SIAM J. SCI. COMPUT.
Vol. 35, No. 5, pp. S150–S170

EFFICIENT SOLUTION OF LARGE-SCALE SADDLE POINT SYSTEMS ARISING IN RICCATI-BASED BOUNDARY FEEDBACK STABILIZATION OF INCOMPRESSIBLE STOKES FLOW*

PETER BENNER†, JENS SAAK†, MARTIN STOLL‡, AND HEIKO K. WEICHELT§

Abstract. We investigate numerical methods for solving large-scale saddle point systems which arise during the feedback control of flow problems. We focus on the instationary Stokes equations that describe instationary, incompressible flows for moderate viscosities. After a mixed finite element discretization we get a differential-algebraic system of differential index two [J. Weickert, *Navier-Stokes Equations as a Differential-Algebraic System*, Preprint SFB393/96-08, Department of Mathematics, Chemnitz University of Technology, Chemnitz, Germany, 1996]. To reduce this index, we follow the analytic ideas of [J.-P. Raymond, *SIAM J. Control Optim.*, 45 (2006), pp. 790–828] coupled with the projection idea of [M. Heinkenschloss, D. C. Sorensen, and K. Sun, *SIAM J. Sci. Comput.*, 30 (2008), pp. 1038–1063]. Avoiding this explicit projection leads to solving a series of large-scale saddle point systems. In this paper we construct iterative methods to solve such saddle point systems by deriving efficient preconditioners based on the approaches of Wathen and colleagues, e.g., [M. Stoll and A. Wathen, *J. Comput. Phys.*, 232 (2013), pp. 498–515]. In addition, the main results can be extended to the nonsymmetric case of linearized Navier–Stokes equations. We conclude with numerical examples showcasing the performance of our preconditioned iterative saddle point solver.

Key words. flow control, Stokes equations, Riccati-based feedback, saddle point systems, Schur complement approximation

AMS subject classifications. 65F08, 65F10, 93D15, 49M15, 76D55

DOI. 10.1137/120881312

1. Introduction.

Stabilization of flow problems is crucial for many areas of engineering, for example, the automotive and aerospace industries, as well as nanotechnology. In the latter case of microfluidic structures, we often encounter flow problems at moderate viscosities or Reynolds numbers that do not require turbulence modeling [27]. In this paper, we are concerned with such a setting; therefore, we follow the Riccati-based feedback approach for stabilization of incompressible flow problems [6]. In contrast to the common idea of distributed control, we consider boundary control, which is more natural in a technical implementation. The analytic approach for feedback boundary stabilization of Stokes and linearized Navier–Stokes equations was given by Raymond in [39, 40, 41]. Raymond used the *Leray* projector to project the velocity functions onto the space of divergence-free vector functions [19] in order to deal with the algebraic constraints imposed by the incompressibility

*Received by the editors June 15, 2012; accepted for publication (in revised form) March 19, 2013; published electronically October 28, 2013. This work was supported by the project *Optimal Control-Based Feedback Stabilization of Multi-Field Flow Problems*. The project is a joint work with Eberhard Bänsch and is part of the DFG priority program 1253: *Optimization with Partial Differential Equations*.

http://www.siam.org/journals/sisc/35-5/88131.html

†Research Group Mathematics in Industry and Technology (MiIT), Chemnitz University of Technology, 09107 Chemnitz, Germany, and Research Group Computational Methods in Systems and Control Theory (CSC), Max Planck Institute for Dynamics of Complex Technical Systems Magdeburg, 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, 39106 Magdeburg, Germany (stollm@mpi-magdeburg.mpg.de).

§Research Group Mathematics in Industry and Technology (MiIT), Chemnitz University of Technology, 09107 Chemnitz, Germany (heiko.weichelt@mathematik.tu-chemnitz.de).

condition. Following Raymond's analytic approach, Bänsch and Benner investigated several ideas for the numerical treatment of the Leray projection in [6]. One of these ideas was to use the projection from balanced truncation model order reduction as discussed by Heinkenschloss, Sorensen, and Sun in [24].

This paper is a first step toward a thorough numerical treatment of the Leray projection, which is the key to robust implementations of optimal control for flow problems. We consider a symmetric and linear approach for instationary, incompressible flow problems, i.e., the Stokes equations,

$$\left.\begin{aligned} \frac{\partial}{\partial t}\mathbf{v}(t,\mathbf{x}) - \nu\Delta\mathbf{v}(t,\mathbf{x}) + \nabla p(t,\mathbf{x}) &= 0, \\ \nabla\cdot\mathbf{v}(t,\mathbf{x}) &= 0 \end{aligned}\right\} \text{ on } (0,\infty)\times\Omega, \tag{1.1}$$

with the time $t \in (0,\infty)$, the spatial variable $\mathbf{x} \in \Omega$, the velocity field $\mathbf{v}(t,\mathbf{x}) \in \mathbb{R}^2$, the pressure $p(t,\mathbf{x}) \in \mathbb{R}$, and the viscosity $\nu \in \mathbb{R}^+$ (see, e.g., [15, subsection 2.1.2]). Additionally, we have $\Omega \subset \mathbb{R}^2$ as a bounded and connected domain with boundary $\Gamma = \partial\Omega$, some Dirichlet boundary conditions, which describe an inflow-outflow problem (see, e.g., [20, Example 5.1.1]), and appropriate initial conditions. Details about the boundary and initial conditions that we used for the numerical test are discussed at the beginning of section 4.

First, we show in section 2 that the discretized system from the feedback control approach leads to differential algebraic equations. After showing why the projection idea of [24] can be used as numerical realization of the *Leray* projector, we end up with large-scale saddle point systems as the major ingredients for computing the Riccati feedback via this projection approach. In section 3, we introduce the solution strategy for these saddle point systems based on the ideas of [20, 47]. Afterwards, we show numerical examples in section 4 and summarize the ideas and results in section 5.

2. Discretization. As is common in instationary control problems [16, 24, 4, 3], we apply the *method of lines* [46] to the *Stokes equations*, which means that we discretize (1.1) with a mixed finite element method [25] in space and get the following system of differential-algebraic equations:

$$M\frac{d}{dt}\mathbf{z}(t) = A\mathbf{z}(t) + G\mathbf{p}(t) + \mathbf{f}(t), \tag{2.1a}$$

$$0 = G^T\mathbf{z}(t), \tag{2.1b}$$

with the discretized velocity $\mathbf{z}(t) \in \mathbb{R}^{n_v}$ and pressure $\mathbf{p}(t) \in \mathbb{R}^{n_p}$, the symmetric positive definite mass matrix $M = M^T \succ 0 \in \mathbb{R}^{n_v\times n_v}$, and the symmetric negative definite system matrix $A = A^T \prec 0 \in \mathbb{R}^{n_v\times n_v}$ [20, section 1.3]. Note that in contrast to the setting used for finite element solvers for instationary systems, the system matrix in the *system theoretic framework* is on the other side of the equality sign, which leads to a switch of signs. Thus, in our setting A is negative definite. The discretized gradient $G \in \mathbb{R}^{n_v\times n_p}$ is of rank n_p in the case of an inflow-outflow problem and inf-sup stable finite elements [20, subsection 5.3]. For this reason we use P_2-P_1 *Taylor–Hood* finite elements, where we have $n_v > n_p$ [25]. The source term $\mathbf{f}(t)$ describes the feedback influence via the boundary and can be expressed as $\mathbf{f}(t) = B\mathbf{u}(t)$ [6, section 2], with the boundary control $\mathbf{u}(t) \in \mathbb{R}^{n_r}$ and the input operator $B \in \mathbb{R}^{n_v\times n_r}$ that maps the control onto the corresponding boundary nodes. Since, in general, one can observe the velocity only in parts of the domain, we add the output equation

$$\mathbf{y}(t) = C\,\mathbf{z}(t), \tag{2.1c}$$

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 selects the part of the domain where we want to measure the velocity, which in our case is a part of the outflow boundary.

Equations (2.1a)–(2.1b) represent a system of differential-algebraic equations (DAEs) of differential index two [50], written in compact form as the matrix pencil

$$\left(\begin{bmatrix} A & G \\ G^T & 0 \end{bmatrix}, \begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix} \right) \tag{2.2}$$

with a singular left-hand side coefficient matrix $\begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix}$. This symmetric matrix pencil has $n_v - n_p$ finite eigenvalues $\lambda_i \in \mathbb{R}^-$ and $2n_p$ infinite eigenvalues $\lambda_\infty = \infty$ [18, Theorem 2.1].

In contrast to ordinary differential equations, where the solution set lies on an affine subspace of the Euclidean space, the solution set of the DAEs lies on a (hidden) manifold of the Euclidean space that implies some additional difficulties referring to the solvability (see, e.g., [50]). To avoid this problem we use the idea of index reduction described in [24, section 3], which is demonstrated in the next subsection for *descriptor systems* like (2.1).

2.1. Projection method. First, we show how the idea of index reduction by Heinkenschloss, Sorensen, and Sun [24], used for balanced truncation model order reduction of descriptor systems (2.1), represents a numerical realization of the *Leray* projector. Namely, we can convert (2.1) into a generalized state space system by using the projector

$$\Pi := I - G(G^T M^{-1} G)^{-1} G^T M^{-1},$$

defined in [24, section 3]. To apply the analytic approach by Raymond [40] using the *Leray* projection, we need a self-adjoint projector onto the space of divergence-free velocity functions. Hence, if $\mathbf{w}(t)$ lies in the range of such a projector, the algebraic constraint $G^T \mathbf{w}(t) = 0$ is fulfilled. Since $\operatorname{range}\left(\Pi^T\right) = \operatorname{null}\left(G^T\right)$, Π^T has the property $\Pi^T \mathbf{w}(t) = \mathbf{w}(t)$. We call

$$(\mathbf{x}, \mathbf{y})_{\mathbb{R}^{n_v}} := \mathbf{x}^T \mathbf{y} \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^{n_v}$$

the inner product of the Euclidean space $\mathbb{R}^{n_v}$ and

$$\langle \mathbf{x}, \mathbf{y} \rangle_M := (M\mathbf{x}, \mathbf{y})_{\mathbb{R}^{n_v}} = \mathbf{x}^T M \mathbf{y} \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^{n_v}$$

the M-inner product for a symmetric matrix $M \in \mathbb{R}^{n_v \times n_v}$. It is easily shown that

$$\langle \Pi^T \mathbf{x}, \mathbf{y} \rangle_M = (M\Pi^T \mathbf{x}, \mathbf{y})_{\mathbb{R}^{n_v}} = (M\mathbf{x}, \Pi^T \mathbf{y})_{\mathbb{R}^{n_v}} = \langle \mathbf{x}, \Pi^T \mathbf{y} \rangle_M \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^{n_v}$$

such that Π^T is self-adjoint with respect to the M-inner product. Additionally, $\operatorname{null}\left(\Pi^T\right) = \operatorname{range}\left(M^{-1}G\right)$, which represents the curl-free components in the nullspace of Π^T. Finally, Π^T decomposes $\mathbb{R}^{n_v}$ into the direct sum of the two spaces $\mathbf{H}_{\mathrm{div0}}(\mathbb{R}^{n_v})$, the space of divergence-free vector functions, and $\mathbf{H}_{\mathrm{curl0}}(\mathbb{R}^{n_v})$, the space of curl-free vector functions [19]. Because projectors are unique we have Π^T as the discrete version of the *Leray* projector with respect to the M-inner product, which is the discrete version of the L_2-inner product.

The projector Π^T ensures that the solution fulfills the algebraic equation (2.1b) and simultaneously resides in the correct solution manifold, the so-called *hidden manifold* [50] defined by

$$\mathbf{0} = G^T M^{-1} A\mathbf{z}(t) + G^T M^{-1} G\mathbf{p}(t) + G^T M^{-1} B\mathbf{u}(t).$$

Thus, the system (2.1) reduces to

$$\Pi M \Pi^T \frac{d}{dt}\mathbf{z}(t) = \Pi A \Pi^T \mathbf{z}(t) + \Pi B\mathbf{u}(t), \tag{2.3a}$$

$$\mathbf{y}(t) = C\Pi^T \mathbf{z}(t), \tag{2.3b}$$

with $\Pi^T\mathbf{z}(t) = \mathbf{z}(t)$. Because the nullspace of Π is nontrivial the matrix $\Pi M \Pi^T$ is not invertible. Therefore, for $\Theta_l, \Theta_r \in \mathbb{R}^{n_v \times (n_v - n_p)}$ satisfying

$$\Theta_l^T \Theta_r = I_{(n_v - n_p)},$$

we consider the decomposition

$$\Pi = \Theta_l \Theta_r^T. \tag{2.4}$$

Since the pencil (2.2) has $n_v - n_p$ finite eigenvalues [18], (2.4) is computable using, for example, the singular value decomposition [1]. If we substitute this decomposition into (2.3), we obtain

$$\Theta_r^T M \Theta_r \frac{d}{dt}\tilde{\mathbf{z}}(t) = \Theta_r^T A \Theta_r \tilde{\mathbf{z}}(t) + \Theta_r^T B\mathbf{u}(t), \tag{2.5a}$$

$$\mathbf{y}(t) = C\Theta_r \tilde{\mathbf{z}}(t), \tag{2.5b}$$

with $\tilde{\mathbf{z}} = \Theta_l^T \mathbf{z} \in \mathbb{R}^{n_v - n_p}$. After replacing $\mathcal{M} = \Theta_r^T M \Theta_r$, $\mathcal{A} = \Theta_r^T A \Theta_r$, $\mathcal{B} = \Theta_r^T B$, and $\mathcal{C} = C\Theta_r$, (2.5) yields

$$\mathcal{M}\frac{d}{dt}\tilde{\mathbf{z}}(t) = \mathcal{A}\tilde{\mathbf{z}}(t) + \mathcal{B}\mathbf{u}(t), \tag{2.6a}$$

$$\mathbf{y}(t) = \mathcal{C}\tilde{\mathbf{z}}(t) \tag{2.6b}$$

as a generalized state space system with a symmetric positive definite mass matrix $\mathcal{M}$.

In [24], Heinkenschloss, Sorensen, and Sun apply balanced truncation model order reduction to the system in (2.6), which requires computing the controllability and observability Gramians $\tilde{P}, \tilde{Q} \in \mathbb{R}^{(n_v - n_p)\times(n_v - n_p)}$, which solve the generalized Lyapunov equations

$$\mathcal{A}\tilde{P}\mathcal{M}^T + \mathcal{M}\tilde{P}\mathcal{A}^T = -\mathcal{B}\mathcal{B}^T, \tag{2.7}$$

$$\mathcal{A}^T\tilde{Q}\mathcal{M} + \mathcal{M}^T\tilde{Q}\mathcal{A} = -\mathcal{C}^T\mathcal{C} \tag{2.8}$$

[24, section 4]. In the next two subsections we show that we have to solve similar equations for the Riccati-based feedback approach. It is clear that for computational purposes we do not want to form Π *explicitly*, and we certainly cannot compute the decomposition presented in (2.4). We will later see how we can work with (2.6) *implicitly* by solving certain saddle point problems.

2.2. The feedback control approach. For an asymptotic stabilization of the Stokes equations (1.1), we apply the *linear quadratic regulator approach (LQR)* to system (2.6). An introduction to LQR for state space systems can be found in [31]. As we consider the generalized state space system (2.6) with $\mathcal{M} \neq I$, we have to modify the results as carried out in [45, Chapter 5.2]. In summary, we minimize

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

subject to (2.6), which can be achieved by the optimal control defined by

$$\mathbf{u}_*(t) = -\underbrace{\mathcal{B}^T X \mathcal{M}}_{=:\mathcal{K}} \tilde{\mathbf{z}}_*(t) = -\mathcal{K} \tilde{\mathbf{z}}_*(t). \tag{2.10}$$

Here, we define the feedback operator $\mathcal{K}$ with X as the solution of the *generalized algebraic Riccati equation (GARE)*

$$0 = \mathcal{C}^T \mathcal{C} + \mathcal{A}^T X \mathcal{M} + \mathcal{M}^T X \mathcal{A} - \mathcal{M}^T X \mathcal{B} \mathcal{B}^T X \mathcal{M} =: \mathfrak{R}(X). \tag{2.11}$$

Thus, we have to solve such a nonlinear matrix equation to get the optimal control $\mathbf{u}_*(t)$. One way to solve this GARE is described in the next subsection.

2.3. Solving the generalized algebraic Riccati equation. A common way to solve the nonlinear matrix equation (2.11) is a Newton-type iteration as described in [2, 29]. The Newton system at step m is given by

$$X^{(m+1)} = X^{(m)} + N^{(m)}, \tag{2.12a}$$

with the update computed via

$$\mathfrak{R}'|_{X^{(m)}}(N^{(m)}) = -\mathfrak{R}(X^{(m)}), \tag{2.12b}$$

where $\mathfrak{R}'|_{X^{(m)}}$ is the Fréchet derivative of the Riccati operator (2.11) at $X^{(m)}$ defined as

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

Therefore, we have to solve

$$\begin{aligned} &(\mathcal{A} - \mathcal{B}\mathcal{B}^T X^{(m)} \mathcal{M})^T N^{(m)} \mathcal{M} + \mathcal{M}^T N^{(m)} (\mathcal{A} - \mathcal{B}\mathcal{B}^T X^{(m)} \mathcal{M}) \\ &\quad = -\mathcal{C}^T \mathcal{C} - \mathcal{A}^T X^{(m)} \mathcal{M} - \mathcal{M}^T X^{(m)} \mathcal{A} + \mathcal{M}^T X^{(m)} \mathcal{B}\mathcal{B}^T X^{(m)} \mathcal{M} \end{aligned} \tag{2.13}$$

to compute the update $N^{(m)} = X^{(m+1)} - X^{(m)}$. If we plug this expression into (2.13), we obtain the generalized Lyapunov equation

$$(\mathcal{A}^{(m)})^T X^{(m+1)} \mathcal{M} + \mathcal{M}^T X^{(m+1)} \mathcal{A}^{(m)} = -(\mathcal{W}^{(m)})^T \mathcal{W}^{(m)}, \tag{2.14}$$

with $\mathcal{A}^{(m)} = \mathcal{A} - \mathcal{B}\mathcal{B}^T X^{(m)} \mathcal{M}$ and a right-hand side split into the low-rank factors $\mathcal{W}^{(m)} = \begin{bmatrix} \mathcal{C} \\ \mathcal{B}^T X^{(m)} \mathcal{M} \end{bmatrix}$ (see [28]). Equation (2.14) has the same structure as (2.8) and has to be solved at each Newton step. Hence, the index reduction method in [24] is applicable.

Algorithm 1. Generalized low-rank Cholesky factor ADI iteration (G-LRCF-ADI).

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

A solution strategy for this kind of equation is the *low-rank ADI iteration* [30, 8], which is extended for the generalized case in [7] as shown in Algorithm 1.

Note that it is a basic ADI result that (optimal) ADI shifts q_i need to be contained in the convex hull of the spectrum of the DAEs (see, e.g., [48]). Thus, we need $q_i \in \mathbb{R}^-$. For details of an effective implementation we refer the reader to [45].

Combining the Newton iteration (2.12), as an outer iteration, and the G-LRCF-ADI (Algorithm 1), as an inner iteration, yields the *generalized low-rank Cholesky factor Newton method (G-LRCF-NM)* [8, 6, 9]. In lines 1 and 4 of Algorithm 1 large-scale linear systems of equations involving the projected matrices have to be solved. Both have the following structure:

$$\left((\mathcal{A}^{(m)})^T + q_i\mathcal{M}\right)\Lambda = \mathcal{Y}, \tag{2.15}$$

with different right-hand sides $\mathcal{Y}$. Because one wants to build neither the dense projector Π^T nor its Θ-decomposition, we recall the results in [24, section 5] which state that the solution of the Θ-projected equation (2.15) is also a solution of the Π-projected equation

$$\Pi\left(A - MX^{(m)}BB^T + q_iM\right)\Pi^T\Lambda = \Pi Y. \tag{2.16}$$

With [24, Lemma 5.2], one has to solve the equivalent linear system

$$\underbrace{\begin{bmatrix} A - MX^{(m)}BB^T + q_iM & G \\ G^T & 0 \end{bmatrix}}_{=:\tilde{\mathbf{A}}} \begin{bmatrix} \Lambda \\ * \end{bmatrix} = \begin{bmatrix} Y \\ 0 \end{bmatrix} \tag{2.17}$$

instead of system (2.16). The complete process for computing the feedback operator $K = B^TXM$ is shown in Algorithm 2.

The linear system (2.17) has to be solved in every ADI step. Note that despite the fact that the discretized diffusion operator A is symmetric, the $(1,1)$-block of the saddle point problem (2.17) is nonsymmetric. Furthermore, note that every Newton step consists of several ADI steps. In the remainder of this paper we show how we can efficiently solve (2.17).

3. Solving large-scale saddle point systems. Linear systems of the form (2.17) are often referred to as *saddle point systems*. A comprehensive overview of the numerical solution of saddle point systems is given in [10]. In the following we discuss the properties of the linear system (2.17) and our strategy for its efficient solution.

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

Input: M, A, G, B, C, and shift parameters $q_i \in \mathbb{R}^- : i = 1, \ldots, n_{\text{ADI}}$
Output: feedback operator K

$K^0 = [\,]$
for $m = 1, 2, \ldots, n_{\text{Newton}}$ **do**
  $W^{(m)} = \begin{bmatrix} C \\ (K^{(m-1)})^T \end{bmatrix}$
  Get V_1 by solving

$$\begin{bmatrix} A - (K^{(m-1)})^T B^T + q_1 M & G \\ G^T & 0 \end{bmatrix} \begin{bmatrix} V_1 \\ * \end{bmatrix} = \begin{bmatrix} \sqrt{-2\,\mathrm{Re}\,(q_1)}\,(W^{(m)})^T \\ 0 \end{bmatrix}$$

  $K_1^{(m)} = B^T V_1 V_1^T M$
  for $i = 2, 3, \ldots, n_{\text{ADI}}$ **do**
    Get $\tilde{V}$ by solving

$$\begin{bmatrix} A - (K^{(m-1)})^T B^T + q_i M & G \\ G^T & 0 \end{bmatrix} \begin{bmatrix} \tilde{V} \\ * \end{bmatrix} = \begin{bmatrix} M V_{i-1} \\ 0 \end{bmatrix}$$

    $V_i = \sqrt{\mathrm{Re}\,(q_i)/\,\mathrm{Re}\,(q_{i-1})}(V_{i-1} - (q_i + \overline{q_{i-1}})\tilde{V})$
    $K_i^{(m)} = K_{i-1}^{(m)} + B^T V_i V_i^T M$
    if $(\frac{||K_i^{(m)} - K_{i-1}^{(m)}||_F}{||K_i^{(m)}||_F} < tol_{\text{ADI}})$ **then**
      break
    end if
  end for
  $K^{(m)} = K_{n_{\text{ADI}}}^{(m)}$
  if $(\frac{||K^{(m)} - K^{(m-1)}||_F}{||K^{(m)}||_F} < tol_{\text{Newton}})$ **then**
    break
  end if
end for
$K = K^{n_{\text{Newton}}}$

3.1. Properties of the saddle point system. The saddle point system arising from the feedback control approach for the Stokes equations is of the form (2.17). Although the matrices A, M, and G are sparse, the low-rank product $K^T B^T$ is dense, and the $(1,1)$-block of $\tilde{\mathbf{A}}$ also becomes dense, making Algorithm 2 inefficient. To avoid this, we can write system (2.17) in the form of a low-rank update

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

or, more compactly,

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

We then use the *Sherman–Morrison–Woodbury* formula [22] for the evaluation of (3.1), that is,

$$(\mathbf{A} - \mathbf{K}^T \mathbf{B}^T)^{-1} = \left(I_{n_v} + \mathbf{A}^{-1}\mathbf{K}^T (I_{n_r} - \mathbf{B}^T \mathbf{A}^{-1} \mathbf{K}^T)^{-1} \mathbf{B}^T\right) \mathbf{A}^{-1}.$$

This means we have to solve for $\mathbf{A}$ and the small dense matrix $(I_{n_r} - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{K}^T)$ of size $n_r \ll n_v$ instead of solving with $\tilde{\mathbf{A}}$. Furthermore, we have to solve for $\mathbf{A}$ with the right-hand side $\mathbf{K}^T$. Therefore, we just add $\mathbf{K}^T$ as additional n_r columns in the right-hand side matrix $\mathbf{Y}$, which results in $\begin{bmatrix} Y & K^T \\ 0 & 0 \end{bmatrix} =: \begin{bmatrix} \tilde{Y} \\ 0 \end{bmatrix}$. Because we consider two boundary control parameters, $n_r = 2$. Finally, we obtain the saddle point system

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

with $M = M^T \succ 0$, $A = A^T \prec 0$, and $q_i \in \mathbb{R}^-$, which means the $(1,1)$-block of $\mathbf{A}$ is symmetric negative definite (see sections 2 and 2.3), whereas the whole matrix $\mathbf{A}$ is indefinite. This system has to be solved for many different right-hand sides and a different shift parameter q_i in every ADI step. Our strategy for computing the solution of (3.2) is discussed next.

3.2. Preconditioned iterative solvers. To solve the system (3.2), we use iterative methods, instead of direct solvers, because the size of the whole system $n = n_v + n_p$ becomes too large for usual finite element discretizations. For three-dimensional problems the fill-in generated by a direct method often allows only small problems to be solved, meaning that the use of iterative methods is imperative. Nevertheless, in their basic form the performance of iterative methods will deteriorate with decreasing mesh-size. This can be avoided if a suitable preconditioner $\mathbf{P} \in \mathbb{R}^{n \times n}$ is introduced and the modified system

$$\mathbf{P}^{-1}\mathbf{A}\mathbf{x} = \mathbf{P}^{-1}\mathbf{b}$$

is solved instead (see [43, 20]). The preconditioner can be chosen as the isomorphism mapping $\mathbf{P} : \mathcal{X}^* \to \mathcal{X}$ if $\mathbf{A} : \mathcal{X} \to \mathcal{X}^*$ is an isomorphism [34]. Based on the results of [35] we design a preconditioner to give a small number of eigenvalue clusters for $\mathbf{P}^{-1}\mathbf{A}$.

If we want to use a symmetric iterative solver (e.g., MINRES [37]), we have to use a symmetric positive definite preconditioner, such as the one given by

$$\mathbf{P} = \begin{bmatrix} -P_F & 0 \\ 0 & -P_{SC} \end{bmatrix}$$

[20, section 6.1], where P_F approximates $F := A + p_i M \prec 0$ and $P_{SC} := G^T F^{-1} G$ is the *Schur complement*, which we cannot form explicitly as this would require the dense matrix $F^{-1} \in \mathbb{R}^{n_v \times n_v}$. A good approximation for the steady Stokes case with moderate viscosities is $P_{SC} \approx -\frac{1}{\nu} M_p$, with M_p the mass matrix defined on the pressure space (see, e.g., [20, section 8.2]). We will call this version

$$\hat{\mathbf{P}}_{\mathbf{M}} := \begin{bmatrix} -P_F & 0 \\ 0 & \frac{1}{\nu} M_p \end{bmatrix}.$$

Note that the matrix F has a form similar to the matrix obtained when the transient Stokes equation is discretized. Hence, the derivation presented now will somehow resemble preconditioning for transient Stokes systems and shows another way to approximate P_{SC}. Beforehand, we will show how to use a nonsymmetric iterative method. The obvious advantage is that we can extend our method to nonsymmetric systems that arise for more general Navier–Stokes systems, which is the long-term goal of these investigations.

A nonsymmetric iterative method for the block structured and potentially nonsymmetric matrix

$$\mathbf{F} := \begin{bmatrix} F & G \\ G^T & 0 \end{bmatrix} \text{ with } F = A + q_i M$$

is GMRES [44]. Based on the ideas in [20, section 8.1] we consider the block structured nonsymmetric left preconditioner

$$\mathbf{P_G} := \begin{bmatrix} P_F & 0 \\ G^T & -P_{SC} \end{bmatrix} \Rightarrow \mathbf{P_G}^{-1} = \begin{bmatrix} P_F^{-1} & 0 \\ P_{SC}^{-1} G^T P_F^{-1} & -P_{SC}^{-1} \end{bmatrix}.$$

Note that in the Stokes case it is possible to use the block-triangular preconditioner $\mathbf{P_G}$ within a symmetric iterative method. Namely, a variant of the CG method [43] introduced by Bramble and Pasciak in [13] can be used, which is only slightly more expensive than MINRES for the same problem.

Applying $\mathbf{P_G}^{-1}$ from the left to $\mathbf{F}$ gives

$$\mathbf{P_G}^{-1}\mathbf{F} = \begin{bmatrix} P_F^{-1} F & 0 \\ P_{SC}^{-1} G^T P_F^{-1} F - P_{SC}^{-1} G^T & P_{SC}^{-1} G^T P_F^{-1} G \end{bmatrix}. \tag{3.3}$$

For $P_F = F$ and $P_{SC} = G^T F^{-1} G$, (3.3) yields

$$\begin{bmatrix} F^{-1} F & 0 \\ P_{SC}^{-1} G^T F^{-1} F - P_{SC}^{-1} G^T & P_{SC}^{-1} G^T F^{-1} G \end{bmatrix} = \begin{bmatrix} I_{n_v} & 0 \\ 0 & I_{n_p} \end{bmatrix}. \tag{3.4}$$

As before, we cannot form $P_{SC} \in \mathbb{R}^{n_p \times n_p}$. Instead we apply the least squares commutator approach based on [20, section 8.2]. Therefore, we consider a shifted diffusion operator on the velocity space

$$\mathcal{F} = -\nu \nabla^2 + q \cdot I$$

and suppose that the analogous operator on the pressure space also exists, that is,

$$\mathcal{F}_p = (-\nu \nabla^2 + q \cdot I)_p.$$

We then define the least squares commutator of the shifted diffusion operators with the gradient operator

$$\mathcal{E} = (\mathcal{F})\nabla - \nabla(\mathcal{F}_p),$$

which should become small in some sense. If we plug in the discrete versions of the operators, we obtain

$$\mathcal{E}_h = (M^{-1} F) M^{-1} G - M^{-1} G (M_p^{-1} F_p).$$

Premultiplying this by $G^T F^{-1} M$ and postmultiplying by $F_p^{-1} M_p$ yields

$$G^T M^{-1} G F_p^{-1} M_p \approx G^T F^{-1} G = P_{SC}.$$

In general it is infeasible to work with $G^T M^{-1} G$ as it is a large dense matrix. In [20, section 5.5.1] it is shown that this matrix is spectrally equivalent to the Laplacian S_p defined on the pressure space if an inf-sup stable discretization is used and the

boundary conditions are considered in a proper way for an inflow-outflow problem [20, section 8.2]. Hence, we get

$$P_{SC} \approx S_p F_p^{-1} M_p \quad \Rightarrow \quad P_{SC}^{-1} \approx M_p^{-1} F_p S_p^{-1},$$

with F_p, the shifted system matrix, defined on the pressure space. Thus, one has to solve a pure Neumann problem, defined on the pressure space and formally denoted as S_p^{-1} (as in [12]), multiply this with the system matrix F_p once, and solve a linear system with symmetric positive definite M_p, to apply the Schur complement approximation. A similar approach is applied to the incompressible Navier–Stokes equations in [11].

Note that for the symmetric Stokes case $F_p = A_p + qM_p$, the Schur complement approximation becomes slightly easier [47, section 3] because the system matrix A_p is just the negative and scaled stiffness matrix $-\nu S_p$ such that

$$P_{SC} \approx S_p(-\nu S_p + qM_p)^{-1} M_p \tag{3.5}$$

$$\Rightarrow \quad P_{SC}^{-1} \approx M_p^{-1}(-\nu S_p + qM_p) S_p^{-1} = -\nu M_p^{-1} + qS_p^{-1}. \tag{3.6}$$

Summarizing, we just solve a pure Neumann problem S_p^{-1} [12] and a linear system with the mass matrix M_p. To increase the efficiency of this method we approximate both solutions, which is sufficient for preconditioning. In detail, we use a *Chebyshev* semi-iteration based on [47, Algorithm 3] to solve with M_p [49]. Furthermore, we use an algebraic *multigrid* method (AMG) [23, 42] to apply S_p. To solve the problem of indefiniteness of S_p, we just pin a boundary node in S_p (see, e.g., [12]). Afterwards we use the AMG package, provided by [26].

Further details regarding this kind of Schur-complement approximation can be found in [17] for generalized Stokes systems, in [32, 14] for steady and unsteady Stokes systems, and in [34] for general partial differential equations. For unsteady Stokes problems the above approximation is known as the Cahouet–Chabard preconditioner [17].

In (3.4) we assumed F to be the best choice for P_F. To get an efficient preconditioner we apply the AMG approximation [26] of F as well.

Again, we can use the simple approximation $P_{SC} \approx -\frac{1}{\nu} M_p$ and define the preconditioners for the nonsymmetric iterative method as

$$\hat{\mathbf{P}}_{\mathbf{G}} := \begin{bmatrix} P_F & 0 \\ G^T & \frac{1}{\nu} M_p \end{bmatrix} \quad \text{and} \quad \mathbf{P}_{\mathbf{G}} := \begin{bmatrix} P_F & 0 \\ G^T & (\nu M_p^{-1} - q_i S_p^{-1})^{-1} \end{bmatrix}.$$

We compare both methods with the appropriate preconditioner in the next section. Additionally, we discuss some problems with the nested iterations and remark on the numerical realization.

4. Numerical examples. The example we use for our domain Ω is the *von Kármán vortex street* depicted in Figure 4.1. The Stokes equations (1.1) are solved with initial condition $\mathbf{v}(0, \mathbf{x}) = 0$. We have a parabolic inflow

$$v(t, \mathbf{x}) = \begin{bmatrix} 4 \cdot (1 - y) \cdot y \\ 0 \end{bmatrix} \quad \text{on } \Gamma_{\text{in}}, \tag{4.1}$$

with a maximum value of 1.0 at Γ_{in}, which has a diameter $d_{\text{in}} = 1$. The fluid passes an elliptic obstacle with the center at the coordinates $(1, 0.5)$, which has a width of

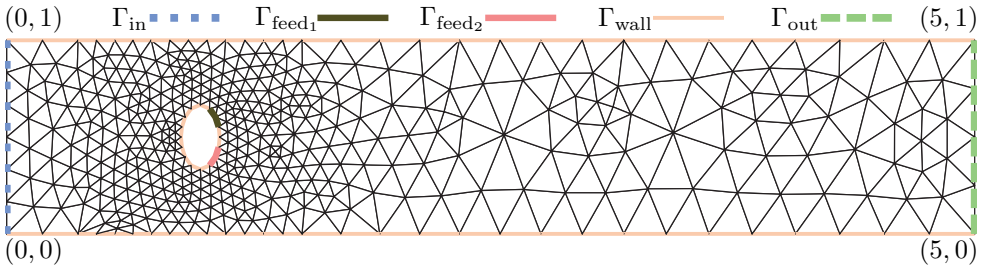


FIG. 4.1. *Coarsest discretization of* von Kármán vortex street *with coordinates and boundary conditions.*

$\frac{1}{5}\,d_{\text{in}}$ and a height of $\frac{1}{3}\,d_{\text{in}}$. The fluid flows out of the domain at Γ_{out} without any barrier which means that we have a so-called *do-nothing condition*

$$-\nu\nabla\mathbf{v}(t,\mathbf{x})\cdot n + p(t,\mathbf{x})n = 0 \quad \text{on } \Gamma_{\text{out}}. \tag{4.2}$$

To influence the flow field we can blow in or exhaust fluid on the back of the obstacle via $\Gamma_{\text{feed}_1}, \Gamma_{\text{feed}_2}$, which are realized as Dirichlet conditions

$$\mathbf{v}(t,\mathbf{x}) = \mathbf{g}_{\text{feed}_i} \quad \text{on } \Gamma_{\text{feed}_i},\ i = 1, 2. \tag{4.3}$$

Naturally, we impose no-slip conditions

$$\mathbf{v}(t,\mathbf{x}) = 0 \quad \text{on } \Gamma_{\text{wall}}. \tag{4.4}$$

The matrices for the numerical tests arise from a standard mixed finite element discretization (e.g., P_2-P_1 *Taylor–Hood* elements as in Figure 4.2) of Ω. Using a *Bänsch refinement* [5], we get five different magnitudes for n_v and n_p, where every second level corresponds to one level of global uniform refinement (see Table 4.1). Figure 4.1 shows the coarsest grid level 1. All computations were done with MATLAB R2010b on a 64-bit server with CPU type Intel Xeon X5650 at 2.67GHz, with 2 CPUs, 12 Cores (6 Cores per CPU), and 48 GB main memory available.

4.1. Solving the saddle point system. First, we compare MINRES and GMRES with the preconditioners defined in section 3.2. Therefore, the MATLAB implementations of both methods are used, and the preconditioning is realized as function handles to increase the efficiency.

On the one hand, for MINRES we use the symmetric preconditioners

$$\hat{\mathbf{P}}_{\mathbf{M}} := \begin{bmatrix} -P_F & 0 \\ 0 & \frac{1}{\nu}M_p \end{bmatrix}, \quad \mathbf{P}_{\mathbf{M}} := \begin{bmatrix} -P_F & 0 \\ 0 & (\nu M_p^{-1} - q_i S_p^{-1})^{-1} \end{bmatrix}.$$

On the other hand, for GMRES we use

$$\hat{\mathbf{P}}_{\mathbf{G}} := \begin{bmatrix} P_F & 0 \\ G^T & \frac{1}{\nu}M_p \end{bmatrix}, \quad \mathbf{P}_{\mathbf{G}} := \begin{bmatrix} P_F & 0 \\ G^T & (\nu M_p^{-1} - q_i S_p^{-1})^{-1} \end{bmatrix}.$$

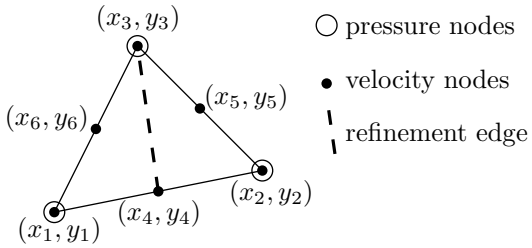


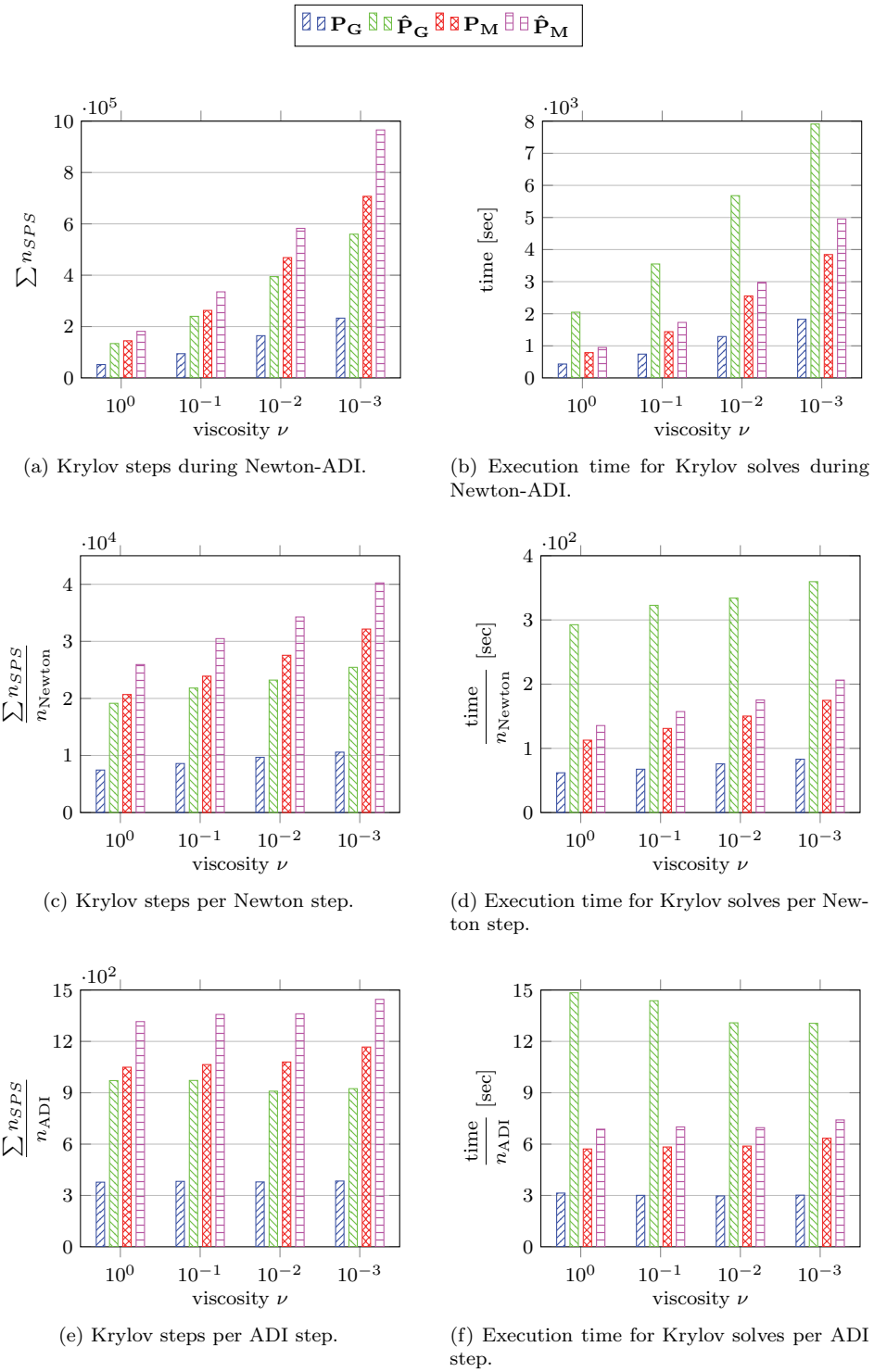
FIG. 4.2. P_2-P_1 Taylor–Hood *element.*

TABLE 4.1
Levels of refinement.

Level	n_v	n_p
1	3452	453
2	8726	1123
3	20512	2615
4	45718	5783
5	99652	12566

We solve the Newton-ADI iteration with all methods for an iteration tolerance of $tol_{\text{solve\_SPS}} = 10^{-12}$. This computation includes 7–22 Newton steps with 13–47 ADI steps. Therein, we have to solve for 7–11 right-hand sides the system (3.2) in every ADI step. In Figure 4.3 we have six subplots, where we show the total number of Krylov steps over all these right-hand sides on the left and the execution time of the Krylov solves on the right. Thereby, Figures 4.3(a) and 4.3(b) show the total number. GMRES with $\mathbf{P_G}$ wins this comparison for each configuration, although every GMRES step needs a little bit more time than a MINRES step, as is visualized in Table 4.2. Note that the unexpectedly high times for each Krylov step with $\hat{\mathbf{P}}_\mathbf{G}$ are easily explained by the growing fraction spent in the orthogonalization phase since in average more Krylov steps are needed in this configuration. Obviously, the number of Krylov solves increases if the viscosity decreases. This issue comes from the increasing number of Newton and ADI steps. Therefore, we scaled the number of Krylov solves and the time with the number of Newton steps in Figures 4.3(c) and 4.3(d). Finally, we scaled the values with the number of ADI steps in Figures 4.3(e) and 4.3(f). This last row in Figure 4.3 is the most important for us. Again, GMRES with $\mathbf{P_G}$ is the best choice, and it is nearly independent of the viscosity. Our tests also showed that after a certain point all methods except GMRES with $\mathbf{P_G}$ showed stagnation of the residual norm. This effect occurs due to the Schur complement approximation not being sufficiently accurate (see [36]). Moreover, we observed that this high accuracy is not needed to achieve our main goal, which is to solve the Newton-ADI iteration (Algorithm 2). The influence of the accuracy of the saddle point solvers on the convergence of the whole algorithm is presented in the next subsection.

4.2. Nested iteration. In Algorithm 2 we have a nested iteration with the outermost Newton iteration, the central ADI iteration, and finally the innermost iteration, where the saddle point system (3.2) has to be solved at every ADI step. To compute one Newton step (2.12b) we need a number of ADI steps to reach the stopping criterion for the Newton iteration $tol_{\text{Newton}} \approx 5 \cdot 10^{-5}$.



(a) Krylov steps during Newton-ADI.

(b) Execution time for Krylov solves during Newton-ADI.

(c) Krylov steps per Newton step.

(d) Execution time for Krylov solves per Newton step.

(e) Krylov steps per ADI step.

(f) Execution time for Krylov solves per ADI step.

FIG. 4.3. *Comparison of the different methods with desired tolerances:* $tol_{solve\_SPS} = 10^{-12}$, $tol_{Newton} = 5 \cdot 10^{-5}$, $tol_{ADI} = 5 \cdot 10^{-7}$ *for refinement level* 1.

TABLE 4.2
Average times for single Krylov step ($tol_{SPS} = 10^{-12}$, refinement level 1).

Preconditioner	∅ time for Krylov step	∅ time for $\mathbf{P}^{-1}\mathbf{x}$	∅ time for $\mathbf{Ax}$
GMRES $\mathbf{P_G}$	$5.178 \cdot 10^{-2}$ sec.	$3.681 \cdot 10^{-3}$ sec.	$1.231 \cdot 10^{-3}$ sec.
GMRES $\hat{\mathbf{P}}_\mathbf{G}$	$6.714 \cdot 10^{-2}$ sec.	$3.656 \cdot 10^{-3}$ sec.	$1.245 \cdot 10^{-3}$ sec.
MINRES $\mathbf{P_M}$	$3.645 \cdot 10^{-2}$ sec.	$3.618 \cdot 10^{-3}$ sec.	$1.223 \cdot 10^{-3}$ sec.
MINRES $\hat{\mathbf{P}}_\mathbf{M}$	$3.431 \cdot 10^{-2}$ sec.	$3.346 \cdot 10^{-3}$ sec.	$1.227 \cdot 10^{-3}$ sec.

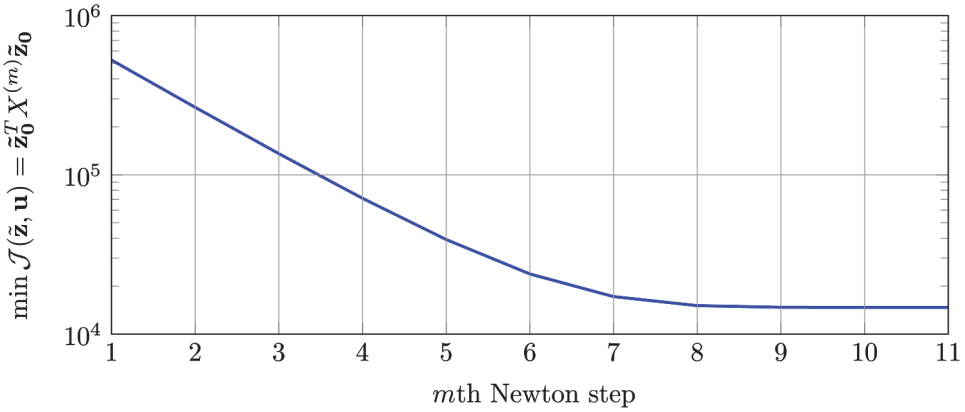


FIG. 4.4. *Evolution of optimal costs during the Newton iteration ($\nu = 10^{-1}$, refinement level 1).*

To visualize the convergence we show the evolution of the optimal costs, that is, the value of the cost functional (2.9), in Figure 4.4. This convergence depends only on the chosen cost functional (2.9) and is thereby dependent on the problem conditions.

The convergence of the ADI iteration depends heavily on the shift parameters q_i. We use the heuristic *Penzl* shifts [38] or the *Wachspress* shifts [48]. Both approaches achieve similar results, and computation times are lower for the *Penzl* shifts, so we use this method for the remaining computations with $tol_{\text{ADI}} \approx 5 \cdot 10^{-7}$.

Depending on how accurately the saddle point system is solved at every ADI step, we need more or fewer ADI steps to reach this tolerance. In Table 4.3 we compare the number of ADI steps and the required computation time in relation to the achieved tolerance $tol_{\text{solve\_SPS}}$ for the methods and preconditioners defined before.

We computed these results for different viscosities and summarized them in different subtables. In the bottom line of every subtable we put the number of Newton and ADI steps for a direct solver as a reference level, while assuming that the direct solver reaches the highest accuracy in the ADI iteration. The best configurations of each method are underlined, and the very best for each viscosity is plotted in boldface. We notice that the configuration of GMRES with $\mathbf{P_G}$ is the best choice for all configurations. If we concentrate on the first column of each subtable, we see that the fastest method needs only as many Newton steps as the direct solver needs. But it is not necessary to have the smallest number of ADI steps to be the fastest method. Summarizing, it shows that it is not easy to determine the best configuration. An extension of the *inexact Newton* idea by [21] onto this nested iteration is one of our future goals. However, it is clear that a minimal accuracy for the ADI, and in turn for GMRES and MINRES, is required to guarantee convergence of the Newton loop.

TABLE 4.3

Comparison of influence between tolerance for Krylov solver (tol_{SPS}) and the number of Newton steps (n_N), number of ADI steps (n_A), and time (in seconds) for refinement level 1 *and different viscosities ν during Newton-ADI iteration.*

	GMRES $\mathbf{P_G}$			GMRES $\hat{\mathbf{P}}_\mathbf{G}$			MINRES $\mathbf{P_M}$			MINRES $\hat{\mathbf{P}}_\mathbf{M}$		
tol_{SPS}	n_N	n_A	time	n_N	n_A	time	n_N	n_A	time	n_N	n_A	time
10^{-4}	–	–	–	–	–	–	–	–	–	–	–	–
10^{-5}	–	–	–	–	–	–	–	–	–	12	529	851
10^{-6}	–	–	–	7	179	926	–	–	–	8	334	777
10^{-7}	**7**	**144**	**279**	7	139	927	7	139	479	–	–	–
10^{-8}	7	139	304	7	138	1261	7	138	546	7	177	686
10^{-9}	7	139	342	7	138	1614	7	138	623	7	156	725
10^{-10}	7	138	374	7	138	1854	7	138	693	7	138	776
10^{-11}	7	138	405	7	138	1992	7	138	776	7	138	893
10^{-12}	7	138	442	7	138	2056	7	138	797	7	138	959
Direct	$n_\text{N} = 7$ $n_\text{A} = 138$											

(a) $\nu = 1$

	GMRES $\mathbf{P_G}$			GMRES $\hat{\mathbf{P}}_\mathbf{G}$			MINRES $\mathbf{P_M}$			MINRES $\hat{\mathbf{P}}_\mathbf{M}$		
tol_{SPS}	n_N	n_A	time	n_N	n_A	time	n_N	n_A	time	n_N	n_A	time
10^{-4}	–	–	–	–	–	–	–	–	–	–	–	–
10^{-5}	–	–	–	11	388	1375	–	–	–	16	698	1188
10^{-6}	12	343	536	11	357	1492	–	–	–	21	910	2426
10^{-7}	**11**	**273**	**504**	11	298	1527	11	262	908	13	499	1821
10^{-8}	11	247	520	11	243	1665	11	247	998	24	880	4021
10^{-9}	11	247	580	11	247	2322	11	247	1131	11	300	1523
10^{-10}	11	247	638	11	247	2950	11	247	1257	11	257	1549
10^{-11}	11	247	693	11	247	3310	11	247	1416	11	247	1651
10^{-12}	11	247	756	11	247	3564	11	247	1454	11	247	1743
Direct	$n_\text{N} = 11$ $n_\text{A} = 247$											

(b) $\nu = 10^{-1}$

	GMRES $\mathbf{P_G}$			GMRES $\hat{\mathbf{P}}_\mathbf{G}$			MINRES $\mathbf{P_M}$			MINRES $\hat{\mathbf{P}}_\mathbf{M}$		
tol_{SPS}	n_N	n_A	time	n_N	n_A	time	n_N	n_A	time	n_N	n_A	time
10^{-4}	–	–	–	–	–	–	–	–	–	–	–	–
10^{-5}	–	–	–	–	–	–	–	–	–	–	–	–
10^{-6}	17	645	1067	–	–	–	–	–	–	–	–	–
10^{-7}	17	525	1001	17	592	2478	17	454	1542	–	–	–
10^{-8}	**17**	**457**	**998**	17	508	2646	17	434	1693	18	560	2733
10^{-9}	17	434	1074	17	429	3024	17	434	1909	17	434	2536
10^{-10}	17	434	1167	17	434	4184	17	434	2118	17	434	2814
10^{-11}	17	434	1222	17	434	5217	17	434	2355	17	428	2897
10^{-12}	17	434	1312	17	434	5701	17	434	2576	17	428	3001
Direct	$n_\text{N} = 17$ $n_\text{A} = 434$											

(c) $\nu = 10^{-2}$

	GMRES $\mathbf{P_G}$			GMRES $\hat{\mathbf{P}}_\mathbf{G}$			MINRES $\mathbf{P_M}$			MINRES $\hat{\mathbf{P}}_\mathbf{M}$		
tol_{SPS}	n_N	n_A	time	n_N	n_A	time	n_N	n_A	time	n_N	n_A	time
10^{-4}	–	–	–	–	–	–	–	–	–	–	–	–
10^{-5}	–	–	–	–	–	–	–	–	–	25	2079	8477
10^{-6}	23	1266	2036	–	–	–	–	–	–	30	1932	8999
10^{-7}	22	1004	1838	–	–	–	22	723	2677	25	968	5929
10^{-8}	**22**	**686**	**1413**	23	1097	5237	22	612	2605	22	672	5001
10^{-9}	22	616	1437	22	812	4842	22	612	2926	23	631	5163
10^{-10}	22	612	1568	22	657	5207	22	606	3414	24	693	5226
10^{-11}	22	612	1707	22	606	6519	22	612	3765	23	661	5071
10^{-12}	22	606	1856	22	606	7940	22	606	3873	24	668	4984
Direct	$n_\text{N} = 22$ $n_\text{A} = 606$											

(d) $\nu = 10^{-3}$

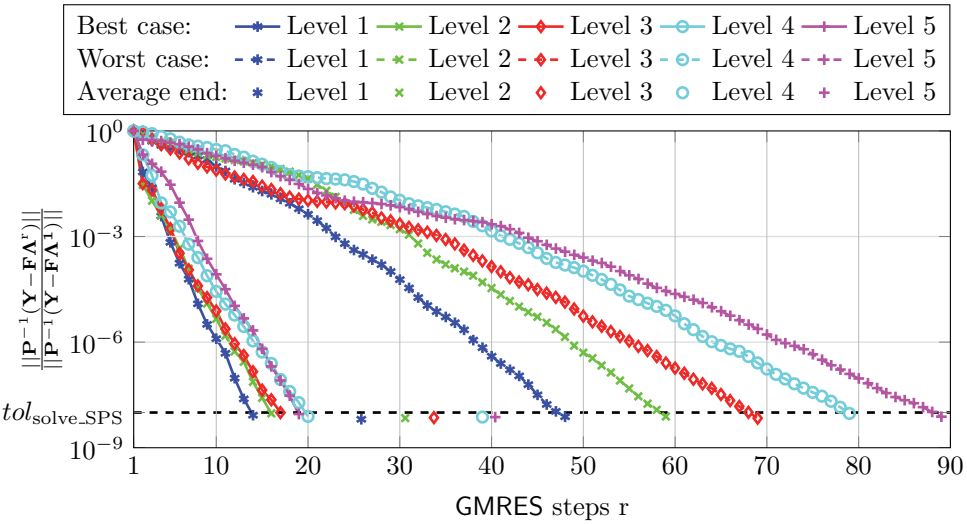


FIG. 4.5. *Relatively preconditioned residuals of* GMRES *with* $\mathbf{P_G}$ *for refinement levels of Table* 4.1 *during the Newton-ADI iteration* ($\nu = 10^{-1}$, $tol_{solve\_SPS} = 10^{-8}$).

In the tables we mark by "–" the cases where the iteration did not converge within 50 steps.

For this level of refinement the direct solver is of course much faster than our iterative method, because the number of unknowns is comparably small. Based on this observation and the fact that we want to deal with the Navier–Stokes equations in the future, we use GMRES as the iterative method in all further considerations and show more parameters which influence convergence.

4.3. Parameter dependence. First, in Figure 4.5 the relative preconditioned residuals of GMRES with $\mathbf{P_G}$ are listed for the different levels of refinement in Table 4.1. We plotted the best case and the worst case as residual vectors, and the average over the final residuals as markers. Thereby, the best cases need between 14 and 19 steps. However, in the worst case the numbers of steps varies between 48 and 89 steps. On average, the number of steps for various levels of refinement shows a very benign growth (between 26 and 40).

Figure 4.6 gives an explanation for the larger variation in the worst case. The condition number of matrix $\mathbf{A}$, for which we solve the saddle point systems, increases as we refine our mesh. In detail, the condition number roughly scales up with a factor of 10 for each refinement level. Considering this fact, we have a robust (with respect to the mesh parameter), preconditioned, iterative method for solving saddle point systems like (3.2).

Two further parameters which influence the properties of the saddle point system are the viscosity and the ADI shift. For different viscosities we get a different system matrix A and also different Schur complement approximations (3.6). The influence of the viscosity concerning the various methods was already shown in Figure 4.3(e), where we evaluated the performance of the various preconditioners. In detail, Figure 4.7(a) shows the influence of the viscosity for refinement Level 1. We averaged the number of GMRES steps over all right-hand sides which appear during the Newton-ADI iteration. We see that we need nearly the same number of iterations if we

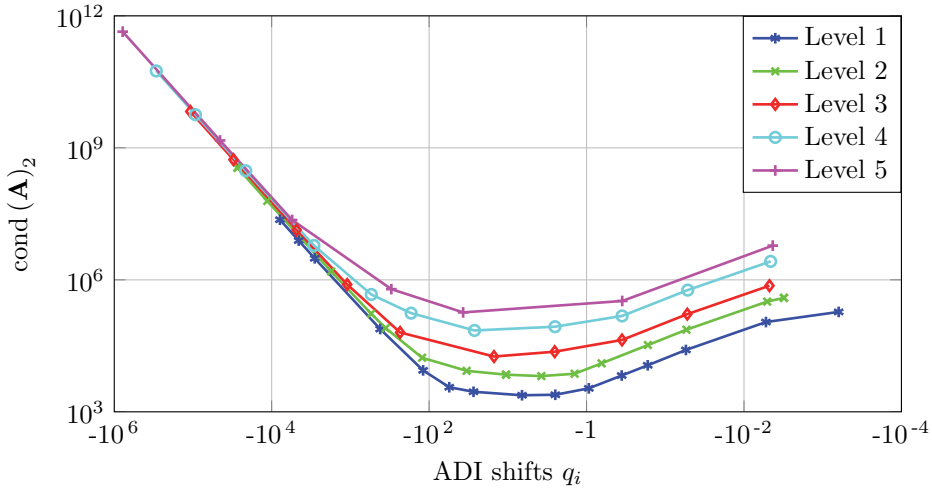
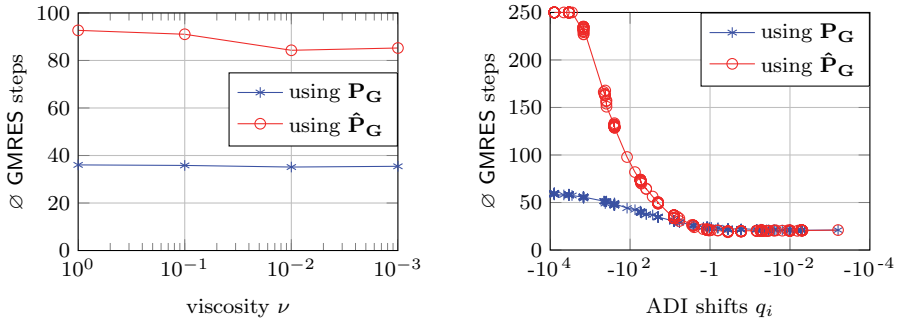


FIG. 4.6. *Condition number of* $\mathbf{A}$ *concerning the ADI shifts during the first Newton step for refinement levels of Table* 4.1 *(*$Re = 10$*).*



(a) Average number of GMRES steps for different viscosities during the Newton-ADI iteration ($tol_{\text{solve\_SPS}} = 10^{-12}$, refinement level 1).

(b) Average number of GMRES steps for the ADI shifts q_i during the Newton-ADI iteration ($\nu = 10^{-1}$, $tol_{\text{solve\_SPS}} = 10^{-12}$, refinement level 1).

FIG. 4.7. *Number of* GMRES *iterations influenced by different parameters.*

vary ν for $\mathbf{P_G}$. The same holds for $\hat{\mathbf{P}}_\mathbf{G}$. The larger number of steps for $\hat{\mathbf{P}}_\mathbf{G}$ can be explained via the more problem-dependent construction of $\mathbf{P_G}$, where the ADI shift q_i is considered as well. To make this clearer, we have a look into the second parameter in Figure 4.7(b). Again we plot the average number of GMRES steps for different ADI shifts q_i which change during Algorithm 2 in each ADI step. This change influences the saddle point system itself and also the least squares Schur complement approximation (3.6). Figure 4.7(b) shows the dependence of the number of iterations on ADI shifts q_i for refinement Level 1 and $\nu = 10^{-1}$, where q_i are the shifts used during the Newton-ADI iteration. In contrast to the viscosity, we notice that we need more iterations if we increase the absolute value of q_i for $\mathbf{P_G}$. But again we have results that are worse if we use $\hat{\mathbf{P}}_\mathbf{G}$. Unfortunately, we cannot influence the absolute value of the shift too much, because both shift concepts create a similar distribution of shifts.

Summarizing, we note that all results shown here point out that the use of GMRES with the $\mathbf{P_G}$ preconditioner is the best choice for solving the saddle point systems (3.2) arising in the Riccati feedback stabilization approach (2.10) for the Stokes equations (1.1). We believe that this preconditioner in combination with the Bramble–Pasciak CG method would perform equally well [13].

Finally, we note that in [33] a study of theoretically robust preconditioners for some partial differential equation problems was recently made. It has been observed that the performance of *multigrid* approximations to the Cahouet–Chabard preconditioner can show some dependence on the crucial parameters, but from our experiments all relevant scenarios could be dealt with by a slight increase in the number of V-cycles.

5. Conclusion and outlook. In this paper, we have considered the numerical solution of the feedback control problem for instationary Stokes flow. Following a theoretical approach of Raymond [39, 40], an LQR problem for the evolution equation resulting from the Leray–Helmholtz projection onto the divergence-free vector fields has to be solved in this setting. This leads to an operator Riccati equation which needs to be solved numerically. After discretization, one obtains an algebraic Riccati equation. Given its nature as a nonlinear system of equations, we employ Newton's method to find its unique root defining the stabilizing feedback. In each Newton step, a Lyapunov equation has to be solved for which we employ the low-rank factor ADI iteration, requiring a linear solve in each step. Employing a stable conforming finite element method like Taylor–Hood for spatial discretization, the semidiscrete problem is not consistent with the discrete Helmholtz projected flow control problem. We have shown how to overcome this difficulty using a trick suggested in [24], which we show to be equivalent to applying the discrete Leray projector to the discretized flow field. We have also demonstrated how the Newton-ADI iteration for the algebraic Riccati equation can be reformulated in this setting so that at convergence, we obtain the solution of the algebraic Riccati equation consistent with the operator Riccati equation corresponding to the Leray-projected LQR problem. As in [24], we have seen that the reformulation of the Lyapunov solution using the ADI iteration leads to the need for solving a linear system with saddle point structure, i.e., the common structure of linear systems related to Stokes flow. As these linear solves have to be solved in the innermost loop of the threefold nested iteration resulting from the Newton-ADI scheme, their solution is the bottleneck in the numerical realization of feedback control for Stokes flow. Therefore, we have focused on deriving an efficient solver for the particular saddle point problems within the ADI iteration which can be seen as nonsymmetric low-rank perturbation of the standard discrete Stokes problem.

For the saddle point problems we have discussed preconditioned Krylov solvers based on the ideas of Wathen and colleagues [20, 47]. We could derive efficient approximations to the Schur complement of the saddle point system based on the well-known commutator approach. The resulting preconditioners then bear some similarity to the Cahouet–Chabard preconditioners. We have illustrated in various examples that the iterative schemes in combination with our preconditioners give robust results with respect to the crucial parameters such as the mesh-parameter or the viscosity. We have investigated the interactions of the nested iterations with respect to the accuracies of the different solvers. Our numerical investigations have shown that a block-triangular preconditioner within GMRES provided the most satisfying results.

In the future, the feedback stabilization of flow problems described by Navier–Stokes equations will be investigated. Among the more challenging tasks will be the computation of the ADI shift parameters for the nonsymmetric linearized Navier–

Stokes operator and the computation of an initial guess for the Newton iteration (which is not necessary in the Stokes case, as a zero initial guess is feasible in this setting). As the linear systems to be solved within the ADI iteration usually have multiple right-hand sides, depending on the number of control variables, we will investigate the acceleration of the iterative solvers via block methods and Krylov subspace recycling techniques.

Acknowledgments. We thank Andy Wathen for inspiring discussions about preconditioning of iterative methods. We would also like to thank David Silvester and the two anonymous referees for their comments, which have greatly improved the quality of the paper.

REFERENCES

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, *LAPACK Users' Guide*, SIAM, Philadelphia, 1992.

[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, *An adaptive finite-element strategy for the three-dimensional time-dependent Navier-Stokes equations*, J. Comput. Appl. Math., 36 (1991), pp. 3–28.

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

[7] P. Benner, *Solving large-scale control problems*, IEEE Control Syst. Mag., 24 (2004), pp. 44–59.

[8] 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.

[9] P. Benner and J. Saak, *A Galerkin-Newton-ADI Method for Solving Large-Scale Algebraic Riccati Equations*, Preprint 090, DFG Priority Programme 1253 Optmization with Partial Differential Equations, 2010; available online from http://www.am.uni-erlangen.de/home/spp1253/wiki/images/2/28/Preprint-SPP1253-090.pdf.

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

[11] M. Benzi, M. A. Olshanskii, and Z. Wang, *Modified augmented Lagrangian preconditioners for the incompressible Navier-Stokes equations.*, Internat. J. Numer. Methods Fluids, 66 (2011), pp. 486–508.

[12] P. Bochev and R. B. Lehoucq, *On the finite element solution of the pure Neumann problem*, SIAM Rev., 47 (2005), pp. 50–66.

[13] J. H. Bramble and J. E. Pasciak, *A preconditioning technique for indefinite systems resulting from mixed approximations of elliptic problems*, Math. Comp., 50 (1988), pp. 1–17.

[14] J. H. Bramble and J. E. Pasciak, *Iterative techniques for time dependent Stokes problems*, Comput. Math. Appl., 33 (1997), pp. 13–30.

[15] J. Bühle, *Properties of Time-Dependent Stokes Flow and the Regularization of Velocity Fluctuations in Particle Suspensions*, Ph.D. thesis, Department of Physics, Philipps-Universität Marburg, Marburg, Germany, 2007.

[16] 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.

[17] J. Cahouet and J.-P. Chabard, *Some fast 3D finite element solvers for the generalized Stokes problem*, Internat. J. Numer. Methods Fluids, 8 (1988), pp. 869–895.

[18] 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.

[19] 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.
[20] 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.
[21] 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.
[22] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins University Press, Baltimore, 1996.
[23] W. Hackbusch, *Multigrid Methods and Applications*, Springer Ser. Comput. Math. 4, Springer-Verlag, Berlin, 1985.
[24] 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.
[25] 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, Huntsville, AL, 1974, pp. 121–132.
[26] *HSL* (2011). *A Collection of Fortran Codes for Large Scale Scientific Computation*, http://www.hsl.rl.ac.uk (2011).
[27] B. Kirby, *Micro- and Nanoscale Fluid Mechanics: Transport in Microfluidic Devices*, Cambridge University Press, Cambridge, UK, 2010.
[28] D. L. Kleinman, *On an iterative technique for Riccati equation computations*, IEEE Trans. Automat. Control, 13 (1968), pp. 114–115.
[29] P. Lancaster and L. Rodman, *The Algebraic Riccati Equation*, Oxford University Press, Oxford, UK, 1995.
[30] J.-R. Li and J. White, *Low rank solution of Lyapunov equations*, SIAM J. Matrix Anal. Appl., 24 (2002), pp. 260–280.
[31] A. Locatelli, *Optimal Control: An Introduction*, Birkhäuser Verlag, Basel, Boston, Berlin, 2001.
[32] Y. Maday, D. Meiron, A. T. Patera, and E. M. Rønquist, *Analysis of iterative methods for the steady and unsteady Stokes problem: Application to spectral element discretizations*, SIAM J. Sci. Comput., 14 (1993), pp. 310–337.
[33] K.-A. Mardal, J. Schöberl, and R. Winther, *A uniformly stable Fortin operator for the Taylor–Hood element*, Numer. Math., 123 (2013), pp. 537–551.
[34] K.-A. Mardal and R. Winther, *Preconditioning discretizations of systems of partial differential equations*, Numer. Linear Algebra Appl., 18 (2011), pp. 1–40.
[35] M. F. Murphy, G. H. Golub, and A. J. Wathen, *A note on preconditioning for indefinite linear systems*, SIAM J. Sci. Comput., 21 (2000), pp. 1969–1972.
[36] M. A. Olshanskii and V. Simoncini, *Acquired clustering properties and solution of certain saddle point systems*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2754–2768.
[37] C. C. Paige and M. A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629.
[38] T. Penzl, *Lyapack Users Guide*, Preprint SFB393/00-33, Preprint Series of the SFB 393, "Numerische Simulation auf massiv parallelen Rechnern," TU Chemnitz, Chemnitz, Germany, 2000; available online from http://www.tu-chemnitz.de/sfb393/sfb00pr.html.
[39] J.-P. Raymond, *Local boundary feedback stabilization of the Navier-Stokes equations*, in Control Systems: Theory, Numerics and Applications (Rome, 2005), Proceedings of Science, SISSA, 2005; available online 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 incompressible Navier-Stokes equations*, J. Math. Pures Appl. (9), 87 (2007), pp. 627–669.
[42] J. W. Ruge and K. Stüben, *Algebraic multigrid*, in Multigrid Methods, S. F. McCormick, ed., Frontiers Appl. Math. 3, SIAM, Philadelphia, 1987, pp. 73–130.
[43] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia, 2003.
[44] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.
[45] J. Saak, *Efficient Numerical Solution of Large Scale Algebraic Matrix Equations in PDE Control and Model Order Reduction*, Ph.D. thesis, Department of Mathematics, Chemnitz University of Technology, Chemnitz, Germany, 2009.
[46] W. E. Schiesser, *The Numerical Method of Lines*, Academic Press, San Diego, CA, 1991.

[47] M. Stoll and A. Wathen, *All-at-once solution of time-dependent Stokes control*, J. Comput. Phys., 232 (2013), pp. 498–515.
[48] E. L. Wachspress, *The ADI Model Problem*, Springer, New York, 2013.
[49] A. Wathen and T. Rees, *Chebyshev semi-iteration in preconditioning for problems including the mass matrix*, Electron. Trans. Numer. Anal., 34 (2008), pp. 125–135.
[50] J. Weickert, *Navier-Stokes Equations as a Differential-Algebraic System*, Preprint SFB393/96-08, Preprint Series of the SFB 393, Department of Mathematics, Chemnitz University of Technology, Chemnitz, Germany, 1996.