EFFICIENT SOLUTION OF LARGE-SCALE SADDLE POINT SYSTEMS ARISING IN FEEDBACK CONTROL OF THE STOKES EQUATIONS

HEIKO K. WEICHELT*

Abstract. To explore feedback control of flow problems we consider the Stokes equations that describe instationary, incompressible flows for low Reynolds numbers. After a standard finite element discretization we get a differential-algebraic system of differential index two. We show how to reduce this index with a projection method to get a generalized state space system, where a linear quadratic control approach can be applied. This leads to large-scale saddle point systems which have to be solved. For obtaining a fast iterative solution of those systems we derive efficient preconditioners based on the approaches due to Wathen et al. [8, 15]. The main results can be extended to non-symmetric Navier-Stokes equations.

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

1. Introduction. Stabilization of flow problems is crucial for many areas of engineering, e.g. car industry, aircraft constructing, to name a few. In this work we follow an approach for feedback stabilization of incompressible flow problems by Riccati-based feedback. The basic ideas for the numerical treatment are described in [4] which uses the analytical approach [13] for the linearized *Navier-Stokes equations*. The first step for a detailed numerical treatment of this topic is explained here. We consider a symmetric and linear approach for instationary, incompressible flow problems, the so called Stokes equations

$$\left.\begin{aligned} \frac{\partial}{\partial t}\mathbf{v}(t,\mathbf{x}) - \frac{1}{\mathrm{Re}}\Delta\mathbf{v}(t,\mathbf{x}) + \nabla p(t,\mathbf{x}) &= \mathbf{0}, \\ \nabla\cdot\mathbf{v}(t,\mathbf{x}) &= \mathbf{0}, \end{aligned}\right\} \text{ on } (0,\infty)\times\Omega, \tag{1.1}$$

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 Reynolds number $\mathrm{Re} \in \mathbb{R}^+$. Additionally, we have $\Omega \subset \mathbb{R}^2$ as a bounded and connected domain with boundary $\Gamma = \partial\Omega$, some Dirichlet boundary conditions and appropriate initial conditions.

First we show that the discretized system within the feedback control approach leads to large-scale saddle point systems. In Section 3, we introduce the solution strategy for the saddle point system before we show some numerical examples in Section 4.

The methods below will be extended to the Navier-Stokes case [8, Section 7] in the future, and we comment on this in Subsection 3.2.

2. Discretized Stokes Control System. We apply a standard finite element discretization [2] to the *Stokes equations* (1.1) and get

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

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

*Research group Mathematics in Industry and Technology (MiIT), Chemnitz University of Technology, Reichenhainer Str. 41, 09126 Chemnitz, Germany (heiko.weichelt@mathematik.tu-chemnitz.de).

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 $M \in \mathbb{R}^{n_v \times n_v}$, the symmetric system matrix $A \in \mathbb{R}^{n_v \times n_v}$ and the discretized gradient $G \in \mathbb{R}^{n_v \times n_p}$ of rank n_p. The source term $\mathbf{f}(t)$ describes the feedback influence via the boundary and can be expressed as $\mathbf{f}(t) = B\mathbf{u}(t)$ [4, Section 2], with the boundary control $\mathbf{u}(t) \in \mathbb{R}^{n_r}$ and the input operator $B \in \mathbb{R}^{n_v \times n_r}$. Since in general, one can only observe the velocity 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 $C \in \mathbb{R}^{n_a \times n_v}$.

Equations (2.1a)-(2.1b) represent a differential-algebraic system (DAE) of differential index two, with $\begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix}$ as a singular left hand side coefficient matrix. Because the solution set does not lie in an affine subspace but on a (hidden) manifold of the Euclidean space, we face some additional difficulties referring to the solvability [16]. To avoid this problem we use the idea of index reduction described in [10, Section 3.] which is demonstrated in the next subsection for so called *descriptor systems* like (2.1).

2.1. Projection Method. We follow the idea of index reduction as in [10], which is used for balanced truncation model order reduction applied to descriptor systems (2.1). We will show in Subsection 2.3 how this idea is also applicable in our context. This means we use the projector

$$\mathit{\Pi} := I - G(G^T M^{-1} G)^{-1} G^T M^{-1}, \tag{2.2}$$

which is defined in [10, Section 3.] as an oblique projector in the Euclidean space. For $0 = G^T\mathbf{z}(t)$ this implies $\mathit{\Pi}^T\mathbf{z}(t) = \mathbf{z}(t)$. The projector (2.2) ensures that the solution fulfills the algebraic equation (2.1b) and simultaneously resides in the correct solution manifold, the so called *hidden manifold* defined by

$$\mathbf{0} = G^T M^{-1} A\mathbf{z}(t) + G^T M^{-1} G\mathbf{p}(t) + G^T M^{-1} B\mathbf{u}(t). \tag{2.3}$$

Thus, system (2.1) reduces to

$$\mathit{\Pi} M \mathit{\Pi}^T \frac{d}{dt}\mathbf{z}(t) = \mathit{\Pi} A \mathit{\Pi}^T \mathbf{z}(t) + \mathit{\Pi} B\mathbf{u}(t), \tag{2.4a}$$

$$\mathbf{y}(t) = C\mathit{\Pi}^T\mathbf{z}(t). \tag{2.4b}$$

Because the nullspace of $\mathit{\Pi}$ is not empty, we further consider the decomposition

$$\mathit{\Pi} = \Theta_l \Theta_r, \tag{2.5}$$

with $\Theta_l, \Theta_r \in \mathbb{R}^{n_v \times (n_v - n_p)}$ satisfying

$$\Theta_l^T \Theta_r = I_{(n_v - n_p)}. \tag{2.6}$$

If we substitute this decomposition into (2.4) we obtain

$$\Theta_r^T M \Theta_r \frac{d}{dt}\tilde{\mathbf{z}}(t) = \Theta_r^T A \Theta_r \tilde{\mathbf{z}}(t) + \Theta_r^T B\mathbf{u}(t), \tag{2.7a}$$

$$\mathbf{y}(t) = C\Theta_r^T \tilde{\mathbf{z}}(t), \tag{2.7b}$$

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 $\mathcal{C} = C\Theta_r^T$, we obtain

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

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

as a generalized state space system with a symmetric positive definite mass matrix $\mathcal{M}$. In [10, Section 4] the generalized Lyapunov equations

$$\mathcal{A}\tilde{P}\mathcal{M}^T + \mathcal{M}\tilde{P}\mathcal{A}^T = -\mathcal{B}\mathcal{B}^T, \tag{2.9}$$

$$\mathcal{A}^T\tilde{Q}\mathcal{M} + \mathcal{M}^T\tilde{Q}\mathcal{A} = -\mathcal{C}^T\mathcal{C}, \tag{2.10}$$

have to be solved for $\tilde{P}, \tilde{Q}$ to use the method for balanced truncation model order reduction. In the next subsections we will 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 (2.5). We will later see how we can work with (2.8) implicitly by solving certain saddle point problems.

2.2. The Feedback Control Approach. For an asymptotic stabilization of (1.1) we want to apply the idea of a *linear quadratic control approach (LQR)* to the system (2.8). An introduction to LQR for state space systems can be found in, e.g., [11]. Because we consider the generalized state space system (2.8) with $\mathcal{M} \neq I$ we have to modify the results slightly, as it is carried out in [14, Chapter 5.2]. There, one ends up with the optimal control defined by

$$\mathbf{u}_*(t) = -\mathcal{B}^T X\mathcal{M}\tilde{\mathbf{z}}_*(t) = -\mathcal{K}\tilde{\mathbf{z}}_*(t) \tag{2.11}$$

for system (2.8). Here $\mathbf{u}_*(t)$ minimizes the chosen cost functional

$$\mathcal{J}(\tilde{\mathbf{z}}(t), \mathbf{u}(t)) = \frac{1}{2}\int_0^\infty \tilde{\mathbf{z}}(t)^T\mathcal{C}^T\mathcal{C}\tilde{\mathbf{z}}(t) + \mathbf{u}(t)^T\mathbf{u}(t)\,\mathrm{dt}, \tag{2.12}$$

and X is the solution of the *Generalized Algebraic Riccati Equation (GARE)*

$$\mathbf{0} = \mathcal{C}^T\mathcal{C} + \mathcal{A}^T X\mathcal{M} + \mathcal{M}^T X\mathcal{A} - \mathcal{M}^T X\mathcal{B}\mathcal{B}^T X\mathcal{M} =: \mathfrak{R}(X). \tag{2.13}$$

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 Generalized Algebraic Riccati Equations. A common way to solve such nonlinear matrix equations is a Newton type method, as described in [14, Chapter 4.5], and for the generalized case in [14, Chapter 5.2]. The Newton system at step m is given by

$$\mathfrak{R}'|_{X^{(m)}}(N^{(m)}) = -\mathfrak{R}(X^{(m)}), \qquad X^{(m+1)} = X^{(m)} + N^{(m)}, \tag{2.14}$$

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

$$\mathfrak{R}'|_{X^{(m)}}: \quad N^{(m)} \mapsto (\mathcal{A} - \mathcal{B}\mathcal{B}^T X^{(m)}\mathcal{M})^T N^{(m)}\mathcal{M} + \mathcal{M}^T N^{(m)}(\mathcal{A} - \mathcal{B}\mathcal{B}^T X^{(m)}\mathcal{M}). \tag{2.15}$$

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

```
Input: $\mathcal{A}^{(m)}, \mathcal{M}, \mathcal{W}^{(m)}$ and shift parameters $\{p_1, \dots, p_{i_{max}}\}$
Output: $Z = Z_{i_{max}} \in \mathbb{C}^{n \times t_{i_{max}}}$, such that $ZZ^H \approx X^{(m+1)}$
1: $V_1 = \sqrt{-2\operatorname{Re}(p_1)}\left(\mathcal{A}^{(m)} + p_1\mathcal{M}\right)^{-T}(\mathcal{W}^{(m)})^T$
2: $Z_1 = V_1$
3: for $i = 2, 3, \dots, i_{max}$ do
4:     $V_i = \sqrt{\operatorname{Re}(p_i)/\operatorname{Re}(p_{i-1})}\left(V_{i-1} - (p_i + \overline{p_{i-1}})\left(\mathcal{A}^{(m)} + p_i\mathcal{M}\right)^{-T}(\mathcal{M}^T V_{i-1})\right)$
5:     $Z_i = [Z_{i-1}\, V_i]$
6: end for
```

Because $N^{(m)} = X^{(m+1)} - X^{(m)}$ the equations (2.14)-(2.15) yield the generalized Lyapunov equation

$$(\mathcal{A}^{(m)})^T X^{(m+1)}\mathcal{M} + \mathcal{M}^T X^{(m+1)}\mathcal{A}^{(m)} = -(\mathcal{W}^{(m)})^T\mathcal{W}^{(m)}, \tag{2.16}$$

with $\mathcal{A}^{(m)} = \mathcal{A} - \mathcal{B}\mathcal{B}^T X^{(m)}\mathcal{M}$ and $\mathcal{W}^{(m)} = \begin{bmatrix} C \\ \mathcal{B}^T X^{(m)}\mathcal{M}\end{bmatrix}$, which has the same structure as in (2.10) and has to be solved in each Newton step. Hence, we see that the index reduction method in [10] is applicable.

A solution strategy for this kind of equation is the *low-rank ADI iteration* [1, Chapter 12], which is extended for the generalized case in [5] as shown in Algorithm 1 using the above notation. For more details we refer to [14].

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

$$\left(\mathcal{A}^{(m)} + p_i\mathcal{M}\right)^T \Lambda = \mathcal{Y}, \tag{2.17}$$

with different right hand sides $\mathcal{Y}$. Because one neither wants to build the dense projector $\mathit{\Pi}$ nor its decomposition, we recall the results in [10, Section 5] which state that the solution of the Θ-projected equation (2.17) is also a solution of the Π-projected equation

$$\mathit{\Pi}\left(A^T - M^T X^{(m)} BB^T + p_i M^T\right)\mathit{\Pi}^T\Lambda = \mathit{\Pi} Y, \tag{2.18}$$

and use [10, Lemma 5.2] to solve the equivalent linear system

$$\begin{bmatrix} A^T - M^T X^{(m)} BB^T + p_i M^T & G \\ G^T & 0 \end{bmatrix}\begin{bmatrix}\Lambda \\ * \end{bmatrix} = \begin{bmatrix} Y \\ 0 \end{bmatrix}, \tag{2.19}$$

instead of system (2.18). The complete process for computing the feedback operator $K = B^T X M$ is shown in Algorithm 2. The linear system (2.19) has to be solved in each ADI step. Note that each Newton step consists of several ADI steps. In the remainder of this paper we show how we can efficiently solve (2.19).

3. Solving Large-Scale Saddle Point Systems.

Systems of the form (2.19) are often referred to as *saddle point systems*. A comprehensive overview about saddle point systems is given in [6]. We explain the properties for the system (2.19) and our choice of iterative solver.

Algorithm 2 Generalized Low-rank Cholesky factor Newton method for Stokes

```
Input: $M, A, G, B, C$, shift parameters $\{p_1, \dots, p_{n_{ADI}}\}$
Output: Feedback operator $K$
 1: $K^0 = [\,]$
 2: for $m = 1, 2, \dots, n_{ARE}$ do
 3:   $W^{(m)} = \begin{bmatrix} C^T & (K^{(m-1)})^T \end{bmatrix}$
 4:   Get $V_1$ by solving
        $$\begin{bmatrix} A^T - (K^{(m-1)})^T B^T + p_1 M^T & G \\ G^T & 0 \end{bmatrix} \begin{bmatrix} V_1 \\ * \end{bmatrix} = \begin{bmatrix} \sqrt{-2\operatorname{Re}(p_1)}\, W^{(m)} \\ 0 \end{bmatrix}$$
 5:   $K_1^{(m)} = B^T V_1 V_1^T M$
 6:   for $i = 2, 3, \dots, n_{ADI}$ do
 7:     Get $\tilde{V}$ by solving
          $$\begin{bmatrix} A^T - (K^{(m-1)})^T B^T + p_i M^T & G \\ G^T & 0 \end{bmatrix} \begin{bmatrix} \tilde{V} \\ * \end{bmatrix} = \begin{bmatrix} M^T V_{i-1} \\ 0 \end{bmatrix}$$
 8:     $V_i = \sqrt{\operatorname{Re}(p_i)/\operatorname{Re}(p_{i-1})} \left( V_{i-1} - (p_i + \overline{p_{i-1}})\tilde{V} \right)$
 9:     $K_i^{(m)} = K_{i-1}^{(m)} + B^T V_i V_i^T M$
10:     if $\left( \frac{||K_i^{(m)} - K_{i-1}^{(m)}||_F}{||K_i^{(m)}||_F} < tol_{adi} \right)$ then
11:       break
12:     end if
13:   end for
14:   $K^{(m)} = K_{n_{ADI}}^{(m)}$
15:   if $\left( \frac{||K^{(m)} - K^{(m-1)}||_F}{||K^{(m)}||_F} < tol_{newton} \right)$ then
16:     break
17:   end if
18: end for
19: $K = K^{n_{ARE}}$
```

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 (2.19), with $M = M^T \succ 0, A = A^T \prec 0$ and $p_i < 0$. Although the matrices A, M and G are sparse, the low rank product $K^T B^T$ will become dense, such that the main part of the matrix will become dense, destroying the efficiency of Algorithm 2. To avoid this, we can write the system in form of a low rank update

$$\Bigg(\underbrace{\begin{bmatrix} A^T + p_i M^T & G \\ G^T & 0 \end{bmatrix}}_{\mathbf{A}} - \underbrace{\begin{bmatrix} K^T \\ 0 \end{bmatrix}}_{\mathbf{K}^T} \underbrace{\begin{bmatrix} B^T & 0 \end{bmatrix}}_{\mathbf{B}^T} \Bigg) \underbrace{\begin{bmatrix} \Lambda \\ * \end{bmatrix}}_{\mathbf{\Lambda}} = \underbrace{\begin{bmatrix} Y \\ 0 \end{bmatrix}}_{\mathbf{Y}}, \tag{3.1}$$

which can be written in compact form as

$$(\mathbf{A} - \mathbf{K}^T \mathbf{B}^T)\mathbf{\Lambda} = \mathbf{Y}, \tag{3.2}$$

and then use the *Sherman-Morrison-Woodburry* formula [9]

$$(\mathbf{A} - \mathbf{K}^T\mathbf{B}^T)^{-1} = \left(I_{n_v} + \mathbf{A}^{-1}\mathbf{K}^T(I_{n_r} - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{K}^T)^{-1}\mathbf{B}^T\right)\mathbf{A}^{-1}. \tag{3.3}$$

This means we have to solve with $\mathbf{A}$ and a small dense matrix of size $n_r \ll n_v$. Furthermore, it appears that we additionally have to solve with $\mathbf{A}$ for the right hand side $\mathbf{K}^T$. But if we look at line 4 of Algorithm 2, we see that we solve for $\mathbf{K}^T$ as the right hand side in the first ADI step, such that we can avoid the additional costs, if we save the result and re-use it later. Finally, we obtain the saddle point system

$$\begin{bmatrix} A^T + p_i M^T & G \\ G^T & 0 \end{bmatrix} \begin{bmatrix} \Lambda \\ * \end{bmatrix} = \begin{bmatrix} Y \\ 0 \end{bmatrix}, \tag{3.4}$$

which we have to solve for a couple of different right hand sides. Our strategy for computing the solution of (3.4) is discussed next.

3.2. Preconditioned Iterative Solvers. To solve the system (3.4) we use iterative methods, instead of direct solvers, because the size $n = n_v + n_p$ becomes large for usual finite element discretizations. If we want to use a symmetric iterative solver (e.g. MINRES [12]) we might use a symmetric positive definite preconditioner such as

$$\mathbf{P} = \begin{bmatrix} -P_F & 0 \\ 0 & P_{SC} \end{bmatrix} \tag{3.5}$$

[8, Section 6.1], where P_F approximates $F = A^T + p_i M^T$ and P_{SC} the *Schur complement* $G^T F^{-1} G$.

Because our primary focus is to derive efficient preconditioners for the Navier-Stokes saddle point system we do not implement and use this version of the preconditioner. Therefore we have explicitly used A^T during the whole derivation, although A is symmetric for the Stokes equations, and investigate a preconditioner for the potential non-symmetric block structured matrix

$$\mathbf{F} := \begin{bmatrix} F & G \\ G^T & 0 \end{bmatrix}, \text{ with } F = A^T + p_i M^T \text{ and } \mathbf{F} \neq \mathbf{F}^T. \tag{3.6}$$

We believe that the additional cost of using a non-symmetric iterative method also for the Stokes equations are minimal because we need only a few GMRES steps using efficient preconditioning techniques.

Based on the ideas in [8, Section 8.1] our choice is the block structured non-symmetric left preconditioner

$$\mathbf{P} := \begin{bmatrix} P_F & 0 \\ G^T & -P_{SC} \end{bmatrix} \Rightarrow \mathbf{P}^{-1} = \begin{bmatrix} P_F^{-1} & 0 \\ P_{SC}^{-1} G^T P_F^{-1} & -P_{SC}^{-1} \end{bmatrix}. \tag{3.7}$$

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

$$\mathbf{P}^{-1}\mathbf{F} = \begin{bmatrix} P_F^{-1}F & 0 \\ P_{SC}^{-1}G^T P_F^{-1} F - P_{SC}^{-1}G^T & P_{SC}^{-1}G^T P_F^{-1} G \end{bmatrix}. \tag{3.8}$$

For $P_F = F$ and $P_{SC} = G^T F^{-1} G$ (3.8) yields

$$\begin{bmatrix} F^{-1}F & 0 \\ P_{SC}^{-1}G^T F^{-1} F - P_{SC}^{-1}G^T & P_{SC}^{-1}G^T F^{-1} G \end{bmatrix} = \begin{bmatrix} I_{n_v} & 0 \\ 0 & I_{n_r} \end{bmatrix}. \tag{3.9}$$

But we cannot form $P_{SC} \in \mathbb{R}^{n_p \times n_p}$ as this would require $F^{-1} \in \mathbb{R}^{n_v \times n_v}$ explicitly. A good approximation for the Schur complement for the Navier-Stokes case is derived from a least-squares commutator approach, i.e.,

$$P_{SC} \approx S_p F_p^{-1} M_p \quad \Rightarrow \quad P_{SC}^{-1} \approx M_p^{-1} F_p S_p^{-1}, \tag{3.10}$$

[8, Section 8.2], with S_p the discretized Laplacian on the pressure space and F_p the system matrix defined on the pressure space. Thus, one has to solve a pure Neumann problem S_p^{-1} [7], multiply with the system matrix on the pressure space F_p once and solve a linear system with symmetric positive definite M_p, to apply the Schur complement approximation. Note that for the symmetric Stokes case $F_p = A_P + p_c M_p$ the Schur complement approximation becomes slightly easier [15, Section 3], because the system matrix A_p is just the negative stiffness matrix S_p scaled by $\nu := \frac{1}{\mathrm{Re}}$, such that

$$P_{SC} \approx S_p(-\nu S_p + p_i M_p)^{-1} M_p, \tag{3.11}$$

$$\Rightarrow \quad P_{SC}^{-1} \approx M_p^{-1}(-\nu S_p + p_i M_p) S_p^{-1} = -\nu M_p^{-1} + p_i S_p^{-1}. \tag{3.12}$$

Hence, we just solve a pure Neumann problem S_p^{-1} [7] and a linear system with the mass matrix M_p for the differently scaled right hand side.

In (3.9) we assumed F as the best choice for P_F. To get an efficient preconditioner we want to apply a MULTIGRID approximation of F in the future [8, Section 8.3.2]. For the proof of concept we use a sparse direct solver in MATLAB® (via the "backslash" operator) in the numerical experiments for our proposed preconditioner.

Some problems with the nested iterations, as well as some mentionable remarks regarding the numerical realization, are shown next.

4. Numerical Examples.

The matrices for the numerical tests arise from a standard mixed finite element discretization (P_2-P_1 *Taylor-Hood* elements) of a *von Kármán vortex street*. Using a *Bänsch-refinement* [3] we get six different magnitudes for n_v and n_p and every second level corresponds to one level of global uniform refinement (see Table 4.1).

Level	n_v	n_p
1	3 452	453
2	8 726	1 123
3	20 512	2 615
4	45 718	5 783
5	99 652	12 566
6	211 452	26 572

Table 4.1: Levels of refinement

All computations are done within MATLAB R2010b on a 64-bit compute server with CPU type Intel® Xeon® @3.46GHz, with 2 CPUs, 12 Cores (6 Cores per CPU) and 144 GB main memory available, of which we only use a minor fraction during all our experiments.

At first we show some results concerning the innermost iteration for solving (3.4).

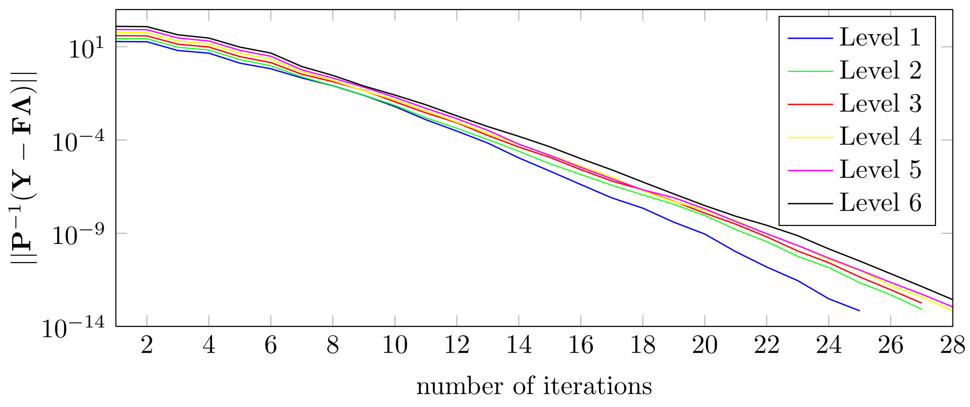
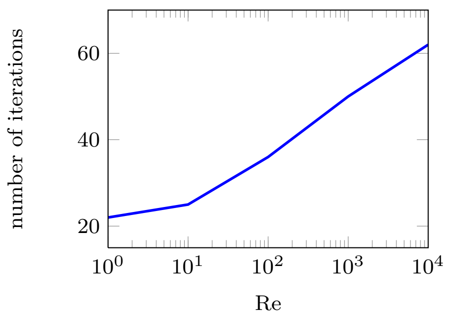


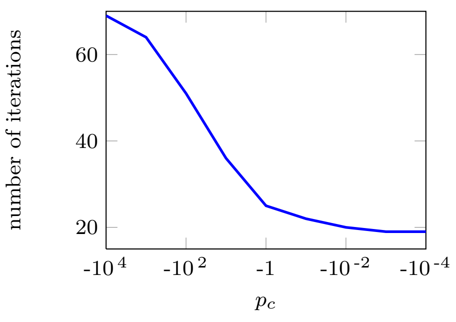
Figure 4.1: Preconditioned residuals of GMRES for refinement levels of Table 4.1 ($p_c = -1, \mathrm{Re} = 10$).

4.1. Solving the Saddle Point System. To solve system (3.4) for one exemplary right hand side we use the non-symmetric iterative solver GMRES that is available in MATLAB with the preconditioner (3.7) and the Schur complement approximation (3.12) introduced above. In Figure 4.1 the preconditioned residuals corresponding to the numbers of iterations for the different levels of refinement from Table 4.1 are listed.

One observes that we do not need more iterations if we refine our mesh. Hence, we have a robust (with respect to the mesh parameter) preconditioned iterative method to solve saddle point systems like (3.4). Another parameter is the Reynolds number Re which can influence the number of iterations. Figure 4.2a shows this influence for refinement Level 1 and the ADI shift $p_c = -1$. We see that we need only a few more iterations if we increase Re. The next parameter which changes during Algorithm 2 in each ADI step is the ADI shift p_c. Figure 4.2b shows the dependence of the number of iterations on p_c for refinement Level 1 and $\mathrm{Re} = 10$. As before, we need slightly more iterations if we increase the absolute value of p_c.



(a) Number of iterations for different Reynolds numbers ($p_c = -1$, Level 1).



(b) Number of iterations for different ADI-shifts (Re = 10, Level 1).

Figure 4.2: Number of iterations depending on parameters.

GMRES tol	∅ GMRES steps	# ADI steps	time
10^{-6}	13	> 500	> 2800 sec.
10^{-7}	15	82	375 sec.
10^{-8}	16	71	348 sec.
10^{-9}	18	65	330 sec.
$\mathbf{10^{-10}}$	**19**	**55**	**308 sec.**
10^{-11}	20	55	335 sec.
10^{-12}	21	55	341 sec.
10^{-13}	23	55	386 sec.
10^{-14}	23	55	458 sec.
10^{-15}	24	55	494 sec.
10^{-16}	25	55	491 sec.
”direct solver”	- -	55	7 sec.

Table 4.2: Relation of the accuracy of GMRES with the time to solve a Newton step (Re = 10, Level 1).

4.2. Solving Nested Iterations. The saddle point system (3.4) has to be solved at each ADI step. To compute one Newton step (2.14) we need a number of ADI steps to reach the stopping criterion for the Newton iteration. Depending on how accurate the saddle point system in each ADI step is solved we need more or less ADI steps. Because we use iterative solvers we can influence this accuracy up to a certain point. In Table 4.2 we compare the number of ADI steps and the required computation time in relation to the accuracy of the GMRES iteration. Notice that we do not necessarily need the highest accuracy GMRES can achieve. For this example we could reach the best time for an intended tolerance of 10^{-10} where we needed on average 19 steps within GMRES and 55 ADI steps. We see that this setup generates the same number of ADI steps as if we would use a direct solver for the saddle point system. For this level of refinement the direct solver is of course faster than our iterative method because the number of unknowns is comparably small.

Finally we note that the choice of the ADI shifts p_c and the stopping criteria for the ADI and the Newton iteration will be presented in detail elsewhere, because the focus of this paper is on the efficient solution of the occurring saddle point systems.

5. Conclusions. In the paper we have shown how we can use the idea of index reduction for balanced truncation model order reduction for the Riccati-based feedback approach, applied to a Stokes flow problem. We have pointed out the properties of the resulting saddle point systems and have introduced an efficient way to solve such equations. Therefore we have investigated preconditioners to be used in iterative methods, and in particular a way to get a good approximation of the Schur complement. We have illustrated some numerical examples how different parameters influence the solvers and have shown the competitiveness of our algorithm.

In the future, we will deal with the non-symmetric Navier-Stokes flow and explore in detail the additional difficulties arising there. We will also investigate the acceleration of the iterative solver in order to deal with multiple right hand sides, which is an advantage of the direct solvers, that solve them almost simultaneously.

Acknowledgments. This paper is a joint work with Peter Benner, Jens Saak and Martin Stoll and is developed within the project *Optimal Control-Based Feedback Stabilization in Multi-Field Flow Problems* which is a part of the DFG priority program 1253: *Optimization with Partial Differential Equations.*

REFERENCES

[1] A. C. Antoulas, *Approximation of Large-Scale Dynamical Systems*, SIAM, 2005.

[2] D. Arnold, F. Brezzi, and M. Fortin, *A stable finite element for the Stokes equations*, Calcolo, 21 (1984), pp. 337–344. 10.1007/BF02576171.

[3] E. Bänsch, *An adaptive finite-element strategy for the three-dimensional time-dependent Navier-Stokes equations*, Journal of Computational and Applied Mathematics, 36 (1991), pp. 3–28.

[4] E. Bänsch and P. Benner, *Stabilization of Incompressible Flow Problems by Riccati-Based Feedback*, In G. Leugering, S. Engell, A. Griewank, M. Hinze, R. Rannacher, V. Schulz, M. Ulbrich, S. Ulbrich (editors), Constrained Optimization and Optimal Control for Partial Differential Equations, Birkhäuser, Basel, 2012.

[5] P. Benner, *Solving large-scale control problems*, IEEE Control Systems Magazine, 14 (2004), pp. 44–59.

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

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

[8] H. Elman, D. Silvester, and A. Wathen, *Finite Elements and Fast Iterative Solvers: with applications in incompressible fluid dynamics*, Oxford University Press, Oxford, 2005.

[9] G. Golub and C. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, third ed., 1996.

[10] 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 Journal on Scientific Computing, 30 (2008), pp. 1038–1063.

[11] A. Locatelli, *Optimal Control: An Introduction*, Birkhäuser Verlag, Basel, Boston, Berlin, 2001.

[12] C. C. Paige and M. A. Saunders, *Solutions of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal, 12 (1975), pp. 617–629.

[13] J. Raymond, *Feedback boundary stabilization of the two-dimensional Navier-Stokes equations*, SIAM Journal on Control and Optimization, 45 (2006), pp. 790–828.

[14] J. Saak, *Efficient Numerical Solution of Large Scale Algebraic Matrix Equations in PDE Control and Model Order Reduction*, PhD thesis, Chemnitz University of Technology, July 2009.

[15] M. Stoll and A. Wathen, *All-at-once solution of time-dependent Stokes control*, Max Planck Institute Magdeburg Preprints 11-03, (2011). `http://www.mpi-magdeburg.mpg.de/preprints/index.php`.

[16] 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, August 1996. `http://www.tu-chemnitz.de/sfb393/Files/PS/sfb96-08.ps.gz`.