SIAM J. Sci. Comput.
Vol. 37, No. 2, pp. A832–A858

RICCATI-BASED BOUNDARY FEEDBACK STABILIZATION OF INCOMPRESSIBLE NAVIER–STOKES FLOWS[*]

EBERHARD BÄNSCH[†], PETER BENNER[‡], JENS SAAK[‡], AND HEIKO K. WEICHELT[§]

Abstract. In this article a boundary feedback stabilization approach for incompressible Navier–Stokes flows is studied. One of the main difficulties encountered is the fact that after space discretization by a mixed finite element method (because of the solenoidal condition) one ends up with a differential algebraic system of index 2. The remedy here is to use a discrete realization of the *Leray projection* used by Raymond [J.-P. Raymond, *SIAM J. Control Optim.*, 45 (2006), pp. 790–828] to analyze and stabilize the continuous problem. Using the discrete projection, a linear quadratic regulator (LQR) approach can be applied to stabilize the (discrete) linearized flow field with respect to small perturbations from a stationary trajectory. We provide a novel argument that the discrete Leray projector is nothing else but the numerical projection method proposed by Heinkenschloss and colleagues in [M. Heinkenschloss, D. C. Sorensen, and K. Sun, *SIAM J. Sci. Comput.*, 30 (2008), pp. 1038–1063]. The nested iteration resulting from applying this approach within the Newton-ADI method to solve the LQR algebraic Riccati equation is the key to compute a feedback matrix that in turn can be applied within a closed-loop simulation. Numerical examples for various parameters influencing the different levels of the nested iteration are given. Finally, the stabilizing property of the computed feedback matrix is demonstrated using the *von Kármán vortex street* within a finite element based flow solver.

Key words. flow control, Navier–Stokes equations, Riccati-based feedback

AMS subject classifications. 93D15, 49M15, 76D55

DOI. 10.1137/140980016

1. Introduction.

Incompressible flow problems are encountered in many technical fields such as chemical engineering, biological research, or microfluids in micro- and nanosystems. These flow fields often deal with systems with moderate Reynolds numbers that do not require turbulence models. Many applications require a stable and controlled velocity field that is the basis for an ongoing reaction or production process. The widely used open-loop control approach [31] is not stable regarding small perturbations, which notoriously occur in real-life processes. Distributed control that basically would require the interaction of the controller on every point of the flow field (or in parts of it) is often impractical. In contrast, the approach of boundary feedback stabilization avoids both problems. Although the feedback stabilization cannot manipulate flow fields in their general behavior, it can be used to stabilize existing open-loop controllers. This approach makes the open-loop controllers robust with respect to occurring perturbations. In [39, 40, 41] Raymond established the func-

[*]Submitted to the journal's Methods and Algorithms for Scientific Computing section September 2, 2014; accepted for publication (in revised form) January 21, 2015; published electronically April 2, 2015.
http://www.siam.org/journals/sisc/37-2/98001.html

[†]Research Group Applied Mathematics III (AMIII), Friedrich-Alexander-Universität Erlangen-Nürnberg, Cauerstr. 11, 91058 Erlangen, Germany (baensch@math.fau.de).

[‡]Research Group Mathematics in Industry and Technology (MiIT), Technische Universität Chemnitz, Reichenhainer Str. 39/41, 09126 Chemnitz, Germany, and Research Group Computational Methods in Systems and Control Theory (CSC), Max Planck Institute for Dynamics of Complex Technical Systems Magdeburg, Sandtorstr. 1, 39106 Magdeburg, Germany (benner@mpi-magdeburg.mpg.de, saak@mpi-magdeburg.mpg.de).

[§]Research Group Computational Methods in Systems and Control Theory (CSC), Max Planck Institute for Dynamics of Complex Technical Systems Magdeburg, Sandtorstr. 1, 39106 Magdeburg, Germany (weichelt@mpi-magdeburg.mpg.de).

tional analytical setting and results for a *linear quadratic regulator* (LQR) approach; see, e.g., [17, 35]. With this approach Raymond was able to stabilize Stokes and Navier–Stokes equations regarding small perturbations from a stationary trajectory. Raymond's approach differs from most other ones (and consequently requires a much more involved analysis) in that the feedback contributes *via the boundary* (in contrast to distributed control), which is the natural procedure in most technical applications.

The main difficulty for the feedback stabilization of flow problems is the solenoidal condition, i.e., the local mass conservation. To this end, Raymond uses the *Leray projector* to project the whole system onto the space of divergence-free velocities in L^2 with vanishing normal components (in a weak sense). Using this projection, one ends up with an evolution equation, where well-known techniques for Riccati-based feedback stabilization can be applied.

The Riccati-based boundary feedback stabilization of incompressible Stokes flows has been treated in [15]. There, the major interest was the efficient solution of the arising large-scale saddle point systems. In the present article, these techniques and ideas are expanded to the more general case of the Navier–Stokes equations and numerical results are shown. Most of the formulations in [15] can be extended straightforwardly to the Navier–Stokes case as well. We will upgrade some ideas and show more efficient realizations in the numerical treatment. Furthermore, we will describe the extension of the finite element flow solver NAVIER [6] regarding a closed-loop simulation.

In what follows, the incompressible Navier–Stokes equations are considered, reading in dimensionless form

$$\frac{\partial}{\partial t}\vec{v}(t,\vec{x}) - \frac{1}{\mathrm{Re}}\Delta\vec{v}(t,\vec{x}) + (\vec{v}(t,\vec{x})\cdot\nabla)\vec{v}(t,\vec{x}) + \nabla\chi(t,\vec{x}) = \vec{f}(t,\vec{x})_{ext}, \tag{1.1a}$$

$$\operatorname{div}\vec{v}(t,\vec{x}) = 0, \tag{1.1b}$$

where both the velocity field, denoted by $\vec{v}(t,\vec{x}) = \begin{bmatrix} v_{x_1}(t,\vec{x}) & v_{x_2}(t,\vec{x})\end{bmatrix}^T \in \mathbb{R}^2$, and the pressure, denoted by $\chi(t,\vec{x}) \in \mathbb{R}$, are defined for $t \in [0,\infty)$ and $\vec{x} = \begin{bmatrix} x_1 & x_2\end{bmatrix}^T \in \Omega \subset \mathbb{R}^2$. Here, Ω is a bounded domain with boundary $\Gamma = \partial\Omega$. The Reynolds number $\mathrm{Re} \in \mathbb{R}^+$ describes the ratio of inertial and viscous forces within the fluid and is defined as

$$\mathrm{Re} := \frac{\rho \cdot v_{ref} \cdot d_{ref}}{\eta} \tag{1.2}$$

with the fluid density $\rho \in \mathbb{R}^+$, the fluid viscosity $\eta \in \mathbb{R}^+$, the reference velocity $v_{ref} \in \mathbb{R}^+$, and the reference length $d_{ref} \in \mathbb{R}^+$. Considering an inflow-outflow problem, the boundary can be partitioned as $\Gamma = \Gamma_{in} \cup \Gamma_{out} \cup \Gamma_{wall} \cup \Gamma_{feed}$. Thus, on the respective parts of the boundary Dirichlet boundary conditions are imposed

$$\vec{v}(t,\vec{x}) = \begin{cases} \vec{g}_{feed}(t,\vec{x}) & \text{on } \Gamma_{feed}, \\ \vec{g}_{in}(\vec{x}) & \text{on } \Gamma_{in}, \\ 0 & \text{on } \Gamma_{wall}. \end{cases} \tag{1.3a}$$

The control variable $\vec{g}_{feed}$ realizes the boundary control and will be explained in more detail in section 2.4. As outflow condition, the so-called do-nothing condition [19, 30] is assumed:

$$-\frac{1}{\mathrm{Re}}\nabla\vec{v}(t,\vec{x})\,\vec{n}(\vec{x}) + \chi(t,\vec{x})\vec{n}(\vec{x}) = 0 \quad \text{on } \Gamma_{out} \tag{1.3b}$$

with $\vec{n}(\vec{x})$ the outward normal to Γ_{out}. Notice that (1.3b) ensures that the correct physical behavior does not significantly depend on the length of the channel, as shown in [30]. Finally, the initial condition

$$\vec{v}(0,\cdot) = 0 \quad \text{in } \Omega \tag{1.3c}$$

is prescribed. Hereafter the arguments $t, \vec{x}$ are skipped for better readability.

The convection of the velocity field is a nonlinear operator defined as

$$(\vec{v}\cdot\nabla)\vec{v} = \begin{bmatrix} v_{x_1}\dfrac{\partial v_{x_1}}{\partial x_1} + v_{x_2}\dfrac{\partial v_{x_1}}{\partial x_2} \\ v_{x_1}\dfrac{\partial v_{x_2}}{\partial x_1} + v_{x_2}\dfrac{\partial v_{x_2}}{\partial x_2} \end{bmatrix} \in \mathbb{R}^2;$$

see, e.g., [23].

The main idea of feedback stabilization is to stabilize stationary, but possibly unstable, solutions of the flow field. Such solutions may result from an open-loop control problem [31]. Usually, these solutions are not robust with respect to small perturbations. The feedback controller measures the deviation that occurs due to perturbations and pushes the system back to the desired stationary trajectory. If these perturbations are small enough, one can show that the Riccati-based feedback stabilization is able to exponentially stabilize unstable solution trajectories, as described in, e.g., [7]. To this end, an LQR approach is used and the unique stabilizing solution of the LQR problem is determined by solving the corresponding Riccati equation; see, e.g., [17, 35].

The rest of this paper is organized as follows. The complete work flow to compute the feedback is described in the next section. In section 3 some special properties of the arising nested iteration are discussed, and the usability of the resulting algorithm is illustrated by numerical experiments in section 4. Section 5 concludes the paper and gives an outlook on open problems and future investigations.

2. Riccati-based feedback stabilization.

We follow the analytic approaches for boundary feedback stabilization by Raymond [40] and seek a numerical realization of those. Raymond applies the Leray projector that projects the velocity field onto the space of divergence-free functions in L^2 to the linearized flow problem; as a result, (1.1b) is fulfilled automatically. This means that one can apply an LQR approach to the projected evolution equation. Raymond shows that one can find a stabilizing feedback that also stabilizes the nonlinear system (1.1) via boundary control influence; see, e.g, [40]. In [15], the numerical realization of this approach is shown for the Stokes equation. In contrast to the Stokes case, the Navier–Stokes equations are nonlinear. The linearization approach is explained in the next subsection.

2.1. Linearization of the NSE.

Before applying the LQR approach to the (nonlinear) Navier–Stokes equations (1.1), consider $(\vec{w}(\vec{x}), \chi_s(\vec{x}))$ as a velocity and pressure pair that fulfills the stationary Navier–Stokes equations

$$-\frac{1}{\mathrm{Re}}\Delta\vec{w} + (\vec{w}\cdot\nabla)\vec{w} + \nabla\chi_s = \vec{f}_{ext}, \tag{2.1a}$$

$$\operatorname{div}\vec{w} = 0, \tag{2.1b}$$

defined in Ω with the same boundary conditions as in (1.3a)–(1.3b) and $\vec{g}_{feed} = 0$. The pair $(\vec{w}, \chi_s)$ represents the desired stationary, although (possibly) unstable, solution of (1.1) that should be stabilized against perturbations. The goal of the LQR approach is to force the solution $(\vec{v}, \chi)$ of (1.1) toward $(\vec{w}, \chi_s)$. That implies that the difference state $\vec{z}(t, \vec{x}) := \vec{v}(t, \vec{x}) - \vec{w}(\vec{x})$ is asymptotically stabilized and $p(t, \vec{x}) := \chi(t, \vec{x}) - \chi_s(\vec{x})$ up to a constant, respectively. This yields the linearized Navier–Stokes equations

$$\frac{\partial}{\partial t}\vec{z} - \frac{1}{\mathrm{Re}}\Delta\vec{z} + (\vec{w}\cdot\nabla)\vec{z} + (\vec{z}\cdot\nabla)\vec{w} + \nabla p = 0, \tag{2.2a}$$

$$\operatorname{div}\vec{z} = 0, \tag{2.2b}$$

defined for $t \in [0, \infty)$ and $\vec{x} \in \Omega \subset \mathbb{R}^2$ with Dirichlet boundary conditions

$$\vec{z}(t, \vec{x}) = \begin{cases} \vec{g}_{feed}(t, \vec{x}) & \text{on } \Gamma_{feed}, \\ 0 & \text{on } \Gamma_{in} \cup \Gamma_{wall}, \end{cases} \tag{2.3a}$$

the *do-nothing* condition

$$-\frac{1}{\mathrm{Re}}\nabla\vec{z}(t, \vec{x})\,\vec{n}(\vec{x}) + p(t, \vec{x})\vec{n}(\vec{x}) = 0 \quad \text{on } \Gamma_{out}, \tag{2.3b}$$

and the initial condition

$$\vec{z}(0, \cdot) = 0 \quad \text{in } \Omega. \tag{2.3c}$$

The present LQR approach is based on finite dimensional matrix equations. To derive those equations, the linearized Navier–Stokes equations (2.2) are discretized in space with the boundary and initial conditions (2.3) and the finite dimensional LQR approach is introduced in detail in the next subsections.

2.2. Discretization. A common way of discretizing instationary control problems [3, 4, 20, 29] is the *method of lines* [45]. In our case a mixed finite element method [32] is used to discretize the linearized Navier–Stokes equations (2.2) in space, leaving the time variable untouched. This yields the system of differential-algebraic equations:

$$M\frac{d}{dt}\mathbf{z}(t) = A\mathbf{z}(t) + G\mathbf{p}(t) + \mathbf{f}(t), \tag{2.4a}$$

$$0 = G^T\mathbf{z}(t), \tag{2.4b}$$

where $\mathbf{z}(t) \in \mathbb{R}^{n_v}$ denotes the nodal vector of the discretized velocity, $\mathbf{p}(t) \in \mathbb{R}^{n_p}$ denotes the discretized pressure, and $\mathbf{f}(t) \in \mathbb{R}^{n_v}$ contains the boundary control influence. We substitute $\mathbf{f}(t) = B\mathbf{u}(t)$ by the input operator B and the optimal control $\mathbf{u}(t)$ that will be described in more detail in subsection 2.4. Furthermore, denote by $M = M^T \succ 0 \in \mathbb{R}^{n_v \times n_v}$ the mass matrix, by $A \in \mathbb{R}^{n_v \times n_v}$ the system matrix [23, section 7.3], and by $G \in \mathbb{R}^{n_v \times n_p}$ the discretized gradient. The system matrix can be split as follows:

$$A = -\frac{1}{\mathrm{Re}}S - K - R$$

with $-S\mathbf{z}$ the discrete counterpart of the Laplacian $\Delta\vec{z}$, $-K\mathbf{z}$ the discrete convection resulting from $(\vec{w}\cdot\nabla)\vec{z}$, and $-R\mathbf{z}$ a discrete reaction process associated to $(\vec{z}\cdot\nabla)\vec{w}$. Notice that the system matrix and the gradient in the setting for the system theoretic framework are usually written on the right-hand side of the equation, which leads to a switch of the sign in contrast to the convention in the finite element community.

Furthermore, it is assumed that the velocity can be observed only in parts of the domain and, therefore, the output equation

$$\mathbf{y}(t) = C\mathbf{z}(t) \tag{2.4c}$$

is added with the output $\mathbf{y}(t) \in \mathbb{R}^{n_a}$ and the output operator $C \in \mathbb{R}^{n_a \times n_v}$ that measures the behavior of the velocity by using the information of the inner nodes. The particular properties of C are discussed in section 4.5.

Using inf-sup stable finite elements (like $\mathcal{P}_2 - \mathcal{P}_1$ *Taylor-Hood* finite elements [32] in our case) for an inflow-outflow problem, it holds that $n_v > n_p$ and the discretized gradient G is of rank n_p [23, section 5.3].

Considering the structure of (2.4a)–(2.4b), one gets a DAE of differential index 2 [48] defined by the matrix pencil

$$\left(\underbrace{\begin{bmatrix} A & G \\ G^T & 0 \end{bmatrix}}_{=:\mathbf{A}}, \underbrace{\begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix}}_{=:\mathbf{M}} \right). \tag{2.5}$$

Here, the right-hand side coefficient matrix $\mathbf{M}$ is singular and the matrix pencil has $n_v - n_p$ finite eigenvalues $\lambda_i \in \mathbb{C} \setminus \{0\}$ and $2n_p$ infinite eigenvalues $\lambda_\infty = \infty$ [21, Theorem 2.1].

DAEs are more involved than ordinary differential equations, since the solution set of the DAE lies on a (usually hidden) manifold. In the case of incompressible flows this manifold is the space of (discretely) divergence-free functions. We use the index reduction idea described in [29, section 3] to avoid this problem and demonstrate the feasibility of this idea for descriptor systems like (2.4) in the next subsection.

Notice that the whole system (2.4) is a descriptor system with multiple inputs (n_r) (see section 2.4 below) and multiple outputs (n_a) (in short, a MIMO system). It is well known that such systems are getting more complicated if the number of inputs n_r or outputs n_a gets larger.

After defining all components in the descriptor system (2.4), an index reduction technique that is a numerical realization of the Leray projection is presented in the next subsection.

2.3. Projection method. In [7], Bänsch and Benner proposed to use the index reduction method defined in [29] as numerical realization of the Leray projection in the case of linearized Navier–Stokes equations. In [15] it is shown that the projector

$$\mathit{\Pi}^T := I_{n_v} - M^{-1}G(G^T M^{-1} G)^{-1} G^T \tag{2.6}$$

used in [29] turns out to be nothing else but the discrete Leray operator (as needed for our LQR approach) in our setting. To this end, it is necessary to view $\mathit{\Pi}^T$ as an orthogonal projector in the appropriate setting, i.e., the discrete L^2 inner product, in contrast to the interpretation of $\mathit{\Pi}^T$ as oblique projection in the Euclidean inner product as in [29]. In this section, the proof of this fact is recalled. Notice that one can use the one-sided projection using Π^T from (2.6) since in the Galerkin FEM the discretized gradient operator on the pressure space and the discrete divergence operator are transposed to each other. For a more general discretization that does not provide this feature, however, the basic projection idea stays valid, but requires using the more general two-sided projection approach discussed in the interpolatory model reduction framework in [27, section 6].

Following the description in [22, 25], $L^2(\Omega)^2$ can be decomposed into

$$L^2(\Omega)^2 = \mathbf{H}(\mathrm{div},0) \perp \mathbf{H}(\mathrm{div},0)^{\perp},$$

where the spaces $\mathbf{H}(\mathrm{div},0)$ and $\mathbf{H}(\mathrm{div},0)^{\perp}$ are characterized by

$$\mathbf{H}(\mathrm{div},0) := \{\vec{v} \in L^2(\Omega)^2 \mid \mathrm{div}\,\vec{v} = 0, \vec{v}\cdot\vec{n}_{|\Gamma} = 0\}, \quad \mathbf{H}(\mathrm{div},0)^{\perp} := \{\nabla p \mid p \in H^1(\Omega)\}.$$

This splitting is equivalent to $\vec{v} = \vec{v}_{\mathrm{div}} + \nabla p$, where $\vec{v}_{\mathrm{div}}$ and p fulfill

$$\begin{aligned} \vec{v}_{\mathrm{div}} + \nabla p &= \vec{v} && \text{in } \Omega,\\ \mathrm{div}\,\vec{v}_{\mathrm{div}} &= 0 && \text{in } \Omega,\\ \vec{v}_{\mathrm{div}}\cdot\vec{n} &= 0 && \text{on } \Gamma. \end{aligned}$$

The operator $P : L^2(\Omega)^2 \to \mathbf{H}(\mathrm{div},0)$ with $P : \vec{v} \mapsto \vec{v}_{\mathrm{div}}$ is called the Leray projection.

The discrete version of the above splitting for the finite element discretization can be written in terms of nodal value vectors as

$$M\mathbf{v}_{\mathrm{div}} + G\mathbf{p} = M\mathbf{v}, \tag{2.7a}$$

$$G^T\mathbf{v}_{\mathrm{div}} = 0 \tag{2.7b}$$

with the discrete velocity field $\mathbf{v} \in \mathbb{R}^{n_v}$, the discretely divergence-free velocity field $\mathbf{v}_{\mathrm{div}} \in \mathbb{R}^{n_v}$, and the discrete pressure $\mathbf{p} \in \mathbb{R}^{n_p}$. M, G are defined as above. Multiplying (2.7a) from the left by G^TM^{-1} and using (2.7b) reveals

$$\mathbf{p} = (G^TM^{-1}G)^{-1}G^T\mathbf{v}.$$

Using this expression for $\mathbf{p}$ in (2.7a) yields

$$\mathbf{v}_{\mathrm{div}} = (I_{n_v} - M^{-1}G(G^TM^{-1}G)^{-1}G^T)\mathbf{v} \quad \Leftrightarrow \quad \mathbf{v}_{\mathrm{div}} = \mathit{\Pi}^T\mathbf{v}.$$

This shows, together with the properties described in [15], that multiplication of a discretized velocity field from the left by $\mathit{\Pi}^T$ is equivalent to applying the discretized Leray projection to a discretized vector function. Of course, building $\mathit{\Pi}^T$ explicitly is prohibitive, since the matrix would become dense. A numerically suitable way to apply $\mathit{\Pi}^T$ is to write the system (2.7) as saddle point problem

$$\begin{bmatrix} M & G \\ G^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{v}_{\mathrm{div}} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} M\mathbf{v} \\ 0 \end{bmatrix}$$

that has to be solved for a given velocity field $\mathbf{v}$ to get the (discretely) divergence-free velocity field $\mathbf{v}_{\mathrm{div}}$. Nevertheless, we want to avoid the projection in general in the process of computing the optimal control $\mathbf{u}(t)$, as shown next.

Following the projection steps in [15, section 2.1] with the identical notation together with the substitution $\mathit{\Pi}^T\mathbf{z} = \mathbf{z}$ and the projector decomposition $\mathit{\Pi} = \Theta_l\Theta_r^T$, the descriptor system (2.4) can be written in terms of $\tilde{\mathbf{z}} = \Theta_l^T\mathbf{z} \in \mathbb{R}^{n_v-n_p}$ as the generalized state space system

$$\mathcal{M}\frac{d}{dt}\tilde{\mathbf{z}}(t) = \mathcal{A}\tilde{\mathbf{z}}(t) + \mathcal{B}\mathbf{u}(t), \tag{2.8a}$$

$$\mathbf{y}(t) = \mathcal{C}\tilde{\mathbf{z}}(t) \tag{2.8b}$$

with $\mathcal{M} = \mathcal{M}^T \succ 0 \in \mathbb{R}^{(n_v-n_p)\times(n_v-n_p)}$ that fits the scheme of [29, equation (4.1)].

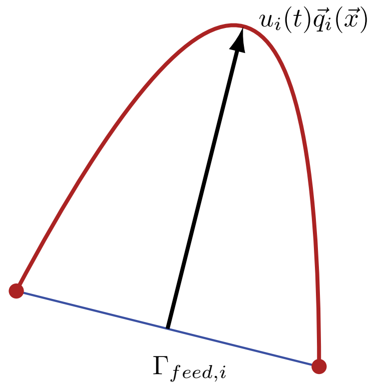


FIG. 1. *The parabolic inflow profile at the control boundary part* $\Gamma_{feed,i}$.

2.4. Boundary control. In this section the incorporation of *normal* boundary control for system (2.8) is described. Notice that $\tilde{\mathbf{z}}$ has zero normal components in a discrete sense.

We mimic the approach in [42], where an operator was constructed that distributed the boundary control into the interior of Ω. This construction was necessary, since Raymond worked with the space $\mathbf{H}(\text{div}, 0)$. Notice that functions in $\mathbf{H}(\text{div}, 0)$ have vanishing normal components on the boundary.

Consider n_r different parts of the control boundary Γ_{feed} , i.e.,

$$\vec{g}_{feed}(t, \vec{x}) = \sum_{i=1}^{n_r} u_i(t)\vec{q}_i(\vec{x}),$$

where $\vec{q}_i$ has support on $\Gamma_{feed,i}$ and $\bigcup_{i=1}^{n_r} \Gamma_{feed,i} = \Gamma_{feed}$. In the example in section 4, a parabolic in-/outflow $\vec{q}_i$ is considered and its intensity is controlled by $u_i(t)$; see Figure 1.

The control operator $\mathcal{B} := \Theta_r^T B$ in (2.8) is the projected version of B that has n_r columns, each of which is computed in the following way.

For $i = 1, \ldots, n_r$ *do:*

1. *Solve the linearized Navier–Stokes equations* (2.4) *with homogeneous Dirichlet boundary condition except for* $\vec{g}_{feed}$ *on* $\Gamma_{feed,i}$*, where the Dirichlet condition* $1 \cdot \vec{q}_i(\vec{x})$ *is imposed. Denote the resulting velocity field by* $\mathbf{v}_i$.
2. *Project* $\tilde{\mathbf{v}}_i := \Pi^T \mathbf{v}_i$.
3. *Multiply* $\tilde{\mathbf{v}}_i = \Pi^T \mathbf{v}_i$ *by the discrete linearized Navier–Stokes operator* A *to get* $\hat{\mathbf{v}}_i$.
4. *Set the ith column* $\mathbf{b}_i$ *of* B *to* $\mathbf{b}_i := \Pi^T \hat{\mathbf{v}}_i$.

In the next subsection, the LQR approach for the projected system (2.8) is formulated and an algorithm to compute the optimal control $\mathbf{u}(t)$ is shown. In contrast to the Stokes formulation in [15], the system matrix $\mathcal{A}$ is nonsymmetric. This yields minor changes in the procedure but does not change the work flow.

2.5. The LQR approach. The LQR approach is supposed to minimize a given quadratic cost functional subject to a given linear system as described, e.g., in Locatelli's introduction about LQR in [37]. Because the generalized state space system (2.8) with a mass matrix $\mathcal{M} \neq I$ on the left-hand side is considered, the procedure changes a little bit, as shown in [43, Chapter 5.2].

The cost functional for our problem setting is defined as

$$\mathcal{J}(\tilde{\mathbf{z}}(t), \mathbf{u}(t)) := \frac{1}{2}\int_0^\infty \lambda \left(\tilde{\mathbf{z}}(t)^T \mathcal{C}^T \mathcal{C} \tilde{\mathbf{z}}(t)\right) + \mathbf{u}(t)^T \mathbf{u}(t) \,\mathrm{dt}, \tag{2.9}$$

which measures the output $\mathbf{y} = \mathcal{C}\tilde{\mathbf{z}}$ and the cost of the optimal control $\mathbf{u}$ in the square of the Euclidean norm $||\mathbf{x}||_2 := \sqrt{\mathbf{x}^T\mathbf{x}}$. To minimize this integral over the infinite time horizon, the output, as well as the control, has to converge asymptotically to zero for $t \to \infty$. The output $\mathbf{y}$ measures the velocity field $\vec{z}$ on the discrete level. In subsection 2.1, $\vec{z}$ is defined as perturbation of the flow field $\vec{v}$ from the desired stationary flow field $\vec{w}$. A zero output for $t \to \infty$ implies that $\vec{v}$ approximates $\vec{w}$ for $t \to \infty$; the flow field $\vec{v}$ achieves the properties of the desired stationary flow field.

The factor λ in the first term of (2.9) is used as regularization. By varying λ one may also achieve qualitatively different results. For ease of notation, the value of λ is set to $\lambda = 1$ in the theoretical derivations. In the actual implementation reported in section 3, variations of λ are of course allowed, and the influence of these variations is investigated in the numerical experiments in section 4.

Following the LQR approach, the optimal control

$$\mathbf{u}_*(t) = -\underbrace{\mathcal{B}^T X \mathcal{M}}_{=:\mathcal{K}} \tilde{\mathbf{z}}_*(t) = -\mathcal{K}\tilde{\mathbf{z}}_*(t)$$

minimizes the cost functional (2.9) subject to (2.8) with $\tilde{\mathbf{z}}_*(t)$ as the optimal trajectory and $X = X^T \succeq 0 \in \mathbb{R}^{(n_v - n_p)\times(n_v - n_p)}$ as the unique stabilizing solution of the *generalized algebraic Riccati equation (GARE)*

$$0 = \mathcal{C}^T\mathcal{C} + \mathcal{A}^T X \mathcal{M} + \mathcal{M} X \mathcal{A} - \mathcal{M} X \mathcal{B}\mathcal{B}^T X \mathcal{M} =: \mathfrak{R}(X). \tag{2.10}$$

The key ingredient to compute the feedback $\mathcal{K} \in \mathbb{R}^{n_r \times (n_v - n_p)}$ that asymptotically stabilizes the generalized state space system (2.8) is to solve the GARE (2.10). A solution approach is shown in the next subsection.

2.6. Solving the GARE. The GARE (2.10) is a quadratic matrix equation for the unknown matrix $X \in \mathbb{R}^{(n_v - n_p)\times(n_v - n_p)}$ that can be solved by a Newton-type iteration. At step m, this iteration is given by

$$X^{(m+1)} = X^{(m)} + N^{(m)}$$

with the update $N^{(m)}$ computed via

$$\mathfrak{R}'|_{X^{(m)}}(N^{(m)}) = -\mathfrak{R}(X^{(m)}),$$

$\mathfrak{R}'|_{X^{(m)}}$ being the Frchet derivative of the Riccati operator (2.10) at $X^{(m)}$ defined as

$$\mathfrak{R}'|_{X^{(m)}}: \quad N^{(m)} \mapsto (\mathcal{A} - \mathcal{B}\mathcal{B}^T X^{(m)}\mathcal{M})^T N^{(m)} \mathcal{M} + \mathcal{M} N^{(m)} (\mathcal{A} - \mathcal{B}\mathcal{B}^T X^{(m)} \mathcal{M});$$

see, e.g., [2, 34]. Following the procedure in [15, section 2.3], the main task to compute the iterate $X^{(m+1)}$ in the Newton step $m+1$ is to solve a generalized Lyapunov equation

$$(\mathcal{A}^{(m)})^T X^{(m+1)} \mathcal{M} + \mathcal{M} X^{(m+1)} \mathcal{A}^{(m)} = -(\mathcal{W}^{(m)})^T \mathcal{W}^{(m)}. \tag{2.11}$$

The closed-loop system matrix $\mathcal{A}^{(m)} = \mathcal{A} - \mathcal{B}\mathcal{K}^{(m)}$ includes the previous computed feedback $\mathcal{K}^{(m)} = \mathcal{B}^T X^{(m)} \mathcal{M}$ from step m, and the low-rank right-hand side factor

Algorithm 1. Generalized low-rank Cholesky factor ADI iteration.

```
Input: $\mathcal{A}^{(m)}, \mathcal{M}, \mathcal{W}^{(m)}$, and shift parameters $q_i \in \mathbb{C}^- : i = 1, \dots, i_{\max}$
Output: $Z = Z_{i_{\max}} \in \mathbb{C}^{n \times t_{i_{\max}}}$, such that $ZZ^H \approx X^{(m+1)}$
1: $V_1 = \sqrt{-2\,\mathrm{Re}\,(q_1)}\left((\mathcal{A}^{(m)})^T + q_1\mathcal{M}\right)^{-1}(\mathcal{W}^{(m)})^T$
2: $Z_1 = V_1$
3: for $i = 2, 3, \dots, i_{\max}$ do
4:     $V_i = \sqrt{\mathrm{Re}\,(q_i)\,/\,\mathrm{Re}\,(q_{i-1})}\left(V_{i-1} - (q_i + \overline{q_{i-1}})\left((\mathcal{A}^{(m)})^T + q_i\mathcal{M}\right)^{-1}(\mathcal{M}V_{i-1})\right)$
5:     $Z_i = [Z_{i-1}\, V_i]$
6: end for
```

$(\mathcal{W}^{(m)})^T = \begin{bmatrix}\mathcal{C}^T & (\mathcal{K}^{(m)})^T\end{bmatrix}$; see, e.g., [33]. The linear matrix equation (2.11) has to be solved in every Newton step.

Applying the *low-rank ADI iteration* [13, 36], more precisely, using the extended formulation for the generalized case as described in [10], the iterate $X^{(m+1)}$ in (2.11) is computed. The above notation yields Algorithm 1. Details about the ADI shift parameters $q_i \in \mathbb{C}^-$ are described in section 3.2.

The spectrum of the projected state space system (2.8) is equivalent to the finite spectrum of the DAE (2.5). The major part of this finite spectrum lies in the open left complex half plane $\mathbb{C}^-$, but with increasing Reynolds number a few finite eigenvalues with positive real part occur [1]. Because of this fact, the system is not asymptotically stable [37], which is, however, necessary to guarantee that the solution of the Lyapunov equation yields a stable closed-loop matrix which is necessary for convergence of Newton's method to the desired solution [10]. Therefore, an initial stabilizing feedback K_0 needs to be computed such that the finite spectrum of the closed-loop system

$$\left(\begin{bmatrix} A - BK_0 & G \\ G^T & 0 \end{bmatrix}, \begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix}\right) \tag{2.12}$$

is a subset of $\mathbb{C}^-$, as shown in subsection 2.7.

The combination of the Newton iteration, as an outer iteration, with the G-LRCF-ADI (Algorithm 1), as an inner iteration, is known as the *generalized low-rank Cholesky factor Newton method* [13, 14]. The main computational work is done in lines 1 and 4 of Algorithm 1 by solving linear systems of equations with large-scale and projected matrices of the form

$$\left((\mathcal{A}^{(m)})^T + q_i\mathcal{M}\right)\Lambda = \mathcal{Y}.$$

Solving these linear systems for different right-hand sides $\mathcal{Y}$ would involve the dense decomposition of the dense projection matrix $\mathit{\Pi}^T$ that we want to avoid using the main results of [29, section 5]. There it is shown that $\Lambda = \mathit{\Pi}^T\Lambda$ can be computed as the solution of

$$\mathit{\Pi}\left((A - BK^{(m)})^T + q_iM\right)\mathit{\Pi}^T\Lambda = \mathit{\Pi}Y,$$

which is equivalent to the solution of the saddle point system

$$\underbrace{\begin{bmatrix} (A - BK^{(m)})^T + q_iM & G \\ G^T & 0 \end{bmatrix}}_{=:(\mathbf{A}^{(m)})^T + q_i\mathbf{M} =: \hat{\mathbf{F}}_i}\begin{bmatrix}\Lambda \\ * \end{bmatrix} = \begin{bmatrix} Y \\ 0 \end{bmatrix} \tag{2.13}$$

Algorithm 2. Generalized low-rank Cholesky factor Newton method for Navier–Stokes

```
Input: $M, A, G, B, C$, initial feedback $K_0$, and ADI shift parameters
       $q_i \in \mathbb{C}^- : i = 1, \ldots, n_{\mathrm{ADI}}$, $tol_{\mathrm{ADI}}$, $tol_{\mathrm{Newton}}$, $\lambda$
Output: feedback operator $K$
 1: for $m = 1, 2, \ldots, n_{\mathrm{Newton}}$ do
 2:   $(W^{(m)})^T = \begin{bmatrix} \sqrt{\lambda}\, C^T & (K^{(m-1)})^T \end{bmatrix}$
 3:   Get $V_1$ by solving
        $$\begin{bmatrix} A^T - (K^{(m-1)})^T B^T + q_1 M & G \\ G^T & 0 \end{bmatrix} \begin{bmatrix} V_1 \\ * \end{bmatrix} = \begin{bmatrix} \sqrt{-2\,\mathrm{Re}\,(q_1)}\,(W^{(m)})^T \\ 0 \end{bmatrix}$$
 4:   $K_1^{(m)} = B^T V_1 \overline{V}_1^T M$
 5:   for $i = 2, 3, \ldots, n_{\mathrm{ADI}}$ do
 6:     Get $\tilde{V}$ by solving
          $$\begin{bmatrix} A^T - (K^{(m-1)})^T B^T + q_i M & G \\ G^T & 0 \end{bmatrix} \begin{bmatrix} \tilde{V} \\ * \end{bmatrix} = \begin{bmatrix} M V_{i-1} \\ 0 \end{bmatrix}$$
 7:     $V_i = \sqrt{\mathrm{Re}\,(q_i)\,/\,\mathrm{Re}\,(q_{i-1})}\left(V_{i-1} - (q_i + \overline{q_{i-1}})\tilde{V}\right)$
 8:     $K_i^{(m)} = K_{i-1}^{(m)} + B^T V_i \overline{V}_i^T M$
 9:     if $\left(\frac{\|K_i^{(m)} - K_{i-1}^{(m)}\|_F}{\|K_i^{(m)}\|_F} < tol_{\mathrm{ADI}}\right)$ then
10:       break
11:     end if
12:   end for
13:   $K^{(m)} = K_{n_{\mathrm{ADI}}}^{(m)}$
14:   if $\left(\frac{\|K^{(m)} - K^{(m-1)}\|_F}{\|K^{(m)}\|_F} < tol_{\mathrm{Newton}}\right)$ then
15:     break
16:   end if
17: end for
18: $K = K^{(n_{\mathrm{Newton}})}$
```

with $K^{(m)} = B^T X^{(m)} M$; see [29, Lemma 5.2]. Notice that the Newton step index m is dropped for better readability in all further considerations.

The saddle point system (2.13) consists of the original matrices and does not include any projection. The entire work flow to compute the feedback matrix K via the generalized low-rank Cholesky factor Newton method avoiding the use of any projection is summarized in Algorithm 2.

After a short note about the initial feedback in the next subsection, the important properties of the nested iteration in Algorithm 2 will be discussed in section 3.

2.7. Initial feedback.

As mentioned above, the matrix pencil (2.5) is not stable in general. But in order to use the ADI method for solving the Lyapunov equation (2.11), a stable pencil is needed. Following the ideas in [28], we show how to construct an initial feedback K_0. Notice that this concept is also used in [1] as a numerical method to stabilize the Navier–Stokes equations.

The majority of the $n_v - n_p$ finite eigenvalues of the pencil (2.5) are stable and only n_{us} eigenvalues are unstable with $n_{us} \ll n_v$. First, all unstable finite eigenvalues $\lambda_{us}^{(i)} \in \mathbb{C}^+$ of the pencil (2.5) are needed together with their corresponding left and right eigenvectors $\omega^i, \eta^i \in \mathbb{C}^{n_v+n_p}$ for $i = 1, \dots, n_{us}$. Solving these large-scale generalized eigenvalue problems in an efficient way is not within the scope of this paper. We use the implicitly restarted shift-and-invert Arnoldi method as implemented in the `eigs` function of MATLAB with the matrix shifting strategy described in [21]; using this strategy all infinite eigenvalues are transformed to a fixed finite eigenvalue and the eigenvectors corresponding to the original finite eigenvalues stay unchanged. This technique to calculate all finite and unstable eigenvalues is not the most robust method for unknown problem settings but it is sufficient in our case.

Using the left and right eigenvectors, the following matrices are defined as

$$\begin{aligned} W &:= \left[\omega^{(1)}, \dots, \omega^{(n_{us})}\right] \in \mathbb{C}^{(n_v+n_p)\times n_{us}} \text{ and} \\ H &:= \left[\eta^{(1)}, \dots, \eta^{(n_{us})}\right] \in \mathbb{C}^{(n_v+n_p)\times n_{us}}. \end{aligned}$$

These matrices are used to project the system onto the subspace spanned by the eigenvectors of the unstable eigenvalues λ_{us}:

$$\tilde{\mathbf{M}} := W^* \begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix} H, \quad \tilde{\mathbf{A}} := W^* \begin{bmatrix} A & G \\ G^T & 0 \end{bmatrix} H, \quad \tilde{\mathbf{B}} := W^* \begin{bmatrix} B \\ 0 \end{bmatrix}.$$

After solving the n_{us}-dimensional generalized Bernoulli equation

$$\tilde{\mathbf{A}}^* X_0 \tilde{\mathbf{M}} + \tilde{\mathbf{M}}^* X_0 \tilde{\mathbf{A}} - \tilde{\mathbf{M}}^* X_0 \tilde{\mathbf{B}} \tilde{\mathbf{B}}^* X_0 \tilde{\mathbf{M}} = 0, \tag{2.14}$$

the initial feedback $K_0 \in \mathbb{R}^{n_r \times n_v}$ can be defined as first n_v columns of

$$\begin{bmatrix} K_0 & 0 \end{bmatrix} := \begin{bmatrix} B^T & 0 \end{bmatrix} W X_0 W^T \begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix}.$$

We solve the (generalized) Bernoulli equation (2.14) with an algorithm based on [9]. Using the initial feedback K_0, the resulting closed-loop pencil

$$\left(\begin{bmatrix} A & G \\ G^T & 0 \end{bmatrix} - \begin{bmatrix} B \\ 0 \end{bmatrix} \cdot \begin{bmatrix} K_0 & 0 \end{bmatrix}, \begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix} \right)$$

is stable and all initially unstable eigenvalues $\lambda_{us}^{(i)}$ appear as stabilized eigenvalues

$$\lambda_{stab}^{(i)} := -\operatorname{Re}\left(\lambda_{us}^{(i)}\right) + j \operatorname{Im}\left(\lambda_{us}^{(i)}\right) \in \mathbb{C}^- \ \forall i = 1, \dots, n_{us}$$

that are mirrored at $j\mathbb{R}$ [9] (with j as the imaginary unit $j^2 = -1$).

Notice that some algorithms compute the eigenvectors in such a way that $\tilde{\mathbf{M}} = I$, which means V and W are orthogonal w.r.t. the inner product $(.,.)_{\tilde{\mathbf{M}}}$. For this constellation one has to solve an ordinary Bernoulli equation.

Using the initial feedback K_0, the nested Algorithm 2 can be used to compute the feedback K that defines $\mathbf{u}(t)$ minimizing the cost functional (2.9). Some details of the nested iteration are shown in the next section.

3. Nested iteration.

3. Nested iteration. The iterative process to compute the feedback matrix K is described in Algorithm 2. It is a nested iteration embedded in a Newton type method, as outermost iteration, an ADI iteration, as central iteration, and a method to solve the large-scale and sparse saddle point systems (2.13), as innermost iteration.

For moderate problem sizes of $n = n_v + n_p = O(10^6)$ a direct solver is the best choice to solve the saddle point systems (3.2). However, if the dimension of the underlying finite element discretization gets larger or one deals with three-dimensional problems, direct solvers generate a considerable fill-in such that the use of iterative methods is imperative; see, e.g., [15]. For the problem size of the numerical example in section 4 a direct solver is suitable. Extending the iterative solution techniques of [15] to the more general Navier–Stokes flows is part of our current research.

In the next subsection, some properties of the saddle point matrix $\hat{\mathbf{F}}_i$ in (2.13) are shown and different parameters of the nested iteration are discussed.

3.1. Properties of the saddle point system. Although the matrices A, M, G in $\hat{\mathbf{F}}_i$ are sparse, the low-rank product $K^T B^T$ is a dense block such that nearly the whole matrix $\hat{\mathbf{F}}_i$ becomes dense. To avoid this problem, (2.13) is written in the form of a low-rank update

$$\Bigg(\underbrace{\begin{bmatrix} A^T + q_i M & G \\ G^T & 0 \end{bmatrix}}_{\mathbf{F}_i} - \underbrace{\begin{bmatrix} K^T \\ 0 \end{bmatrix}}_{\mathbf{K}^T} \underbrace{\begin{bmatrix} B^T & 0 \end{bmatrix}}_{\mathbf{B}^T}\Bigg) \underbrace{\begin{bmatrix} \Lambda \\ * \end{bmatrix}}_{\mathbf{\Lambda}} = \underbrace{\begin{bmatrix} Y \\ 0 \end{bmatrix}}_{\mathbf{Y}},$$

that is (in a compact notation),

$$(\mathbf{F}_i - \mathbf{K}^T\mathbf{B}^T)\mathbf{\Lambda} = \mathbf{Y}. \tag{3.1}$$

To evaluate (3.1), [15, section 3.1] is recalled, where the *Sherman–Morrison–Woodbury* formula (see, e.g., [26])

$$(\mathbf{F}_i - \mathbf{K}^T\mathbf{B}^T)^{-1} = (I_{n_v} + {\mathbf{F}_i}^{-1}\mathbf{K}^T(I_{n_r} - \mathbf{B}^T{\mathbf{F}_i}^{-1}\mathbf{K}^T)^{-1}\mathbf{B}^T){\mathbf{F}_i}^{-1}$$

is used. In addition to solving with $\mathbf{F}_i$, one needs to solve with a small dense matrix $I_{n_r} - \mathbf{B}^T{\mathbf{F}_i}^{-1}\mathbf{K}^T$ of dimension $n_r \ll n_v$ to perform the solve with $\hat{\mathbf{F}}_i$. The extra solve for $\mathbf{F}_i$ with the right-hand-side $\mathbf{K}^T$ can be easily done by adding $\mathbf{K}^T$ as n_r additional columns to the matrix $\mathbf{Y}$, such that the new right-hand side becomes $\left[\begin{smallmatrix} Y & K^T \\ 0 & 0 \end{smallmatrix}\right] =: \left[\begin{smallmatrix} \tilde{Y} \\ 0 \end{smallmatrix}\right]$. In the configuration with two boundary control parameters, $n_r = 2$. The resulting saddle point system that has to be solved in every ADI step during each Newton step then is of the form

$$\begin{bmatrix} A^T + q_i M & G \\ G^T & 0 \end{bmatrix} \begin{bmatrix} \Lambda \\ * \end{bmatrix} = \begin{bmatrix} \tilde{Y} \\ 0 \end{bmatrix}, \tag{3.2}$$

where $M = M^T \succ 0$, $q_i \in \mathbb{C}^-$, and a nonsymmetric matrix A with eigenvalues in $\mathbb{C}\backslash\{0\}$. Notice that for $|q_i|$ large enough all eigenvalues of $A^T + q_i M$ are located in $\mathbb{C}^-$, which is relevant for some preconditioning techniques of the iterative solver. Details concerning these influences will be skipped in this paper and the reader is referred to [15]. Notice that the whole saddle point matrix $\mathbf{F}_i$ will usually have eigenvalues in all of $\mathbb{C}$ independent of $q_i \in \mathbb{C}^-$.

3.2. ADI shifts. The ADI shifts q_i influence the eigenvalues of the matrix $\mathbf{F}_i$ and their proper choice is crucial for the convergence of the ADI method. Among the many different ways and methods to choose q_i, we focused on the use of the heuristic *Penzl* shifts [38]. A basic ADI result tells us that (optimal) ADI shifts q_i have to be included in the convex hull of the finite spectrum of the closed-loop DAE (2.12) as described in, e.g., [46]. Using the efficient implementation to compute $q_i \in \mathbb{C}^-$ as described in [43], we run into problems, because the infinite eigenvalues of the pencil (2.12) destroy the convergence for eigenvalues with large, although finite, magnitude. This difficulty can be avoided with the matrix shifting techniques of [21] as used in subsection 2.7. Using the shifts q_i in a cyclic way provides acceptable convergence results in our cases. Further investigations to improve the convergence by an optimized selection of the ADI shifts q_i is part of our current research.

3.3. Reynolds number. The second parameter that influences the matrices and, as a result, the convergence behavior of the nested iteration is the Reynolds number. For high Reynolds numbers the system gets more convection dominated and stable finite eigenvalues are getting closer to the imaginary axis. At a certain point, the so-called critical Reynolds number, a few eigenvalues cross the imaginary axis and end up as unstable eigenvalues. In [44] it is pointed out that the critical Reynolds number changes only slightly during refinement if one starts with a sufficiently fine initial discretization.

In the numerical examples in section 4 the critical Reynolds number lies between Re = 200 and Re = 300 for all refinement levels. The determination of the exact critical Reynolds number for each refinement level is a numerically challenging task and not within the scope of this work.

As described in subsection 2.7, one needs to compute an initial feedback to ensure the convergence of the ADI algorithm, Algorithm 1, if the Reynolds number is higher than the critical Reynolds number.

The eigenvalue behavior is depicted in Figure 2 for different Reynolds numbers, where a selection of the eigenvalues close to the imaginary axis $j\mathbb{R}$ of the matrix pencil (2.5) is plotted for the numerical example of section 4 using the initial mesh of Level 1 in Table 1.

4. Numerical examples. In order to test our numerical method for boundary feedback stabilization of Navier–Stokes flows, the *von Kármán vortex street* is used as depicted in Figure 3. The flow through a channel $\Omega \subset \mathbb{R}^2$ with diameter $d_{in} = 1$ and around an obstacle of elliptic shape centered at the coordinates $(1, 0.5)$, $\frac{1}{5}d_{in}$ wide, and $\frac{1}{3}d_{in}$ high is considered. The boundary conditions of the Navier–Stokes equations (1.1) are taken as described in section 1. The parabolic inflow condition from (1.3a) is defined as

$$\vec{v}(t, \vec{x}) = \vec{g}_{in}(\vec{x}) := \begin{bmatrix} 4 \cdot (1 - x_2) \cdot x_2 \\ 0 \end{bmatrix} \quad \text{on } \Gamma_{in}$$

with a maximum of $v_{max} = 1.0$ at $\vec{x} = \begin{bmatrix} 0 & 0.5 \end{bmatrix}^T$. This condition is consistent with the *no-slip* condition (1.3a) at $(0, 0)$ and $(0, 1)$ that are parts of Γ_{wall} as well. Furthermore, we set $v_{ref} = v_{max}$ and $d_{ref} = d_{in}$ in the definition of the Reynolds number (1.2).

We use the finite element flow solver NAVIER [6] for the numerical tests to assemble the matrices. NAVIER uses a standard mixed finite element discretization of Ω (i.e., $\mathcal{P}_2$-$\mathcal{P}_1$ *Taylor–Hood* elements [32]) as shown in Figure 4. NAVIER is implemented in FORTRAN90 and the matrices are stored using the so-called matrix market format [18].

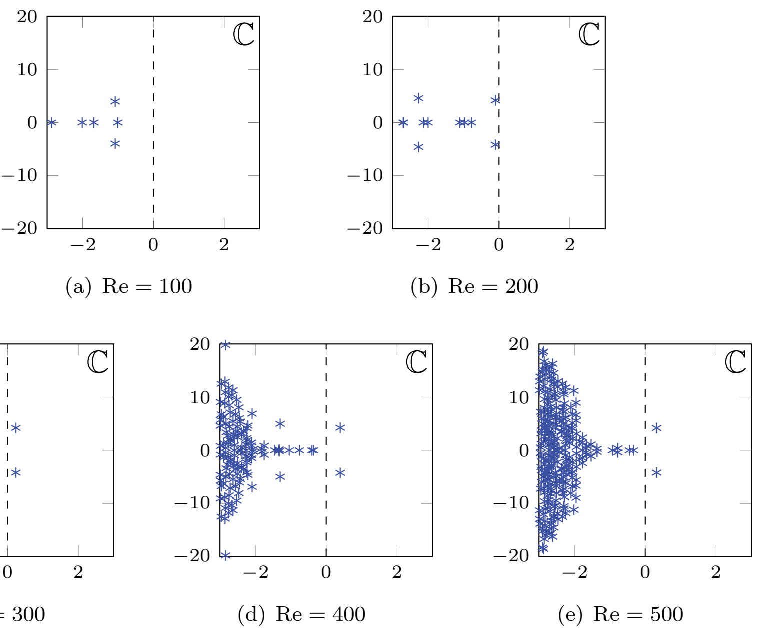


FIG. 2. *Eigenvalues of matrix pencil* (2.5) *that are close to* $j\mathbb{R}$ *for different Reynolds numbers (refinement: Level* 1*).*

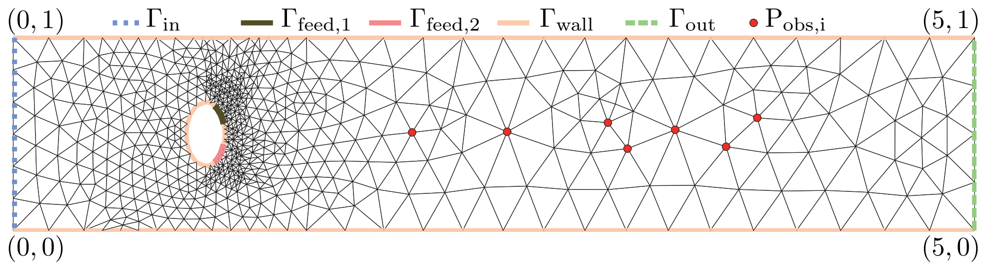


FIG. 3. *Initial triangulation of von Kármán vortex street with coordinates, boundary conditions, and observation points.*

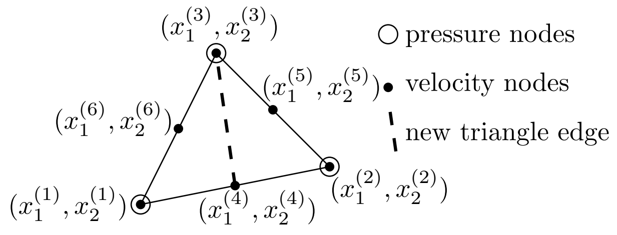
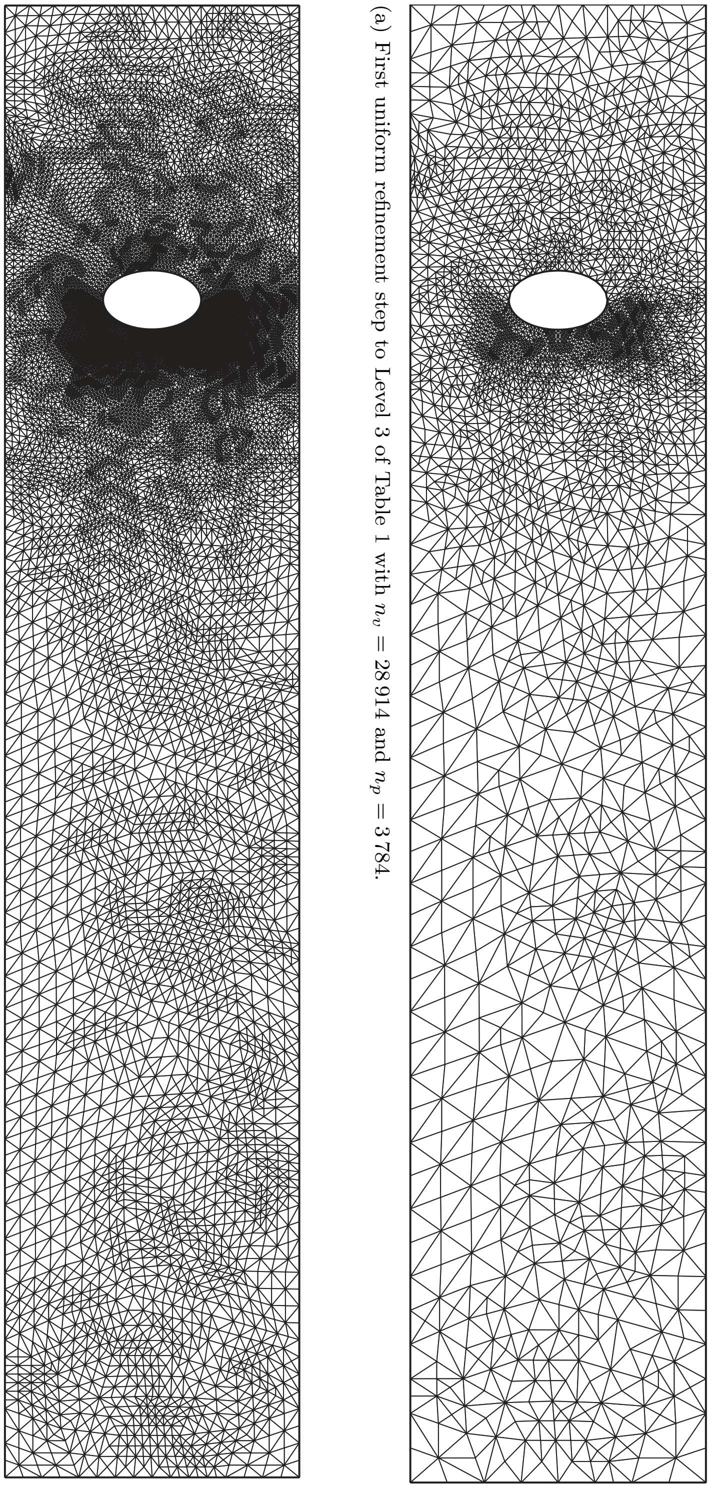


FIG. 4. $\mathcal{P}_2$-$\mathcal{P}_1$ *Taylor–Hood element.*

Starting from an initial triangulation, the mesh was refined using *bisection refinement* [5]. In the subsequent computations, six triangulations of different refinement levels were used; see Table 1 and Figure 5.

The computations for the resulting matrix equations are performed with MATLAB on a 64-bit server with Intel Xeon X5650 at 2.67 GHz, with 2 CPUs, 12 cores (6 cores per CPU), and 48 GB main memory available.



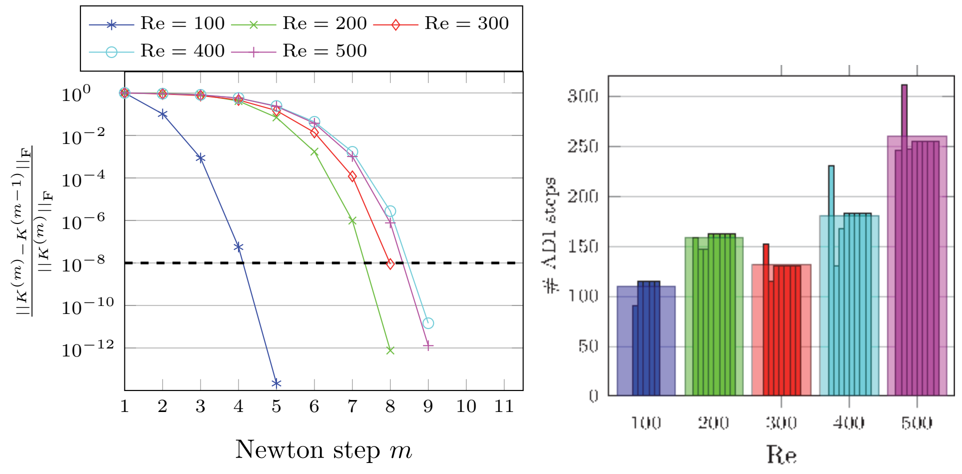
(a) First uniform refinement step to Level 3 of Table 1 with $n_v = 28\,914$ and $n_p = 3\,784$.

(b) Second uniform refinement step to Level 5 of Table 1 with $n_v = 140\,110$ and $n_p = 17\,878$.

FIG. 5. *Refined triangulations of von Kármán vortex street in Figure 3 using bisection refinement* [5].

TABLE 1
Levels of refinement.

Level	n_v	n_p
1	4796	672
2	12,292	1650
3	28,914	3784
4	64,634	8318
5	140,110	17,878
6	296,888	37,601



(a) Convergence behavior of Newton iteration for different Reynolds numbers.

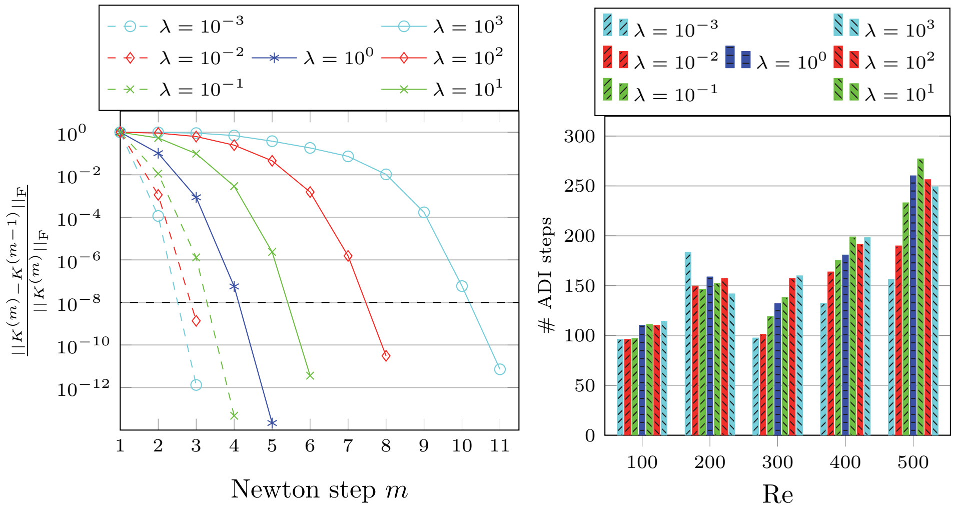
(b) Number of ADI steps per Newton step (small bars) and average over all Newton steps (wide bars).

FIG. 6. *Influence of Reynolds number for Newton and ADI convergence ($\lambda = 10^0$, $tol_{Newton} = 10^{-8}$, $tol_{ADI} = 10^{-7}$; refinement: Level 1).*

In section 3, various parameters and properties of the nested iteration are discussed. Using the above configuration, we show different results regarding these parameters and properties for the refinement levels in Table 1. At the end of this section, the realization of a closed-loop simulation within NAVIER is shown.

4.1. Parameters in the nested iteration. The influence of the different parameters is shown by comparing the convergence behavior for different problem settings. Unless otherwise stated, only one parameter is changed in each configuration. The two parameters that mostly influence the convergence of the outer Newton-ADI are the Reynolds number $\mathrm{Re} \in \mathbb{R}^+$ and the regularization parameter $\lambda \in \mathbb{R}^+$. The convergence influence of the ADI shifts is not discussed further, because the Penzl shifts are used for all configurations. However, the ADI shifts play a certain role within the iterative solution process for the saddle point systems (3.2), as shown for Stokes flows in [15] and for coupled flow problems in [16].

4.1.1. Reynolds number. Five different Reynolds numbers are used for the tests and the regularization parameter λ is fixed at $\lambda = 1$. Notice that for $\mathrm{Re} < 100$, the influence of the convection term $(\vec{v} \cdot \nabla)\vec{v}$ in (1.1a) is negligible and the flow behaves like a Stokes flow [15].



(a) Convergence behavior of Newton iteration for different λ and Re $= 100$.

(b) Average number of ADI steps per Newton step.

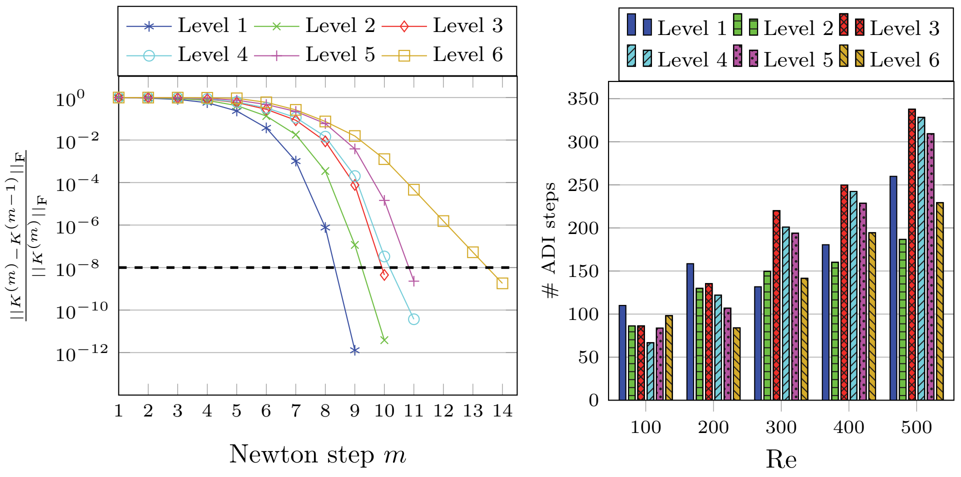
FIG. 7. *Influence of regularization parameter λ for Newton and ADI convergence ($tol_{Newton} = 10^{-8}$, $tol_{ADI} = 10^{-7}$; refinement: Level 1).*

Figure 6 shows the influence of the Reynolds number on the Newton-ADI iteration. From Figure 6(a) one can see that the number of Newton steps increases for increasing Reynolds numbers. However, there is an upper bound at $m = 9$. Although the number of Newton steps remains constant for Re ≥ 400, the number of ADI steps per Newton step still increases, as depicted in Figure 6(b), which shows details about the number of ADI steps per Newton steps depicted as small bars clustered for the different Reynolds numbers. A wider and slightly transparent block denotes the average number of ADI steps for each Reynolds number. Thus, the computing time to evaluate the Newton-ADI iteration increases for increasing Reynolds numbers. Notice that the initial feedback K_0 needs to be computed additionally for Re ≥ 300. This is represented by one additional Newton step in Figure 6(a). Hence, the system with Re $= 300$ needs fewer ADI steps on average but the Newton convergence is slower than for Re $= 200$. To summarize, the nested iteration is able to compute the feedback K for all considered Reynolds numbers with a reasonable computational effort. This behavior does not qualitatively change if one changes the regularization parameter λ. The quantitative influence of λ is shown below.

4.1.2. Regularization parameter $\boldsymbol{\lambda}$. The regularization parameter λ is introduced in the cost functional (2.9). For $\lambda > 1$ the output $\mathbf{y} = \mathcal{C}\tilde{\mathbf{z}}$ is penalized; the feedback K should force the system with more effort to the desired state. It is natural that we need higher control costs to achieve this.

In contrast, for $\lambda < 1$ the influence of the output is reduced compared to the control cost. The control cost $\mathbf{u}^T\mathbf{u}$ is penalized in an indirect way in that the controller K tries to force the system to the desired state, but with a reduced amount of control effort.

The convergence behavior of the Newton-ADI is illustrated for $10^{-3} \leq \lambda \leq 10^{3}$ in Figure 7. Figure 7(a) shows the convergence of the Newton iteration for Re $= 100$. The number of Newton steps increases monotonically with respect to λ. This seems



(a) Convergence behavior of Newton iteration for different refinement levels (Re = 500).

(b) Average number of ADI steps per Newton step during Newton-ADI iteration.

FIG. 8. *Influence of refinement level for Newton and ADI convergence ($\lambda = 10^0$, $tol_{Newton} = 10^{-8}$, $tol_{ADI} = 10^{-7}$).*

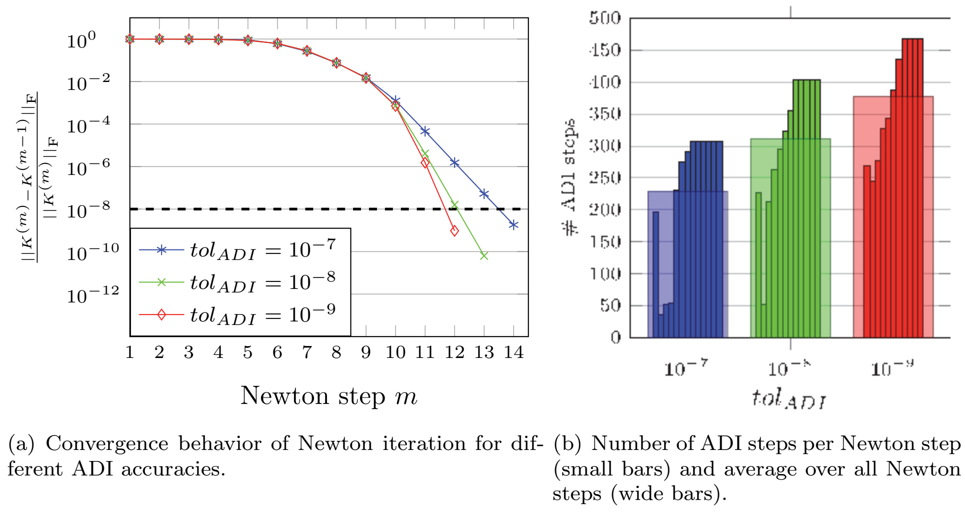
to be natural, because the feedback K has to work more effectively if the output is more penalized; the feedback K has to provide more "information" to achieve its aim in a shorter amount of time. For small λ there exists a lower bound at $m = 3$. The Newton convergence for different λ stays qualitatively the same for different Reynolds numbers. The number of steps varies quantitatively, as shown in Figure 6(a). Figure 7(b) illustrates the average number of ADI steps within the Newton iteration for a varying Reynolds number and regularization parameter λ. The number of ADI steps is nearly constant for Re = 100 and Re = 200, except for the configuration Re = 200, $\lambda = 10^{-3}$.

For Re $\geq$ 300 the number of steps increases as λ gets larger. For Re = 500 there are two configurations that do not perfectly fit into this scheme. The influence of the initial feedback K_0 for Re > 200 influences the number of steps as well, such that some configurations with supposedly higher complexity need fewer ADI steps, because one starts with a quite good initial guess.

The best choice of λ in the closed-loop simulation is not obvious. Moreover, the underlying finite element discretization gives us natural bounds for the input $\mathbf{u}$ and, hence, for λ as well. Furthermore, the best choice of λ is a separate optimization task and depends on the design of the test configuration, but it will not be discussed further in this paper.

4.2. Refinement levels. The accuracy of the finite element approximation of the equations in (2.2) increases if one refines the mesh. As described above, we start with the coarse grid triangulation in Figure 3 and use a *bisection refinement* [5] to refine the triangulation five times. Each time, the triangles are split into two subtriangles by inserting a new edge, as depicted in Figure 4, that bisects the *refinement edge* in the existing triangle. Notice that every second refinement level corresponds to one uniform refinement [5] in the whole domain. Refinement levels 3 and 5 are illustrated in Figure 5.

In order to resolve the feedback influence as described in subsection 4.5 already on the coarsest mesh, the initial triangulation is adaptively refined; see Figure 3. The



(a) Convergence behavior of Newton iteration for different ADI accuracies.

(b) Number of ADI steps per Newton step (small bars) and average over all Newton steps (wide bars).

FIG. 9. *Influence of ADI accuracy to Newton convergence (Re = 500; refinement: Level 6; $tol_{Newton} = 10^{-8}$).*

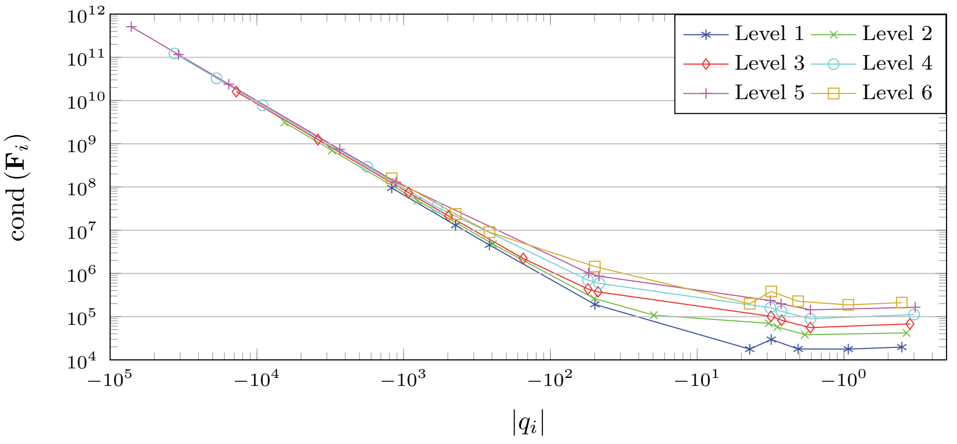


FIG. 10. *Condition number of matrix $\mathbf{F}_i$ for different ADI shifts q_i during the first Newton step for different refinement levels (Re = 500).*

mesh size is smaller close to the control boundaries $\Gamma_{\text{feed},1}, \Gamma_{\text{feed},2}$. Finer meshes yield more detailed information about the physical behavior of the flow and can be used to compute the optimal control $\mathbf{u}(t)$. Similar to the influence of higher regularization parameters in subsection 4.1.2, the number of Newton steps increases to process the higher amount of information.

The Newton-ADI algorithm converges quadratically for all refinement levels up to Level 5 within 9 to 11 steps, as illustrated in Figure 8(a) for Re = 500, $\lambda = 10^0$, and an ADI tolerance $tol_{\text{ADI}} = 10^{-7}$. Only Level 6 does not show quadratic convergence anymore such that significantly more Newton steps are needed. This problem can be observed for all configurations with Re $\geq$ 200.

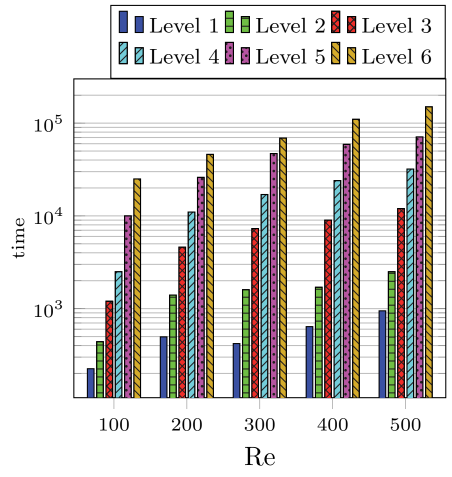


FIG. 11. *Total time in seconds for whole Newton-ADI iteration ($\lambda = 10^0$, $tol_{Newton} = 10^{-8}$, $tol_{ADI} = 10^{-7}$).*

The comparison of the average number of ADI steps per Newton step is depicted in Figure 8(b). For Re = 100 the number of steps varies slightly. For all other Reynolds numbers the number of ADI steps decreases if the refinement level increases. For Re > 200 the first two refinement levels do not share this behavior. We believe that this unstable behavior for flows with Re > 200 is due to insufficient mesh resolution for these refinement levels. The reduction of ADI steps in Level 6 is drastic and leads to worse convergence of the Newton iteration. This means the ADI tolerance is not sufficient to ensure quadratic convergence of the Newton iteration. Figure 9(a) shows the convergence of the Newton-ADI iteration for different ADI tolerances. Using a higher ADI accuracy ensures the quadratic Newton convergence. However, the amount of ADI steps increases drastically as well. This example shows the necessity of an automatic tolerance adjustment as in the inexact Newton iteration (see, e.g., [24]) that is part of recent research.

A variation of Reynolds number or regularization parameter λ only changes the quantitative results regarding the explanations in subsection 4.1 and is not further illustrated. Another aspect that leads to increased numbers of Newton steps is the increasing condition number of the shifted saddle point system $\mathbf{F}_i$, as plotted in Figure 10 for all ADI shifts during the first Newton step. Thereby, the horizontal axis depicts the magnitude of all ADI shifts for all six refinement levels for Re = 500.

4.3. Timings for Newton-ADI iteration. Various parameters influence the number of Newton and ADI iteration steps. The most time-consuming operation within the nested iteration is to solve the arising saddle point system (3.2) in each ADI step. The ADI shift and the Reynolds number influence the condition number and, as a result, the time to solve these systems. Nevertheless, the average time to solve one single saddle point system is nearly constant for a fixed refinement level and is not illustrated in more detail. The most important parameter for the computation time is the refinement level. The direct solver needs more time and memory due to increasing system dimensions. The total times for the whole Newton-ADI iteration are depicted in Figure 11 and are given in detail in Table 2. The influence of the Reynolds number that is described in subsection 4.1.1 is reflected in the timings as expected.

TABLE 2
Total time in seconds for whole Newton-ADI iteration ($\lambda = 10^0$, $tol_{Newton} = 10^{-8}$, $tol_{ADI} = 10^{-7}$).

		Reynolds number				
		100	200	300	400	500
Level	1	$2.2 \cdot 10^2$	$5.0 \cdot 10^2$	$4.2 \cdot 10^2$	$6.4 \cdot 10^2$	$9.5 \cdot 10^2$
	2	$4.4 \cdot 10^2$	$1.5 \cdot 10^3$	$1.6 \cdot 10^3$	$1.7 \cdot 10^3$	$2.5 \cdot 10^3$
	3	$1.3 \cdot 10^3$	$4.6 \cdot 10^3$	$7.3 \cdot 10^3$	$9.1 \cdot 10^3$	$1.2 \cdot 10^4$
	4	$2.6 \cdot 10^3$	$1.1 \cdot 10^4$	$1.7 \cdot 10^4$	$2.4 \cdot 10^4$	$3.2 \cdot 10^4$
	5	$1.0 \cdot 10^4$	$2.6 \cdot 10^4$	$4.7 \cdot 10^4$	$5.9 \cdot 10^4$	$7.1 \cdot 10^4$
	6	$2.5 \cdot 10^4$	$4.6 \cdot 10^4$	$7.0 \cdot 10^4$	$1.1 \cdot 10^5$	$1.5 \cdot 10^5$

Furthermore, the influence of the mesh refinement can be seen easily. For comparison we want to mention that the matrix assembling times of NAVIER and the times to compute the initial feedback are not part of these times, because both processes can be done beforehand independently of all parameters that are used later on.

4.4. Control cost. The number of Newton steps associates in a natural way to the difficulty of the optimization problem. The feedback matrices computed by using more Newton steps force the system more strictly to the desired state. Such a strict forcing requires larger control cost, which is implicitly shown in Figure 7. To show this in a more detailed way we compute the stationary optimal control $\mathbf{u_w}$ with respect to the linearization point $\vec{w}$ of system (2.1) and its corresponding discretized velocity field $\mathbf{w}$. Thereby, $\mathbf{u_w}$ is measured as the square of the Euclidean norm:

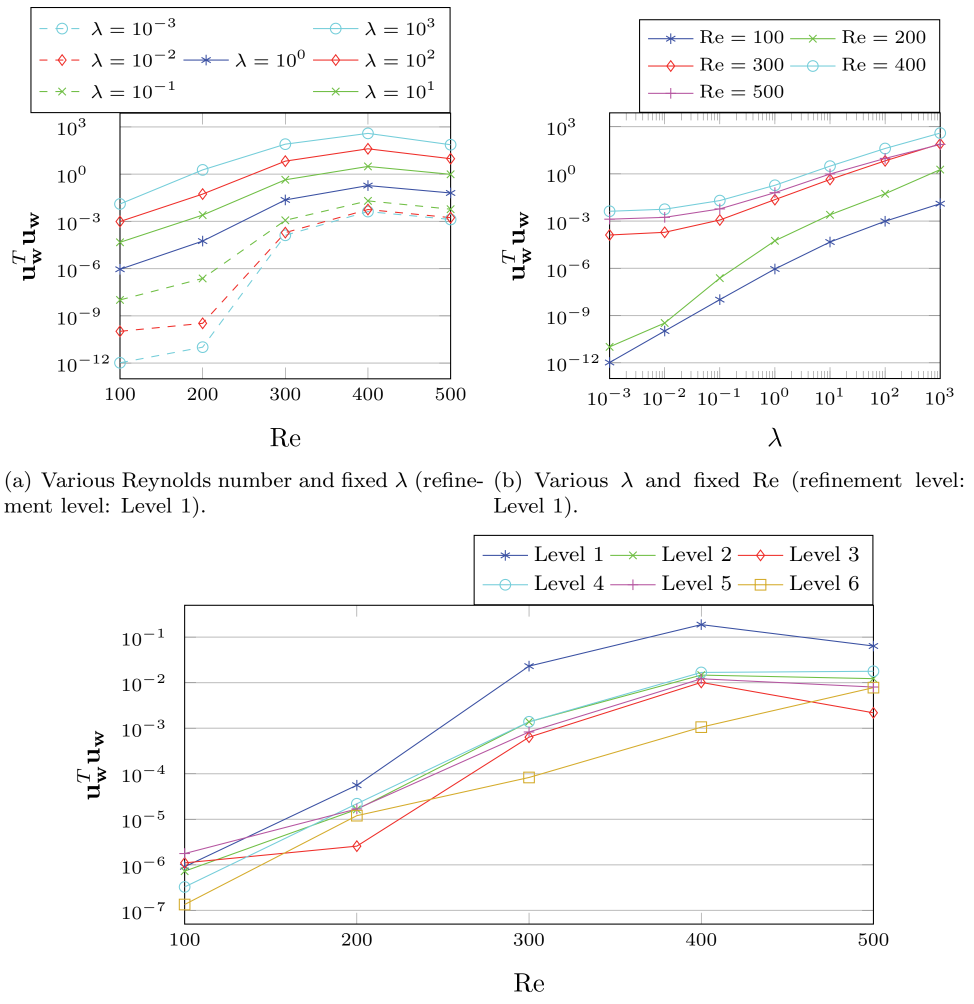
$$||\mathbf{u_w}||_2^2 = \mathbf{u_w}^T\mathbf{u_w} = \mathbf{w}^T K^T K\mathbf{w}.$$

Using the same configurations as above, Figure 12(a) shows that the control cost increases for higher Reynolds numbers up to an upper bound; this is the same behavior as can be seen for the number of Newton steps in Figure 6(a). It is natural that the control cost is smaller for lower λ, because the control cost is implicitly penalized for $\lambda < 1$. To illustrate the monotone increase for increasing λ, the Reynolds number is fixed and the influence of the varying λ is shown in Figure 12(b). There is a noticeable gap between $\mathrm{Re} \leq 200$ and $\mathrm{Re} \geq 300$ that is related to the instability of systems with higher Reynolds numbers, as explained in section 3.3.

In Figure 12(c), the initial control costs are compared for different Reynolds numbers regarding the refinement levels. Levels 1 to 5 show a convergent behavior. This does not hold for Level 6 in the case of $\mathrm{Re} \in \{100, 300, 400\}$ anymore. We think a better understanding of stopping criteria and their connections to ensure the convergence within the Newton-ADI iteration would solve this problem.

4.5. Closed-loop simulation. In this subsection the usability of the feedback matrix K computed via Algorithm 2 to stabilize the flow problem (1.1) during a forward simulation is verified. Here, by stabilization we mean smoothing vortexes behind the obstacle that occur for Reynolds numbers $\mathrm{Re} > 200$.

A laminar stationary flow field $\vec{w}$ is considered as linearization point. The matrix K, computed in MATLAB®, is imported into the finite element flow solver NAVIER using the matrix market format [18].



(a) Various Reynolds number and fixed λ (refinement level: Level 1).

(b) Various λ and fixed Re (refinement level: Level 1).

(c) Various Reynolds number and fixed refinement level ($\lambda = 10^0$).

FIG. 12. *Initial control cost with* $\mathbf{u}_\mathbf{w} = -K\mathbf{w}$*, where* $\mathbf{w}$ *is the discretized version of the stationary velocity field* $\vec{w}$ *in* (2.1).

In [40], Raymond shows that the optimal control $\mathbf{u}(t)$ computed for the linearized system (2.2) also stabilizes the nonlinear system (1.1) if one assumes $\vec{v} \approx \vec{w}$; this means the deviations of the velocity field $\vec{v}$ from the stationary velocity field $\vec{w}$ are small enough [7].

For the controlled forward simulation we proceed as follows. The same finite element discretization as above is used, such that the spatially discretized nonlinear system can be written as

$$M\frac{d}{dt}\mathbf{v}(t) = A(\mathbf{v}(t))\mathbf{v}(t) + G\tilde{\mathbf{p}}(t) + \tilde{B}\mathbf{u}(t), \tag{4.1a}$$

$$0 = G^T\mathbf{v}(t), \tag{4.1b}$$

$$\mathbf{y}(t) = C\mathbf{v}(t) \tag{4.1c}$$

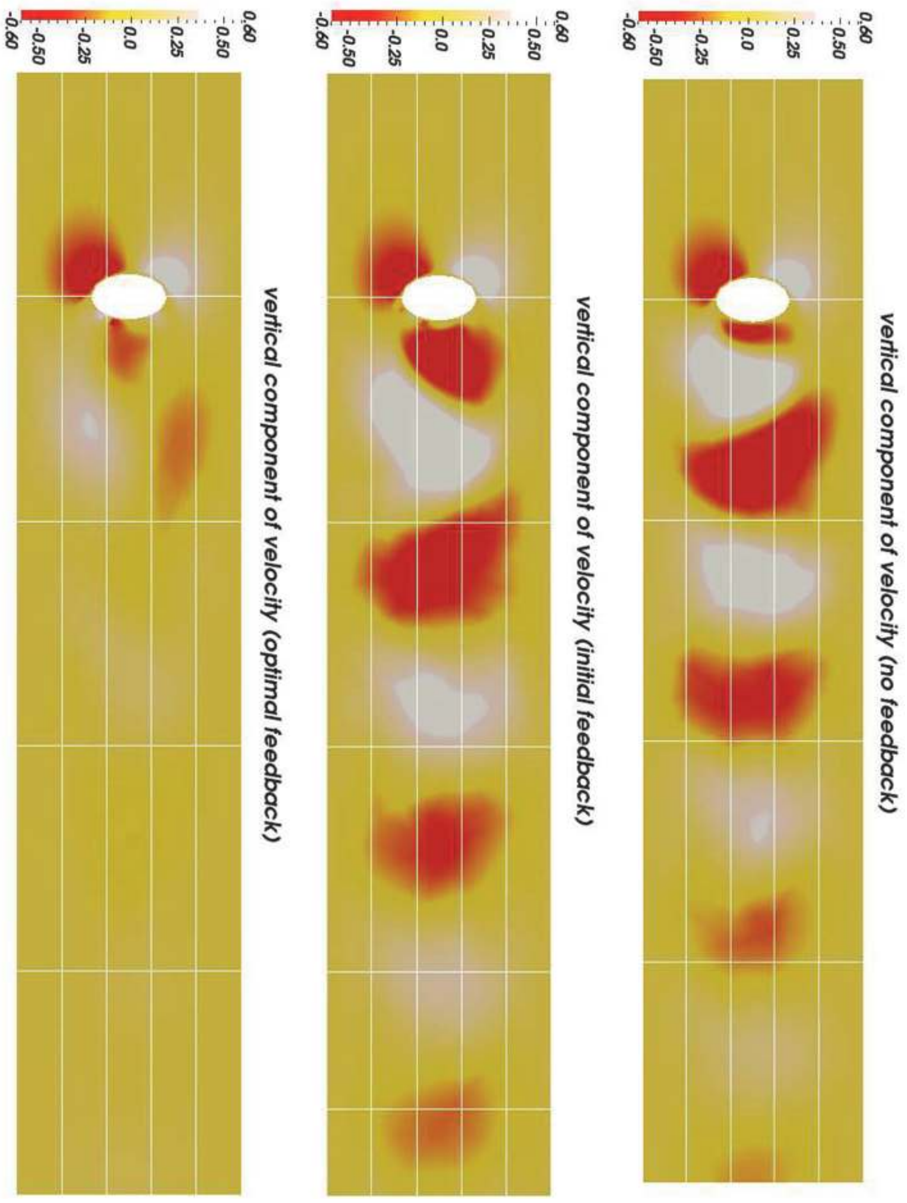


FIG. 13. *Screenshots of closed-loop simulation for* $Re = 300, \lambda = 2$ *at* $t = 25$.

with the discrete velocity $\mathbf{v}(t) \in \mathbb{R}^{n_v}$, the discrete pressure $\tilde{\mathbf{p}}(t) \in \mathbb{R}^{n_\mathbf{p}}$, and the nonlinear Navier–Stokes operator $A(\mathbf{v}(t))$ that depends on the current velocity field $\mathbf{v}(t)$. Additionally, we have the discrete versions of the boundary conditions (1.3a)–(1.3b). We start with an initial condition

$$\mathbf{v}(0) = \mathbf{w},$$

such that the perturbations from the stationary trajectory are zero at the beginning. See [6] for more details on solving the nonlinear system (4.1).

The actual $\mathbf{u}(t)$ is computed on the discrete level via

$$\mathbf{u}(t) = -K\mathbf{z}(t) = -(K\mathbf{v}(t) - K\mathbf{w}),$$

which means the feedback matrix K is applied to the stationary linearization point $\mathbf{w}$ and the recent velocity field $\mathbf{v}(t)$. The values of $\mathbf{u}(t)$ describe the magnitude of the parabolic inflow or outflow conditions via the control boundaries $\Gamma_{feed,i}\ \forall i = 1, 2$; compare Figures 1 and 3. In this setting the input operator $\tilde{B}$ maps the parabolic control inflow at the feedback boundaries $\Gamma_{feed,i}\ \forall i = 1, 2$ with the magnitudes $u_i(t)$ to the control Dirichlet conditions

$$\mathbf{v}(t) = \mathbf{g}_{feed}(t) := \tilde{B}u_i(t) \text{ on } \Gamma_{feed,i}\ \forall i = 1, 2.$$

Using these boundary conditions, the closed-loop forward simulation can be solved.

The output operator C measures the vertical velocity component v_{x_2} at seven different nodes $\mathbf{P}_{\mathrm{obs,i}}$ of the FE grid behind the obstacle as depicted in Figure 3. Getting these components as small as possible is a way to measure how laminar the flow field is.

The choice of λ influences the quality of the results. To find the optimal value λ one would need to solve an optimization process; we do not show details regarding this problem in this paper. Another difficulty that occurs is that the computed control $\mathbf{u}(t)$ is not directly constrained. Thus, the required inflow via the control boundaries $\Gamma_{feed,i}$ can exceed the resolution capabilities of the underlying finite element grid.

Figure 13 shows a flow field for Re = 300. The vertical component of the velocity is depicted in red, as maximal value downward, and white, as maximal value upward. As explained above, the observation matrix C measures the vertical velocities only in a few nodes behind the obstacle. We divided the domain into equally sized rectangles of five rows and five columns for better illustration. The measurement nodes are located in the vertical center row in the third and fourth horizontal sections from the left. To achieve the goal of smoothing all vortexes, the flow field should not consist of vertical velocity components; a zero vertical velocity corresponds to yellow in the applied color scale. Due to the nature of our cost functional, we focus especially on the above-mentioned measuring rectangles on which deviations from zero are penalized.

The top picture shows the velocity field at $t = 25$ without any feedback influence. The occurring vortexes are shown by the red and white areas that move away from the obstacle in an alternating order.

The middle picture shows the influence of the initial feedback K_0. Although K_0 numerically stabilizes the matrix pencil, the forward simulation is not stabilized with respect to occurring vortexes.

Finally, the bottom picture represents the closed-loop simulation for the feedback matrix K computed with Algorithm 2 using the regularization parameter $\lambda = 2$. Nearly all vortexes are smoothed and the flow field is laminar, especially in the rectangles in our focus. Here, the value $\lambda = 2$ is an experimentally determined value.

For $\lambda < 2$ the flow field still consists of vortexes. For $\lambda > 2$ the finite element grid cannot handle the in-/outflow conditions required for the computed feedback. More drastic feedback influence would require a local mesh refinement. Unfortunately, this would require us to compute a new feedback matrix. Recently, we investigated the connection between coarse grid computed feedback influences applied to fine grid forward simulations. A video of the closed-loop forward simulation is available online [47].

5. Conclusions and outlook.

In this paper we have investigated the numerical realization of a boundary feedback stabilization for instationary Navier–Stokes flows. Based on an approach by Raymond [39, 40], the linearization around a given stationary trajectory is the starting point to apply an LQR approach. Raymond uses the Leray projector to define the evolution equation for the regulator approach on the divergence-free vector field. The stabilizing feedback operator is based on the solution of an operator Riccati equation. Using a finite element discretization in space yields an algebraic Riccati equation that has to be solved numerically. These large-scale and nonlinear equations can be solved with a Newton type method. Starting with a suitable initial guess, the Newton iteration converges superlinearly to the unique root that defines the stabilizing feedback. Every Newton step consists of solving a Lyapunov equation, which is solved by applying a low-rank factor ADI method. To this end, a linear solve has to be done in each ADI step.

We have used a Taylor–Hood [32] mixed spatial discretization as a stable conforming finite element method to semidiscretize the evolution equation. Thus, the incompressibility condition is not automatically built into the trial space and requires additional effort. In [29], Heinkenschloss and colleagues proposed a trick to overcome this difficulty. We have extended the ideas in [15] that demonstrate the equivalence of this approach with the use of a discrete Leray projection for discretized flow fields. We have shown that the Newton-ADI method to solve an algebraic Riccati equation can be reformulated for such a DAE setting. The crucial step to using the Newton-ADI method without the explicit projection is the linear solve of large-scale nonsymmetric saddle point systems. The occurring structure is common for Navier–Stokes flow related problems.

Compared to the Stokes case [15], the Navier–Stokes case has the need of an initial feedback to guarantee convergence of the Newton-ADI method. In this paper we have shown a way to compute such an initial feedback based on the solution of a Bernoulli equation. Using this initial feedback, the threefold nested iteration determines a stabilizing feedback matrix.

We have illustrated the influence of the occurring parameters with some numerical examples based on the forward simulation of the von Kármán vortex street. A closed-loop simulation for this example has verified the stabilizing property of the feedback matrix.

In the future we are going to investigate more details about coupled multi-field flow problems as in [8]. Furthermore, the parameter dependence within the nested iteration, as well as the handling of complex ADI shifts, will be a challenging task. We hope to benefit from some recent ADI improvements such as avoiding complex arithmetic [11] or efficient low-rank residual computations [12].

Moreover, we plan to expand the iterative solution strategies in [15] to the Navier–Stokes and coupled multi-field flow problems on finer meshes. To this end, we will investigate efficient block preconditioning methods. To deal with the special structure of the saddle point systems arising in the Newton-ADI algorithm we will investigate recycling and block Krylov methods to accelerate the time-consuming handling of multiple right-hand sides for iterative linear solvers.

Acknowledgments. The work reported in this paper is part of the project Optimal Control-Based Feedback Stabilization of Multi-Field Flow Problems. This project is embedded in DFG priority program 1253, Optimization with Partial Differential Equations. We thank Stephan Weller for help with implementation issues in NAVIER.

REFERENCES

[1] L. Amodei and J. M. Buchot, *A stabilization algorithm of the Navier-Stokes equations based on algebraic Bernoulli equation*, Numer. Linear Algebra Appl., 19 (2012), pp. 700–727.

[2] W. F. Arnold and A. J. Laub, *Generalized eigenproblem algorithms and software for algebraic Riccati equations*, Proc. IEEE, 72 (1984), pp. 1746–1754.

[3] H. T. Banks and K. Ito, *A numerical algorithm for optimal feedback gains in high dimensional linear quadratic regulator problems*, SIAM J. Control Optim., 29 (1991), pp. 499–515.

[4] H. T. Banks and K. Kunisch, *The linear regulator problem for parabolic systems*, SIAM J. Control Optim., 22 (1984), pp. 684–698.

[5] E. Bänsch, *Local mesh refinement in* 2 *and* 3 *dimensions*, IMPACT Comput. Sci. Eng., 3 (1991), pp. 181–191.

[6] E. Bänsch, *Simulation of instationary, incompressible flows*, Acta Math. Univ. Comenian., 67 (1998), pp. 101–114.

[7] E. Bänsch and P. Benner, *Stabilization of incompressible flow problems by Riccati-based feedback*, in Constrained Optimization and Optimal Control for Partial Differential Equations, G. Leugering, S. Engell, A. Griewank, M. Hinze, R. Rannacher, V. Schulz, M. Ulbrich, and S. Ulbrich, eds., Internat. Ser. Numer. Math. 160, Birkhäuser, Basel, 2012, pp. 5–20.

[8] E. Bänsch, P. Benner, J. Saak, and H. K. Weichelt, *Optimal control-based feedback stabilization of multi-field flow problems*, in Trends in PDE Constrained Optimization, G. Leugering, P. Benner, S. Engell, A. Griewank, H. Harbrecht, M. Hinze, R. Rannacher, and S. Ulbrich, eds., Internat. Ser. Numer. Math. 165, Birkhäuser, Basel, 2014, pp. 173–188.

[9] S. Barrachina, P. Benner, and E. S. Quintana-Ortí, *Efficient algorithms for generalized algebraic Bernoulli equations based on the matrix sign function*, Numer. Algorithms, 46 (2007), pp. 351–368.

[10] P. Benner, *Solving large-scale control problems*, IEEE Control Syst. Mag., 14 (2004), pp. 44–59.

[11] P. Benner, P. Kürschner, and J. Saak, *Efficient handling of complex shift parameters in the low-rank Cholesky factor ADI method*, Numer. Algorithms, 62 (2013), pp. 225–251.

[12] P. Benner, P. Kürschner, and J. Saak, *An improved numerical method for balanced truncation for symmetric second order systems*, Math. Comput. Model. Dyn. Syst., 19 (2013), pp. 593–615.

[13] P. Benner, J.-R. Li, and T. Penzl, *Numerical solution of large Lyapunov equations, Riccati equations, and linear-quadratic control problems*, Numer. Linear Algebra Appl., 15 (2008), pp. 755–777.

[14] P. Benner and J. Saak, *A Galerkin-Newton-ADI Method for Solving Large-Scale Algebraic Riccati Equations*, Preprint SPP1253-090, DFG-SPP1253, 2010; available online from http://www.am.uni-erlangen.de/home/spp1253/wiki/images/2/28/Preprint-SPP1253-090.pdf.

[15] P. Benner, J. Saak, M. Stoll, and H. K. Weichelt, *Efficient solution of large-scale saddle point systems arising in Riccati-based boundary feedback stabilization of incompressible Stokes flow*, SIAM J. Sci. Comput., 35 (2013), pp. S150–S170.

[16] P. Benner, J. Saak, M. Stoll, and H. K. Weichelt, *Efficient solvers for large-scale saddle point systems arising in feedback stabilization of multi-field flow problems*, in System Modeling and Optimization, C. Pötzsche, C. Heuberger, B. Kaltenbacher, and F. Rendl, eds., IFIP Adv. Inf. Commun. Technol. 443, Springer, New York, 2014, pp. 11–20.

[17] A. Bensoussan, G. Da Prato, M. C. Delfour, and S. K. Mitter, *Representation and Control of Infinite Dimensional Systems, Vol.* I, Systems Control Found. Appl., Birknäuser, Boston, 1992.

[18] R. F. Boisvert, R. Pozo, and K. A. Remington, *The Matrix Market Exchange Formats: Initial Design*, NIST Interim Report 5935, National Institutes of Standards and Technology, 1996.

[19] M. O. Bristeau, R. Glowinski, and J. Périaux, *Numerical methods for the Navier-Stokes equations. Applications to the simulation of compressible and incompressible viscous flows*, in Finite Elements in Physics (Lausanne, 1986), North-Holland, Amsterdam, 1987, pp. 73–187.

[20] J. A. Burns, E. W. Sachs, and L. Zietsman, *Mesh independence of Kleinman–Newton iterations for Riccati equations in Hilbert space*, SIAM J. Control Optim., 47 (2008), pp. 2663–2692.

[21] K. A. Cliffe, T. J. Garratt, and A. Spence, *Eigenvalues of block matrices arising from problems in fluid mechanics*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 1310–1318.

[22] E. Deriaz and V. Perrier, *Orthogonal Helmholtz decomposition in arbitrary dimension using divergence-free and curl-free wavelets*, Appl. Comput. Harmon. Anal., 26 (2009), pp. 249–269.

[23] 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, 2005.

[24] F. Feitzinger, T. Hylla, and E. W. Sachs, *Inexact Kleinman-Newton method for Riccati equations*, SIAM J. Matrix Anal. Appl., 31 (2009), pp. 272–288.

[25] V. Girault and P. A. Raviart, *Finite Element Methods for Navier-Stokes Equations*, Springer-Verlag, Berlin, 1986.

[26] G. H. Golub and C. F. van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins University Press, Baltimore, 1996.

[27] S. Gugercin, T. Stykel, and S. Wyatt, *Model reduction of descriptor systems by interpolatory projection methods*, SIAM J. Sci. Comput., 35 (2013), pp. B1010–B1033.

[28] S. Hein, *MPC-LQG-Based Optimal Control of Parabolic PDEs*, Dissertation, TU Chemnitz, 2009; available from http://archiv.tu-chemnitz.de/pub/2010/0013.

[29] M. Heinkenschloss, D. C. Sorensen, and K. Sun, *Balanced truncation model reduction for a class of descriptor systems with application to the Oseen equations*, SIAM J. Sci. Comput., 30 (2008), pp. 1038–1063.

[30] J. G. Heywood, R. Rannacher, and S. Turek, *Artificial boundaries and flux and pressure conditions for the incompressible Navier-Stokes equations*, Internat. J. Numer. Methods Fluids, 22 (1996), pp. 325–352.

[31] M. Hinze and K. Kunisch, *Second order methods for boundary control of the instationary Navier-Stokes system*, Z. Angew. Math. Mech., 84 (2004), pp. 171–187.

[32] P. Hood and C. Taylor, *Navier-Stokes equations using mixed interpolation*, in Finite Element Methods in Flow Problems, J. T. Oden, R. H. Gallagher, C. Taylor, and O. C. Zienkiewicz, eds., University of Alabama in Huntsville Press, 1974, pp. 121–132.

[33] D. L. Kleinman, *On an iterative technique for Riccati equation computations*, IEEE Trans. Automat. Control, 13 (1968), pp. 114–115.

[34] P. Lancaster and L. Rodman, *The Algebraic Riccati Equation*, Oxford University Press, Oxford, UK, 1995.

[35] I. Lasiecka and R. Triggiani, *Control Theory for Partial Differential Equations: Continuous and Approximation Theories* I. *Abstract Parabolic Systems*, Cambridge University Press, Cambridge, UK, 2000.

[36] J.-R. Li and J. White, *Low rank solution of Lyapunov equations*, SIAM J. Matrix Anal. Appl., 24 (2002), pp. 260–280.

[37] A. Locatelli, *Optimal Control: An Introduction*, Birkhäuser, Basel, 2001.

[38] T. Penzl, Lyapack *Users Guide*, Preprint SFB393/00-33, TU Chemnitz, Germany, 2000; available from http://www.tu-chemnitz.de/sfb393/sfb00pr.html.

[39] J.-P. Raymond, *Local boundary feedback stabilization of the Navier-Stokes equations*, in Proceedings of Control Systems: Theory, Numerics and Applications, Rome, 2005; available from http://pos.sissa.it.

[40] J.-P. Raymond, *Feedback boundary stabilization of the two-dimensional Navier–Stokes equations*, SIAM J. Control Optim., 45 (2006), pp. 790–828.

[41] J.-P. Raymond, *Feedback boundary stabilization of the three-dimensional incombressible Navier–Stokes equations*, J. Math. Pures Appl. (9), 87 (2007), pp. 627–669.

[42] J.-P. Raymond, *Stokes and Navier–Stokes equations with nonhomogeneous boundary conditions*, Ann. Inst. H. Poincaré Anal. Non Linéare, 24 (2007), pp. 921–951.

[43] J. Saak, *Efficient Numerical Solution of Large Scale Algebraic Matrix Equations in PDE Control and Model Order Reduction*, Dissertation, TU Chemnitz, 2009; available from http://nbn-resolving.de/urn:nbn:de:bsz:ch1-200901642.

[44] M. Schäfer and S. Turek, *Benchmark Computations of Laminar Flow Around a Cylinder*, Preprint 96–03, Universität Heidelberg, 1996.

[45] W. E. Schiesser, *Numerical Methods of Lines*, Academic Press, New York, 1991.

[46] E. L. Wachspress, *The ADI Model Problem*, Springer, New York, 2013.

[47] H. K. Weichelt, *Riccati-Based Feedback Stabilized Navier-Stokes Flow*, https://zenodo.org/record/7110#.VLzNwV3KzEU (2013).

[48] J. Weickert, *Navier-Stokes Equations as a Differential-Algebraic System*, Preprint SFB393/96-08, Department of Mathematics, Chemnitz University of Technology, 1996.