbol{\mu}_h)$,

$$
\begin{aligned}
\check{\mathbf{M}}_1 &= \Theta \otimes \hat{W}^\top (\alpha_1 \mathbf{M}_{\Omega\Omega} + \alpha_2 \mathbf{L}_{\Omega\Omega}) \hat{W},\\
\check{\mathbf{M}}_2 &= \Theta \otimes \beta \hat{W}^\top \mathbf{M}_{\Omega\Omega} \hat{W},\\
\check{\mathbf{M}}_3 &= I_n \otimes \hat{W}^\top \mathbf{M}_{\Omega\Omega} \hat{W},\\
\check{\mathbf{K}}_{\Omega\Omega} &= C \otimes \hat{W}^\top \mathbf{M}_{\Omega\Omega} \hat{W} + I_n \otimes \hat{W}^\top \mathbf{L}_{\Omega\Omega} \hat{W} + \sum_{i=1}^{r_{\bar{\mathbf{y}}_h}} D_i \otimes \hat{W}^\top \mathbf{N}_{i,\Omega\Omega} \hat{W},\\
\check{\mathbf{K}}^*_{\Omega\Omega} &= \check{\mathbf{K}}^\top_{\Omega\Omega} + \sum_{i=1}^{r_{\bar{\mathbf{y}}_h}} D_i \otimes \hat{W}^\top (\mathbf{H}_{i,\Omega\Omega} - \mathbf{N}_{i,\Omega\Omega} - \mathbf{N}^\top_{i,\Omega\Omega}) \hat{W}.
\end{aligned}
\tag{3.12}
$$

We compute the SVD $\begin{bmatrix} V_{\mathbf{q}_h} & V_{\mathbf{u}_h} & V_{\boldsymbol{\lambda}_h} \end{bmatrix} \approx \check{V} S P^\top$ to derive the common basis.

3. In the third step, we update the pressures via standard low-rank algebra,

$$
\begin{aligned}
\left(I_n \otimes B_\Omega B_\Omega^\top\right) p_h &= \left(I_n \otimes B_\Omega\right)\left(b_\Omega - \mathbf{K}_{\Omega:}\boldsymbol{\mu}_h - \mathbf{K}_{\Omega\Omega}\mathbf{q}_h + \mathbf{M}_3 \mathbf{u}_h\right),\\
\left(I_n \otimes B_\Omega B_\Omega^\top\right) \xi_h &= \left(I_n \otimes B_\Omega\right)\left(\Theta \otimes \alpha_1 \mathbf{M}\mathbf{y}_d - \mathbf{K}^*_{\Omega\Omega}\boldsymbol{\lambda}_h - \mathbf{M}_1 \mathbf{q}_h\right).
\end{aligned}
\tag{3.13}
$$

3.4. Preconditioning for the spatial system. The systems on the temporal factor (3.11), (3.12) and pressure (3.13) have moderate sizes and are essentially sparse. Our concern is now the spatial system (3.9), which can be too large for a direct solver even for a rank-1 solution. We solve this system by a preconditioned GMRES method. First, we use a block Jacobi approximation with respect to the rank dimension: the preconditioner reads $\tilde{A} = \mathrm{blkdiag}(\tilde{A}_1, \ldots, \tilde{A}_r)$, where

$$
\tilde{A}_i = \begin{bmatrix} \tilde{\mathbf{M}}_{1,i} & 0 & 0 & \tilde{\mathbf{K}}^*_{i,\Omega\Omega} & B_\Omega^\top \\ 0 & 0 & 0 & B_\Omega & 0 \\ 0 & 0 & \tilde{\mathbf{M}}_{2,i} & -\mathbf{M}^\top & 0 \\ \tilde{\mathbf{K}}_{i,\Omega\Omega} & B_\Omega^\top & -\mathbf{M} & 0 & 0 \\ B_\Omega & 0 & 0 & 0 & 0 \end{bmatrix},
\qquad
\begin{aligned}
\tilde{\mathbf{M}}_{1,i} &= (\check{v}_i \otimes I_m)^\top \mathbf{M}_1 (\check{v}_i \otimes I_m),\\
\tilde{\mathbf{M}}_{2,i} &= (\check{v}_i \otimes I_m)^\top \mathbf{M}_2 (\check{v}_i \otimes I_m),\\
\tilde{\mathbf{K}}_{i,\Omega\Omega} &= (\check{v}_i \otimes I_m)^\top \mathbf{K}_{\Omega\Omega} (\check{v}_i \otimes I_m),\\
\tilde{\mathbf{K}}^*_{i,\Omega\Omega} &= (\check{v}_i \otimes I_m)^\top \mathbf{K}^*_{\Omega\Omega} (\check{v}_i \otimes I_m),
\end{aligned}
$$

for $i = 1, \dots, r$. Solution of each of these systems is of the same complexity as solution of the stationary problem. We precondition it by a block-triangular factor of the LU decomposition, where the Schur complement is approximated in the factored form using the matching argument [54]. Given the right-hand side $f = \begin{bmatrix} f_1 & f_2 & f_3 \end{bmatrix}^\top$, the inverse triangular factor can be applied as follows:

1. $y_1 = \tilde{\mathcal{M}}_{1,i}^{-1} f_1$,
2. $y_2 = \tilde{\mathbf{M}}_{2,i}^{-1} f_2$,
3. $$y_3 = \mathcal{S}^{-1} \left(\tilde{\mathcal{K}}_{i,\Omega\Omega} y_1 - \begin{pmatrix} \mathbf{M} \\ 0 \end{pmatrix} y_2 - f_3 \right),$$

followed by the assembly $y = \begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix}^\top$, where the first matrix is augmented as

$$\tilde{\mathcal{M}}_{1,i} = \begin{bmatrix} \tilde{\mathbf{M}}_{1,i} & 0 \\ 0 & \tau h^2 \beta I \end{bmatrix} \quad \text{and} \quad \tilde{\mathcal{K}}_{i,\Omega\Omega} = \begin{bmatrix} \tilde{\mathbf{K}}_{i,\Omega\Omega} & B_\Omega^\top \\ B_\Omega & 0 \end{bmatrix}.$$

Now, instead of the exact Schur complement, we use the factored approximation

$$\mathcal{S}^{-1} = \begin{bmatrix} \tilde{\mathbf{K}}_{i,\Omega\Omega} + \frac{1}{\sqrt{\beta}} \mathbf{M} & B_\Omega^\top \\ B_\Omega & 0 \end{bmatrix}^{-1} \tilde{\mathcal{M}}_{1,i} \begin{bmatrix} \tilde{\mathbf{K}}^*_{i,\Omega\Omega} + \frac{1}{\sqrt{\beta}} \mathbf{M} & B_\Omega^\top \\ B_\Omega & 0 \end{bmatrix}^{-1}. \tag{3.14}$$

The matrices in the last equation are of the form of the forward stationary Navier–Stokes equation and, for moderate grids, can be treated by the direct linear solver. The use of preconditioned iterative solvers allows applicability to large-scale systems (see [7, 15, 54, 41] for preconditioning strategies).

4. Existence of low-rank solutions. Efficiency of low-rank decompositions depends heavily on the particular values of the rank. However, it is difficult to estimate the ranks of the solution of the forward Navier–Stokes equation; moreover, in a highly turbulent regime they may actually reach the maximal values. We begin the analysis with the Stokes equation, which is to be followed by the analysis of the inverse Navier–Stokes equation. The ranks in the inverse problem can actually be smaller than the ranks of the forward problem if the desired state is taken to be low-rank, for example, as the Stokes solution.

LEMMA 4.1. *Given the Stokes system* (2.5)–(2.6)*, after the discretization,*

$$\begin{bmatrix} C \otimes \mathbf{M} + I \otimes \mathbf{L} & I \otimes B^\top \\ I \otimes B & \end{bmatrix} \begin{bmatrix} \mathbf{y}_h \\ p_h \end{bmatrix} = \begin{bmatrix} \mathbf{u}_h \\ 0 \end{bmatrix}, \tag{4.1}$$

where C is the time difference matrix (2.19) *of size n, and $\mathbf{L}$ and $\mathbf{M}$ are the finite element discretization of Laplace and mass operators, respectively. Suppose the right-hand side is given in the low-rank form* (2.29)*, the boundary condition is given in the low-rank form* (3.4)*, and the solution is approximated in the form*

$$\begin{bmatrix} \mathbf{y}_h \\ p_h \end{bmatrix} \approx \sum_{i=1}^{r} \check{v}_i \otimes \begin{bmatrix} w_{i,\mathbf{y}} \\ w_{i,p} \end{bmatrix}$$

up to an accuracy ε.

Then the rank of the solution is bounded by

$$r = \mathcal{O}\left(\left(\log \varepsilon^{-1} + \log h^{-1} + \log \tau^{-1} \right)^2 (r_{\mathbf{u}_h} + r_\Gamma) \right).$$

Proof. First, we exclude the boundary condition as described in the previous section, and arrive at the Stokes system on $\mathbf{q}_h$,

$$\begin{bmatrix} C \otimes \mathbf{M}_{\Omega\Omega} + I \otimes \mathbf{L}_{\Omega\Omega} & I \otimes B_\Omega^\top \\ I \otimes B_\Omega & \end{bmatrix} \begin{bmatrix} \mathbf{q}_h \\ p_h \end{bmatrix} = \begin{bmatrix} f_\Omega \\ 0 \end{bmatrix},$$

with $f_\Omega = \mathbf{u}_\Omega - C \otimes \begin{pmatrix} \mathbf{M}_{\Omega\Gamma} & \mathbf{M}_{\Omega\Gamma} \end{pmatrix} \boldsymbol{\mu}_h - I \otimes \begin{pmatrix} \mathbf{L}_{\Omega\Gamma} & \mathbf{L}_{\Omega\Gamma} \end{pmatrix} \boldsymbol{\mu}_h$.

Denoting by Φ the orthonormal basis of the kernel of B_Ω, we conclude that $\mathbf{q}_h \in \operatorname{span}(I \otimes \Phi)$, or $\mathbf{q}_h = (I \otimes \Phi)\tilde{\mathbf{q}}_h$. The coefficients $\tilde{\mathbf{q}}_h$ can be found by projecting the velocity equation onto $I \otimes \Phi^\top$. Since $\Phi^\top B_I^\top = (B_I \Phi)^\top = 0$, we have

$$\left(C \otimes \tilde{\mathbf{M}} + I \otimes \tilde{\mathbf{L}}\right) \tilde{\mathbf{q}}_h = \left(I \otimes \Phi^\top\right) f_\Omega, \tag{4.2}$$

where

$$\tilde{\mathbf{M}} = \Phi^\top \mathbf{M}_{\Omega\Omega} \Phi, \quad \tilde{\mathbf{L}} = \Phi^\top \mathbf{L}_{\Omega\Omega} \Phi.$$

Since both $\mathbf{M}_{\Omega\Omega}$ and $\mathbf{L}_{\Omega\Omega}$ are symmetric positive definite and Φ is orthogonal, it holds that $\tilde{\mathbf{M}}$ and $\tilde{\mathbf{L}}$ are symmetric positive definite. We can premultiply (4.2) by $(I \otimes \tilde{\mathbf{M}})^{-1}$ to then get

$$\tilde{\mathbf{q}}_h = \left(C \otimes I_m + I_n \otimes \tilde{\mathbf{M}}^{-1}\tilde{\mathbf{L}}\right)^{-1} \left(\left(I \otimes \tilde{\mathbf{M}}^{-1}\Phi^\top\right) f_\Omega\right).$$

The terms in the first brackets commute, and each of them is positive definite. The inverse can be approximated in the low-rank form by the exponential quadrature [16, 20]: for given R and k we introduce $t_k = \exp(k\pi/\sqrt{R})$, $c_k = t_k \pi/\sqrt{R}$; then

$$\tilde{\mathbf{K}}^{-1} \equiv \left(C \otimes I_m + I_n \otimes \tilde{\mathbf{M}}^{-1}\tilde{\mathbf{L}}\right)^{-1} \approx \sum_{k=-R}^{R} c_k \exp(-t_k C) \otimes \exp(-t_k \tilde{\mathbf{M}}^{-1}\tilde{\mathbf{L}}),$$

where the accuracy is estimated by $\mathcal{O}(\|\tilde{\mathbf{K}}\|_2 \mathrm{e}^{-\pi\sqrt{2R}})$, provided that $\|\tilde{\mathbf{K}}^{-1}\| = \mathcal{O}(1)$. Therefore, the rank of $\tilde{\mathbf{K}}^{-1}$ is estimated by $\mathcal{O}((\log \varepsilon^{-1} + \log \operatorname{cond} \tilde{\mathbf{K}})^2)$.

Remember that the rank of $\mathbf{u}_h$, and, therefore, of $\mathbf{u}_\Omega$, is bounded by $r_{\mathbf{u}_h}$. The solution $\mathbf{q}_h$ in the initial finite element basis is restored without changing the rank, by multiplying $I \otimes \Phi$ by $\tilde{\mathbf{q}}_h$. Moreover, $\operatorname{cond} \tilde{\mathbf{K}} = \mathcal{O}(h^{-2} + \tau^{-1})$, where h is the spatial, and τ is the time grid steps. Finally, $\operatorname{rank}(\mathbf{q}_h) \le (2R+1)(r_{\mathbf{u}_h} + 2r_\Gamma)$, and the rank of the velocity $\mathbf{y}_h$ is estimated immediately, since the rank of $\boldsymbol{\mu}_h$ is r_Γ.

From (3.8) we see that the pressure rank is $r_{\mathbf{u}_h} + \operatorname{rank}(\mathbf{K}_{\Omega:}\mathbf{y})$, where now $\mathbf{K}_{\Omega:}$ has rank 2, which concludes the lemma. ∎

If the right-hand side and boundary conditions are sufficiently smooth functions, it often holds that $r_{\mathbf{u}_h}$ and r_Γ can be bounded by $\log \varepsilon^{-1}$ for the accuracy ε; see, e.g., [32]. Moreover, space and time grid steps h and τ can be adjusted to deliver the same discretization error ε. The Taylor–Hood finite elements have an approximation order $\varepsilon = \mathcal{O}(h^2)$, while the implicit Euler discretization in time provides $\varepsilon = \mathcal{O}(\tau)$. Plugging the corresponding h and τ into the rank estimate of Lemma 4.1, we obtain the following.

COROLLARY 4.2. *In the assumptions above, the rank of the ε-approximation of the solution of the Stokes equation can be bounded as*

$$r = \mathcal{O}\left(\log^3 \varepsilon^{-1}\right).$$

The total number of degrees of freedom in the low-rank representation is

$$\mathcal{O}(h^{-2} + \tau^{-1}) r = \mathcal{O}(\varepsilon^{-1} \log^3 \varepsilon^{-1}).$$

For comparison, the full solution requires

$$\mathcal{O}(h^{-2} \cdot \tau^{-1}) = \mathcal{O}(\varepsilon^{-2})$$

degrees of freedom.

Remark 4.3. We can use the solution of the Stokes equation as a desired state in the misfit optimization (2.3), constrained by the Navier–Stokes equation (with smaller viscosity). This is reasonable if we want to get rid of turbulences. Under certain conditions [55] it can be shown that the misfit decays with the regularization parameter, $\|\mathbf{y}_h - \mathbf{y}_d\| = \mathcal{O}(\sqrt{\beta})$. Therefore, if we select $\varepsilon \sim \sqrt{\beta}$, Lemma 4.1 is also valid for the solution of the inverse Navier–Stokes problem.

If the state can be approximated by a low-rank decomposition, the low-rankness of the control comes straightforwardly, taking into account the Kronecker form of the forward operator (2.31). It is enough to multiply (2.31) by a rank-$r_{\mathbf{y}_h}$ representation.

COROLLARY 4.4. *Let the state be decomposed in a low-rank form* (2.27) *with the rank* $r_{\mathbf{y}_h}$. *Then the control admits a low-rank decomposition* (2.29) *with the rank* $r_{\mathbf{u}_h} \le r_{\mathbf{y}_h}^2 + 2r_{\mathbf{y}_h}$.

5. Numerical results. Our implementation is based on the IFISS package [14] and the Tensor Train Toolbox [47], both of which are MATLAB-based packages. Nevertheless, the methods presented here are usable in any other computational environment.

The benchmark problem for which we implemented our approach is the well-known *backward facing step* as described in [15, 14]. The domain is set to have length $L = 5$ (see Figure 11). The inflow condition $\mathbf{y}_1|_{x_1=-1} = x_2(1-x_2)(1-\mathrm{e}^{-0.1t})$ is imposed at the left boundary, the Neumann boundary condition at the right boundary $x_1 = L$, and the zero condition at other walls. Other default parameters are given in Table 1. In the experiments below, we will vary each of them separately. The spatial discretization performed within IFISS utilizes the well-known *Taylor–Hood* $\mathbf{Q}_2/\mathbf{Q}_1$ finite elements.

TABLE 1
Default simulation parameters.

n	h	T	ε	ν	β	α_1	α_2
2^{11}	1/32	200	10^{-4}	10^{-3}	10^{-3}	1	0

We consider two types of the objective functional (2.4). First, we minimize only the distance to the desired state (i.e., $\alpha_1 = 1$, $\alpha_2 = 0$), where the desired state is the solution of the Stokes equation. Second, we minimize only the vorticity without any desired state by setting $\alpha_1 = 0$ and $\alpha_2 = 1$. Both observation and control domains coincide with the whole domain, $Q_o = Q_c = Q$.

There are three ways to estimate the error in the solution. First, we compute the relative residual of the KKT system (2.33),

$$\begin{aligned}\text{residual} = &\left\| \mathcal{M}_1 \begin{bmatrix} \mathbf{y}_h \\ p_h \end{bmatrix} + \mathcal{K}^* \begin{bmatrix} \boldsymbol{\lambda}_h \\ \xi_h \end{bmatrix} - b_1 \right\|_F / \|b_1\|_F \\ &+ \left\| \mathcal{M}_2 \mathbf{u}_h - \mathcal{M}_3^\top \begin{bmatrix} \boldsymbol{\lambda}_h \\ \xi \end{bmatrix} \right\|_F / \|\mathcal{M}_2 \mathbf{u}_h\|_F \\ &+ \left\| \mathcal{K} \begin{bmatrix} \mathbf{y}_h \\ p_h \end{bmatrix} - \mathcal{M}_3 \mathbf{u} - b_3 \right\|_F / \|b_3\|_F.\end{aligned}$$

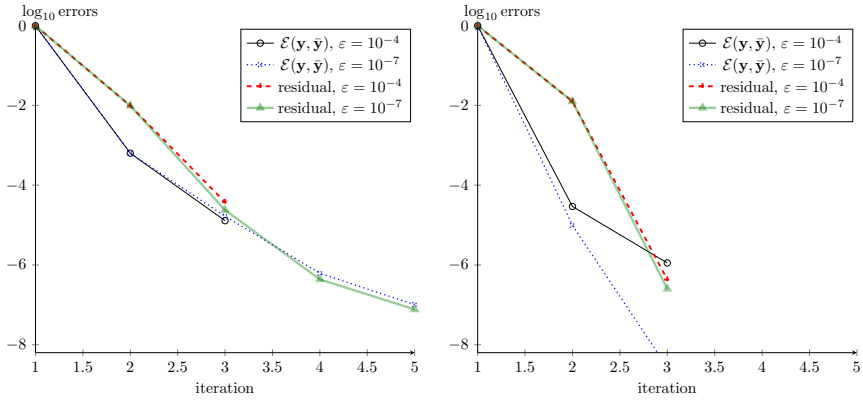

FIG. 1. *Nonlinear convergence. Left: Misfit minimization ($\alpha_1 = 1, \alpha_2 = 0$). Right: Vorticity minimization ($\alpha_1 = 0, \alpha_2 = 1$).*

Second, we can solve the problem with two thresholds, e.g., ε and 0.1ε. Denoting, e.g., the state velocity of the former as $\mathbf{y}$, and of the latter as $\mathbf{y}_\star$, we can compute

$$\mathcal{E}(\mathbf{y}, \mathbf{y}_\star) = \|\mathbf{y} - \mathbf{y}_\star\|_F / \|\mathbf{y}_\star\|_F,$$

and similarly for the control $\mathbf{u}$ and other quantities. Let us assume that the true error $\|\mathbf{y} - \mathbf{y}_{ex}\|$ depends almost linearly on ε, $\|\mathbf{y} - \mathbf{y}_{ex}\| = C\varepsilon + o(\varepsilon)$. Justification of this linear dependence is given by Figures 4 and 7. Then

$$\|\mathbf{y} - \mathbf{y}_{ex}\| = C\varepsilon + o(\varepsilon) \le \|\mathbf{y} - \mathbf{y}_\star\| + \|\mathbf{y}_\star - \mathbf{y}_{ex}\| = \|\mathbf{y} - \mathbf{y}_\star\| + C \cdot 0.1\varepsilon + o(\varepsilon),$$

and so $\|\mathbf{y} - \mathbf{y}_{ex}\| \le \frac{1}{0.9}\|\mathbf{y} - \mathbf{y}_\star\| + o(\varepsilon)$.

Third, we can measure the distance between two Picard iterations, $\mathcal{E}(\mathbf{y}, \bar{\mathbf{y}})$.

5.1. Convergence of the Picard iteration. In the first test, we check the convergence of the residual with the Picard iteration in the low-rank scheme. We test both distance and vorticity minimization and two accuracy thresholds, $\varepsilon = 10^{-4}$ and $\varepsilon = 10^{-7}$. The results are presented in Figure 1.

We see that the convergence is very fast and attained in three iterations in both lower-accuracy tests. The vorticity minimization converges faster than the misfit minimization, since the Stokes solution might actually be more turbulent than the one with the minimal vorticity.

5.2. Optimization of a tracking type functional.

5.2.1. Comparison with the full scheme. An important justification for a new approach is a comparison with an established method. In our case, we compare the low-rank scheme (LR) with the classical preconditioned GMRES for the full KKT system without low-rank approximations. Since the full data do not fit into the memory on fine grids, we perform two tests with $h = 1/8$ and $h = 1/16$, while for the low-rank method we are able to compute the solution for much finer meshes. The results are reported in Figures 2 and 3, respectively. In the horizontal axes, we vary the number of time steps n. Note that the full scheme is unable to proceed with $n = 512$ and $h = 1/16$, while it is still possible with $h = 1/8$.

We see that the residuals are almost the same in both schemes, although the control error grows with the time grid refinement. Another interesting quantity is the

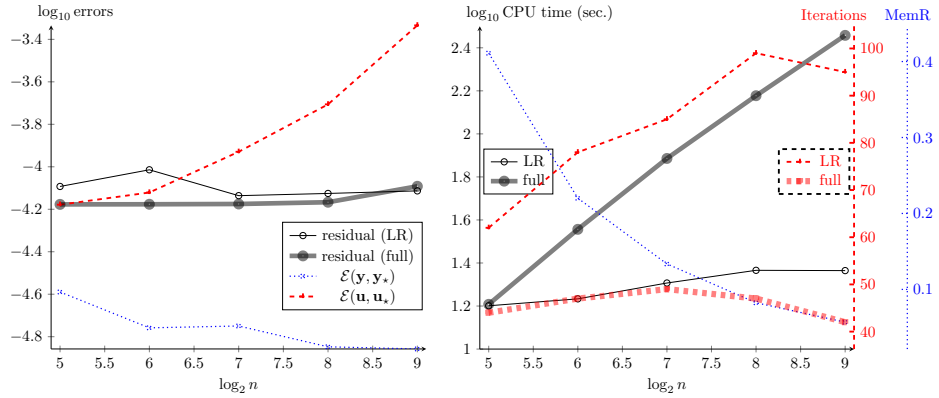

FIG. 2. *Tracking functional, LR versus full, $h = 1/8$. Left: Residual and errors with respect to the reference solutions. Right: CPU time, total number of local iterations, rank.*

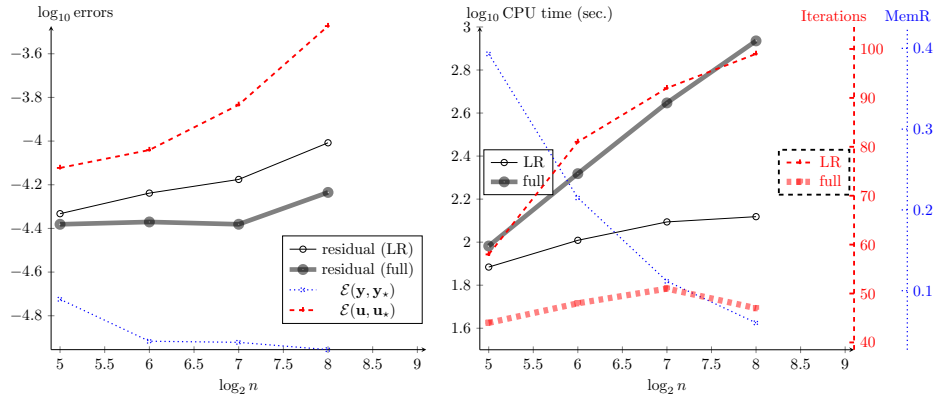

FIG. 3. *Tracking functional, LR versus full, $h = 1/16$. Left: Residual and errors with respect to the reference solutions. Right: CPU time, total number of local iterations, rank.*

number of GMRES iterations. Both schemes perform the block Jacobi preconditioning with respect to time. However, the low-rank method invokes the GMRES (for the spatial factor) twice in each Picard iteration due to the ALS procedure. Moreover, it is being perturbed by the SVD truncation. This results in a higher number of iterations. However, the CPU time of the full scheme is at least equal to the time of the low-rank scheme for the coarsest grid and becomes larger for finer grids. The ratio of memory costs, needed for the low-rank and full solutions, is computed as $\text{MemR} = \frac{(n+m)r}{nm}$. It decreases with the system size and falls below 10% when the number of time steps exceeds several hundreds.

5.2.2. Experiment with the total accuracy. Now we investigate how the scheme performs with simultaneous refinement of the time and spatial grid sizes and the low-rank approximation accuracy. The corresponding errors are independent; hence the total error in the solution is the sum of the three. To synchronize the number of temporal and spatial degrees of freedom with the total accuracy, we first set the low-rank truncation accuracy to $\varepsilon = 10^{-6}$ and determine discretization errors via log-linear fits; see Table 2. As the output quantities, we consider the squared total

TABLE 2
Tracking functional, total vorticity $\|\mathrm{curl}\mathbf{y}\|^2$*, and mass* $\|\mathbf{y}\|^2$ *versus discretization parameters.*

Space			Time		
$1/h$	$\|\mathrm{curl}\mathbf{y}\|^2$	$\|\mathbf{y}\|^2$	n	$\|\mathrm{curl}\mathbf{y}\|^2$	$\|\mathbf{y}\|^2$
8	10.2315	1.81515	256	10.4985	1.82330
16	10.4201	1.81753	512	10.4902	1.82143
32	10.4842	1.82003	1024	10.4862	1.82050
64	10.5107	1.82115	2048	10.4842	1.82003

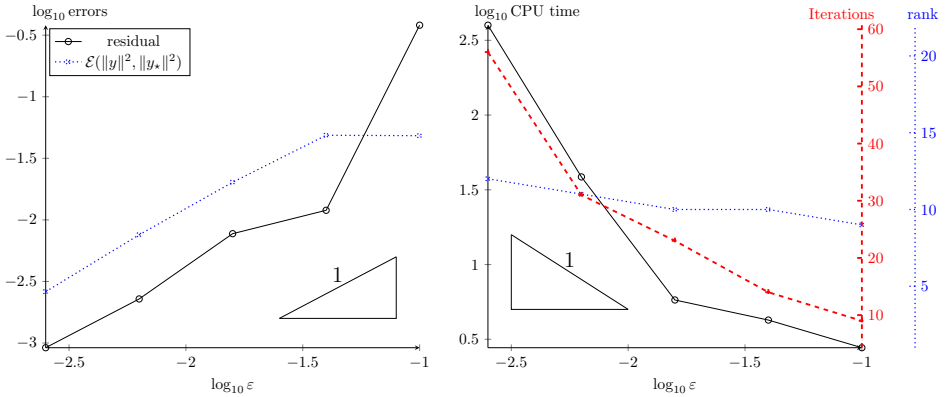

FIG. 4. *Tracking functional,* n *and* h *vary together with* ε*. Left: Residual and error with respect to the reference solution (computed with* $\varepsilon = 10^{-4}$*, corresponding to* $n = 2^{12}$*,* $h = 2^{-8}$*). Right: CPU time, total number of local iterations, rank.*

vorticity and the total mass of the solution.

Using Runge's rule of estimating the error, as well as the least squares fit, we obtain the following dependencies:

$$\log_2 h = \frac{\log_2 \varepsilon + 0.36}{1.62} \quad \text{and} \quad \log_2 n = \frac{\log_2 \varepsilon + 0.604}{-1.1}.$$

In Figure 4, we vary ε and the corresponding h and n accordingly.

We see that the actual error depends linearly on ε, which indicates that both grid parameters are varied properly. The CPU time grows a bit faster than inversely proportional to ε, since the spatial discretization order is less than 2, while the number of spatial degrees of freedom grows as h^{-2}. The lower discretization order in space is due to the corner singularity at the point where the channel expands.

5.2.3. Experiment with β. In this test, we vary the control regularization parameter. The results are shown in Figure 5. As an additional quantity, we report the distance to the desired state, $\mathcal{E}(\mathbf{y}, \mathbf{y}_d)$. This distance decays proportionally to $\sqrt{\beta}$, until being contaminated by the SVD error of level ε.

The scheme is quite robust until β becomes too large. In the right panel of Figure 5, we also show the number of Picard iterations until convergence. We see that for small β the scheme needs three Picard iterations (in agreement with Figure 1), but for $\beta \geq 0.1$ the convergence becomes slower. In particular, for $\beta \geq 0.1$, we were out of the CPU time due to the number of Picard iterations. In the future it might be more appropriate to use Newton methods for strongly nonlinear systems. In general, smaller values for β are more important as they allow the state to better approximate the desired state.

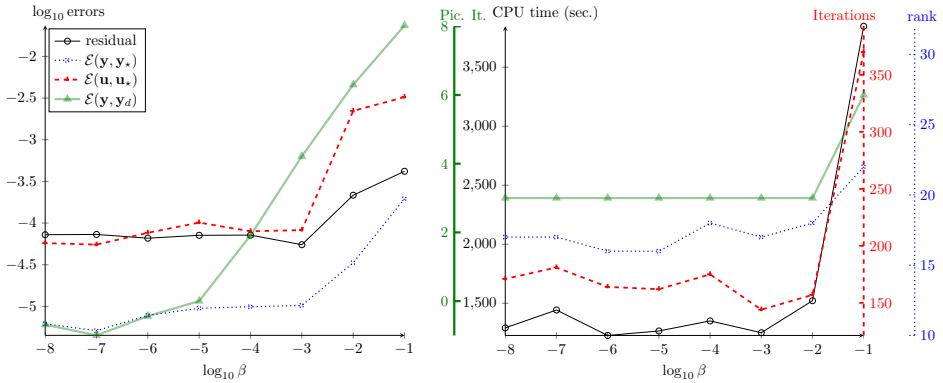

FIG. 5. *Tracking functional, β varies. Left: Residual and errors with respect to the reference solutions. Right: CPU time, total number of local iterations, rank.*

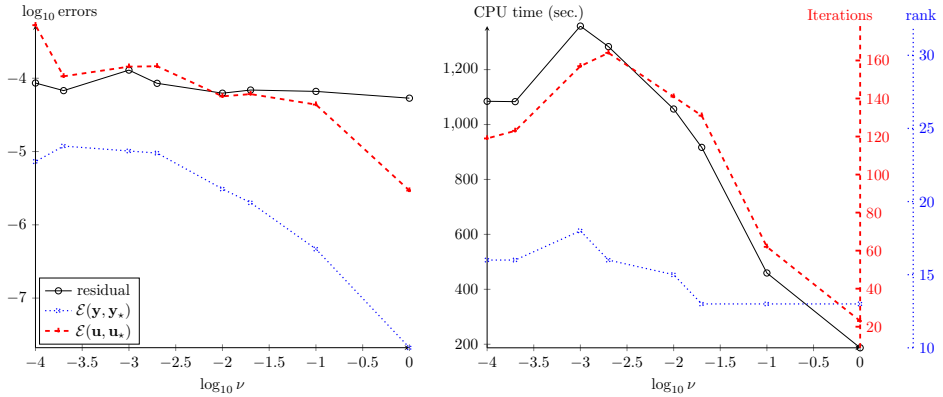

FIG. 6. *Tracking functional, ν varies. Left: Residual and errors with respect to the reference solutions. Right: CPU time, total number of local iterations, rank.*

5.2.4. Experiment with ν. Another parameter to vary is the viscosity. The results are shown in Figure 6. It is natural that all performance indicators improve with larger viscosity as the system becomes closer to the Stokes regime. Nonetheless, even a convection dominated simulation with $\nu = 1/10000$ is tractable. The justification for the existence of low-rank solutions is given in the optimization problem as we assume that the desired state, i.e., a dominating part of the right-hand side of the linear system, is of low rank and hence the solution typically is, too. This is not necessarily true for the forward problem.

5.3. Optimization of a vorticity functional. In this section, we consider the case of the vorticity minimization. The default parameters are the same as in the previous section. The results are presented in the same layout. Table 3 and Figure 7 show the performance of the scheme with respect to the total space-time approximation error. In Figure 8 we vary the time interval T, in Figure 9 we investigate the regularization parameter β, and in Figure 10 we show the role of the viscosity ν.

The behavior of the method for the vorticity minimization is highly similar to the case of the misfit minimization. Here we outline the main differences. First, the

TABLE 3
Vorticity functional, total vorticity $\|\mathrm{curl}\mathbf{y}\|^2$ *and mass* $\|\mathbf{y}\|^2$ *versus discretization parameters.*

Space			Time		
$1/h$	$\|\mathrm{curl}\mathbf{y}\|^2$	$\|\mathbf{y}\|^2$	n	$\|\mathrm{curl}\mathbf{y}\|^2$	$\|\mathbf{y}\|^2$
8	9.88707	1.85858	256	10.0755	1.87087
16	10.0043	1.86465	512	10.0648	1.86889
32	10.0569	1.86742	1024	10.0595	1.86791
64	10.0812	1.86859	2048	10.0569	1.86742

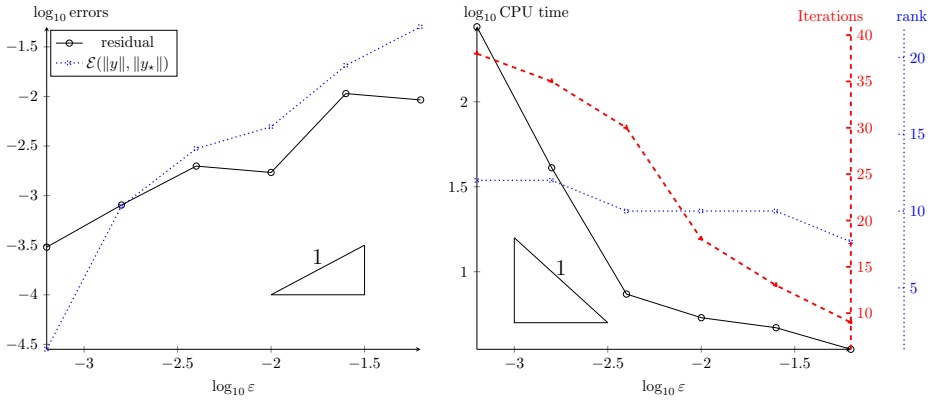

FIG. 7. *Vorticity functional,* h *and* m *vary together with* ε*. Left: Residual and error with respect to the reference solution (computed with* $n = 2^{12}$*,* $h = 2^{-8}$*,* $\varepsilon = 10^{-4}$*). Right: CPU time, total number of local iterations, rank.*

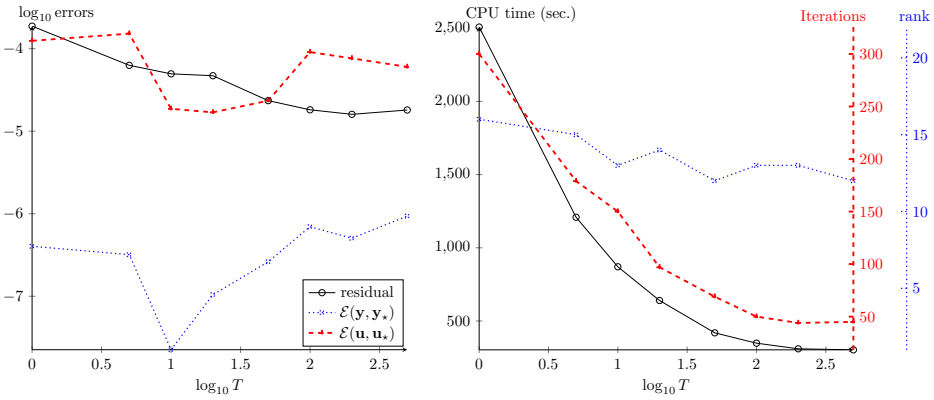

FIG. 8. *Vorticity functional,* T *varies. Left: Residual and errors with respect to the reference solutions. Right: CPU time, total number of local iterations, rank.*

solution with minimal vorticity exhibits smaller ranks than the Stokes solution: 10–15 versus 20–30 in the previous section. This leads to smaller computational times. However, the Laplace operator in the observation matrix $\mathcal{M}_1$ leads to higher condition numbers of the KKT matrix, and hence higher numbers of GMRES iterations.

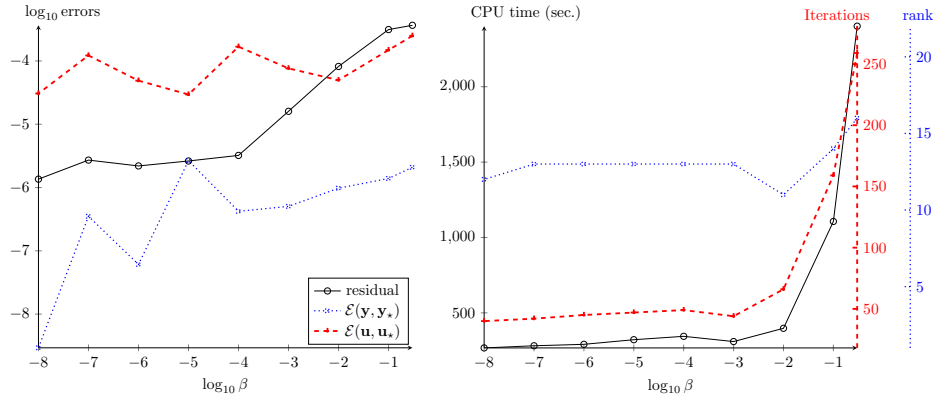

FIG. 9. *Vorticity functional, β varies. Left: Residual and errors with respect to the reference solutions. Right: CPU time, total number of local iterations, rank.*

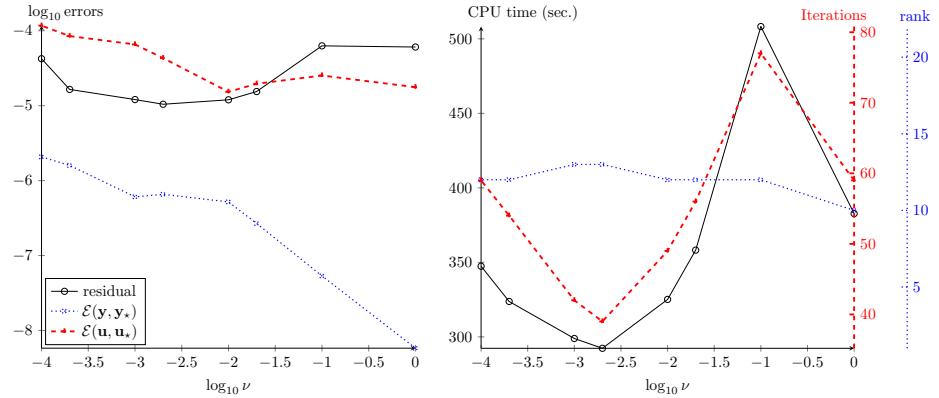

FIG. 10. *Vorticity functional, ν varies. Left: Residual and errors with respect to the reference solutions. Right: CPU time, total number of local iterations, rank.*

The velocity error is smaller than in the tracking optimization case. First, Runge's rule according to Table 3 gives a larger offset:

$$\log_2 h = \frac{\log_2 \varepsilon + 3.51}{1.34} \quad \text{and} \quad \log_2 n = \frac{\log_2 \varepsilon + 0.474}{-1.07}.$$

The total error in the velocity in Figure 7 demonstrates a smaller error scale: between 10^{-3} on average, compared to 10^{-2} in the previous experiment. The control errors and the residuals follow the same trends.

Comparing Tables 2 and 3, we conclude that the optimized vorticity is indeed smaller than the vorticity of the Stokes solution. To see how the vorticity minimization influences the behavior of the fluid, we show the streamline plots in Figure 11. In the left planes we show the snapshots of the solution of the forward Navier–Stokes equations with the default parameters, taken at $t = 12$ and $t = 200$, and in the right planes we show the solution of the optimal control problem at the same time steps. We see that the flow becomes much less turbulent when the control is employed.

6. Conclusions. We have shown in this paper that a low-rank approach for solving the optimal control problem subject to the Navier–Stokes equations is pos-

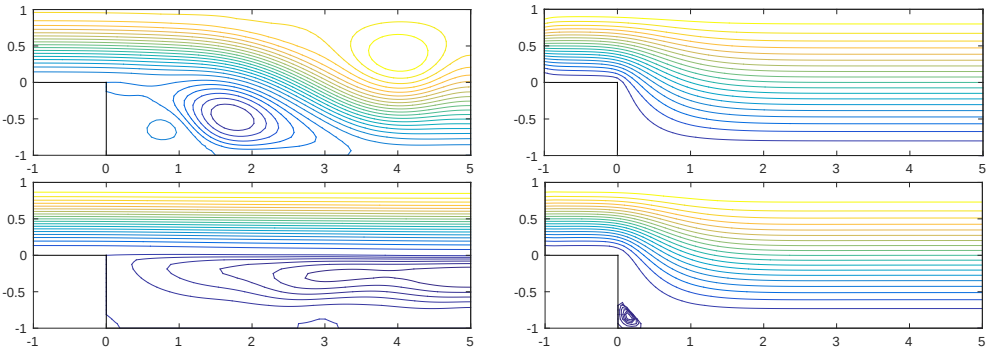

FIG. 11. *Vorticity plots at $t = 12$ (top) and $t = 200$ (bottom) for the forward (uncontrolled) Navier–Stokes system (left), and the controlled system with the minimized vorticity (right).*

sible. In order to achieve this we have established the low-rank formulation for two different objective functions and then introduced a scheme that utilizes the low-rank nature of the desired state to carry this low rank through an alternating iteration procedure. For this we had to rely on efficient low-rank techniques in combination with sophisticated spatial preconditioners for Navier–Stokes systems. We further establish existence results for the low-rank solutions to the Stokes equations. In our numerical results we have performed a parameter study with respect to the convergence of our proposed scheme. We have shown that our method is robust with respect to parameter changes while maintaining a consistent low rank for even large-scale setups.

Acknowledgment. This work began when S. Dolgov was with MPI Magdeburg.

REFERENCES

[1] A. Ammar, B. Mokdad, F. Chinesta, and R. Keunings, *A new family of solvers for some classes of multidimensional partial differential equations encountered in kinetic theory modeling of complex fluids*, J. Non-Newtonian Fluid Mech., 139 (2006), pp. 153–176, https://doi.org/10.1016/j.jnnfm.2006.07.007.

[2] M. J. Balajewicz, E. H. Dowell, and B. R. Noack, *Low-dimensional modelling of high-Reynolds-number shear flows incorporating constraints from the Navier-Stokes equation*, J. Fluid Mech., 729 (2013), pp. 285–308, https://doi.org/10.1017/jfm.2013.278.

[3] J. Ballani and L. Grasedyck, *A projection method to solve linear systems in tensor format*, Numer. Linear Algebra Appl., 20 (2013), pp. 27–43, https://doi.org/10.1002/nla.1818.

[4] P. Benner and T. Breiten, *Low rank methods for a class of generalized Lyapunov equations and related issues*, Numer. Math., 124 (2013), pp. 441–470, https://doi.org/10.1007/s00211-013-0521-0.

[5] P. Benner, S. Dolgov, A. Onwunta, and M. Stoll, *Low-rank solvers for unsteady Stokes-Brinkman optimal control problem with random data*, Comput. Methods Appl. Mech. Engrg., 304 (2016), pp. 26–54, https://doi.org/10.1016/j.cma.2016.02.004.

[6] P. Benner, J.-R. Li, and T. Penzl, *Numerical solution of large-scale Lyapunov equations, Riccati equations, and linear-quadratic optimal control problems*, Numer. Linear Algebra Appl., 15 (2008), pp. 755–777, https://doi.org/10.1002/nla.622.

[7] M. Benzi, G. H. Golub, and J. Liesen, *Numerical solution of saddle point problems*, Acta Numer., 14 (2005), pp. 1–137.

[8] L. T. Biegler, O. Ghattas, M. Heinkenschloss, and B. van Bloemen Waanders, *Large-scale PDE-constrained optimization: An introduction*, in Large-Scale PDE-Constrained Optimization, Lect. Notes Comput. Sci. Eng. 30, Springer, Berlin, 2003, pp. 3–13.

[9] A. Borzì and R. Griesse, *Experiences with a space-time multigrid method for the optimal control of a chemical turbulence model*, Internat. J. Numer. Methods Fluids, 47 (2005), pp. 879–885, https://doi.org/10.1002/fld.904.

[10] A. Borzì and V. Schulz, *Multigrid methods for PDE optimization.*, SIAM Rev., 51 (2009), pp. 361–395, https://doi.org/10.1137/060671590.

[11] J. Burkardt, M. Gunzburger, and H.-C. Lee, *POD and CVT-based reduced-order modeling of Navier-Stokes flows*, Comput. Methods Appl. Mech. Engrg., 196 (2006), pp. 337–355, https://doi.org/10.1016/j.cma.2006.04.004.

[12] S. V. Dolgov and D. V. Savostyanov, *Alternating minimal energy methods for linear systems in higher dimensions*, SIAM J Sci. Comput., 36 (2014), pp. A2248–A2271, https://doi.org/10.1137/140953289.

[13] S. V. Dolgov, A. P. Smirnov, and E. E. Tyrtyshnikov, *Low-rank approximation in the numerical modeling of the Farley-Buneman instability in ionospheric plasma*, J. Comput. Phys., 263 (2014), pp. 268–282, https://doi.org/10.1016/j.jcp.2014.01.029.

[14] H. C. Elman, A. Ramage, and D. J. Silvester, *Algorithm* 886: *IFISS, a Matlab toolbox for modelling incompressible flow*, ACM Trans. Math. Software, 33 (2007), 14.

[15] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Finite Elements and Fast Iterative Solvers: With Applications in Incompressible Fluid Dynamics*, Oxford University Press, Oxford, UK, 2014.

[16] L. Grasedyck, *Existence and computation of low Kronecker-rank approximations for large systems in tensor product structure*, Computing, 72 (2004), pp. 247–265.

[17] L. Grasedyck, D. Kressner, and C. Tobler, *A literature survey of low-rank tensor approximation techniques*, GAMM-Mitt., 36 (2013), pp. 53–78, https://doi.org/10.1002/gamm.201310004.

[18] M. D. Gunzburger and S. Manservisi, *Analysis and approximation of the velocity tracking problem for Navier–Stokes flows with distributed control*, SIAM J. Numer. Anal., 37 (2000), pp. 1481–1512, https://doi.org/10.1137/S0036142997329414.

[19] W. Hackbusch, *Tensor Spaces and Numerical Tensor Calculus*, Springer–Verlag, Berlin, 2012.

[20] W. Hackbusch and B. N. Khoromskij, *Low-rank Kronecker-product approximation to multi-dimensional nonlocal operators. Part* II. *HKT representation of certain operators*, Computing, 76 (2006), pp. 203–225, https://doi.org/10.1007/s00607-005-0145-z.

[21] W. Hackbusch and S. Kühn, *A new scheme for the tensor representation*, J. Fourier Anal. Appl., 15 (2009), pp. 706–722, https://doi.org/10.1007/s00041-009-9094-9.

[22] R. Herzog and E. Sachs, *Preconditioned conjugate gradient method for optimal control problems with control and state constraints*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2291–2317, https://doi.org/10.1137/090779127.

[23] M. Hinze, *Optimal and Instantaneous Control of the Instationary Navier-Stokes Equations*, Habilitation, TU Berlin, Berlin, Germany, 2000.

[24] M. Hinze, M. Köster, and S. Turek, *A Hierarchical Space-Time Solver for Distributed Control of the Stokes Equation*, DFG Priority Programme 1253, Preprint SPP1253-16-01, 2008; available online at http://www.am.uni-erlangen.de/home/spp1253/wiki/images/6/63/Preprint-spp1253-16-01.pdf.

[25] M. Hinze, R. Pinnau, M. Ulbrich, and S. Ulbrich, *Optimization with PDE Constraints*, Math. Model. Theory Appl. 23, Springer-Verlag, New York, 2009.

[26] S. Holtz, T. Rohwedder, and R. Schneider, *The alternating linear scheme for tensor optimization in the tensor train format*, SIAM J. Sci. Comput., 34 (2012), pp. A683–A713, https://doi.org/10.1137/100818893.

[27] T. Huckle and K. Waldherr, *Subspace iteration method in terms of matrix product states*, Proc. Appl. Math. Mech., 12 (2012), pp. 641–642, https://doi.org/10.1002/pamm.201210309.

[28] K. Ito and K. Kunisch, *Lagrange Multiplier Approach to Variational Problems and Applications*, Adv. Des. Control 15, SIAM, Philadelphia, 2008, https://doi.org/10.1137/1.9780898718614.

[29] K. Ito, K. Kunisch, V. Schulz, and I. Gherman, *Approximate nullspace iterations for KKT systems*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 1835–1847, https://doi.org/10.1137/080724952.

[30] H. Kasumba and K. Kunisch, *Vortex control in channel flows using translational invariant cost functionals*, Comput. Optim. Appl., 52 (2012), pp. 691–717.

[31] V. Khoromskaia, *Black-box Hartree–Fock solver by tensor numerical methods*, Comput. Methods Appl. Math., 14 (2014), pp. 89–111, https://doi.org/10.1515/cmam-2013-0023.

[32] B. N. Khoromskij, *Structured rank-*$(r_1, \ldots, r_d)$ *decomposition of function-related operators in* $\mathbb{R}^d$, Comput. Methods Appl. Math, 6 (2006), pp. 194–220.

[33] B. N. Khoromskij, $O(d \log N)$*-quantics approximation of* N*-*d *tensors in high-dimensional numerical modeling*, Constr. Approx., 34 (2011), pp. 257–280, https://doi.org/10.1007/s00365-011-9131-1.

[34] B. N. Khoromskij, *Tensor numerical methods for multidimensional PDEs: Theoretical analysis and initial applications*, ESAIM Proc., 48 (2015), pp. 1–28, https://doi.org/10.1051/proc/201448001.
[35] K. Kormann, *A semi-Lagrangian Vlasov solver in tensor train format*, SIAM J. Sci. Comput., 37 (2015), pp. B613–B632, https://doi.org/10.1137/140971270.
[36] D. Kressner and C. Tobler, *Krylov subspace methods for linear systems with tensor product structure*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 1688–1714, https://doi.org/10.1137/090756843.
[37] P. Kroonenberg and J. de Leeuw, *Principal component analysis of three-mode data by means of alternating least squares algorithms*, Psychometrika, 45 (1980), pp. 69–97.
[38] K. Kunisch and B. Vexler, *Optimal vortex reduction for instationary flows based on translation invariant cost functionals*, SIAM J. Control Optim., 46 (2007), pp. 1368–1397, https://doi.org/10.1137/050632774.
[39] K. Kunisch and S. Volkwein, *Galerkin POD methods for parabolic problems*, Numer. Math., 90 (2001), pp. 117–148.
[40] T. Mach and J. Saak, *Towards an ADI Iteration for Tensor Structured Equations*, MPI Magdeburg Preprint 2011-12, Max Planck Institute for Dynamics of Complex Technical Systems, Magdeburg, Germany, 2011; available online at http://www.de.mpi-magdeburg.mpg.de/preprints/2011/MPIMD11-12.pdf.
[41] K. Mardal and R. Winther, *Preconditioning discretizations of systems of partial differential equations*, Numer. Linear Algebra Appl., 18 (2011), pp. 1–40.
[42] B. R. Noack, P. Papas, and P. A. Monkewitz, *The need for a pressure-term representation in empirical Galerkin models of incompressible shear flows*, J. Fluid Mech., 523 (2005), pp. 339–365, https://doi.org/10.1017/S0022112004002149.
[43] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer Ser. Oper. Res., Springer-Verlag, New York, 1999.
[44] A. Nouy, *Proper generalized decompositions and separated representations for the numerical solution of high dimensional stochastic problems*, Arch. Comput. Methods Eng., 17 (2010), pp. 403–434, https://doi.org/10.1007/s11831-010-9054-1.
[45] I. V. Oseledets, *Approximation of $2^d \times 2^d$ matrices using tensor decomposition*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2130–2145, https://doi.org/10.1137/090757861.
[46] I. V. Oseledets, *Tensor-train decomposition*, SIAM J. Sci. Comput., 33 (2011), pp. 2295–2317, https://doi.org/10.1137/090752286.
[47] I. V. Oseledets, S. Dolgov, V. Kazeev, D. Savostyanov, O. Lebedeva, P. Zhlobich, T. Mach, and L. Song, *TT-Toolbox*, https://github.com/oseledets/TT-Toolbox.
[48] C. C. Paige and M. A. Saunders, *Solutions of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629, https://doi.org/10.1137/0712047.
[49] J. W. Pearson, *Preconditioned iterative methods for Navier–Stokes control problems*, J. Comput. Phys., 292 (2015), pp. 194–207, https://doi.org/10.1016/j.jcp.2015.03.029.
[50] J. W. Pearson, M. Stoll, and A. J. Wathen, *Regularization-robust preconditioners for time-dependent PDE-constrained optimization problems*, SIAM J. Matrix Anal. Appl, 33 (2012), pp. 1126–1152, https://doi.org/10.1137/110847949.
[51] M. V. Rakhuba and I. V. Oseledets, *Grid-based electronic structure calculations: The tensor decomposition approach*, J. Comput. Phys., 312 (2016), pp. 19–30, https://doi.org/10.1016/j.jcp.2016.02.023.
[52] T. Rohwedder and A. Uschmajew, *On local convergence of alternating schemes for optimization of convex problems in the tensor train format*, SIAM J. Numer. Anal., 51 (2013), pp. 1134–1162, https://doi.org/10.1137/110857520.
[53] M. Stoll and T. Breiten, *A low-rank in time approach to PDE-constrained optimization*, SIAM J. Sci. Comput., 37 (2015), pp. B1–B29, https://doi.org/10.1137/130926365.
[54] M. Stoll and A. Wathen, *All-at-once solution of time-dependent Stokes control*, J. Comput. Phys., 232 (2013), pp. 498–515.
[55] U. Tautenhahn and Q.-n. Jin, *Tikhonov regularization and a posteriori rules for solving nonlinear ill posed problems*, Inverse Problems, 19 (2003), pp. 1–21.
[56] V. Temlyakov, *Greedy Approximation*, Cambridge University Press, Cambridge, UK, 2011.
[57] F. Tröltzsch, *Optimal Control of Partial Differential Equations: Theory, Methods and Applications*, American Mathematical Society, Providence, RI, 2010.
[58] M. Ulbrich, *Semismooth Newton Methods for Variational Inequalities and Constrained Optimization Problems in Function Spaces*, SIAM, Philadelphia, 2011, https://doi.org/10.1137/1.9781611970692.
[59] E. L. Wachspress, *The ADI Model Problem*, Springer, New York, 2013.
[60] S. R. White, *Density-matrix algorithms for quantum renormalization groups*, Phys. Rev. B, 48 (1993), pp. 10345–10356, https://doi.org/10.1103/PhysRevB.48.10345.