© 2016. This manuscript version is made available under the CC-BY-NC-ND 4.0 license
http://creativecommons.org/licenses/by-nc-nd/4.0/

Fast iterative solvers for large matrix systems arising from time-dependent Stokes control problems

John W. Pearson

School of Mathematics, Statistics and Actuarial Science, University of Kent, Cornwallis Building (East), Canterbury, Kent, CT2 7NF, UK

Abstract

In this manuscript we consider the development of fast iterative solvers for Stokes control problems, an important class of PDE-constrained optimization problems. In particular we wish to develop effective preconditioners for the matrix systems arising from finite element discretizations of time-dependent variants of such problems. To do this we consider a suitable rearrangement of the matrix systems, and exploit the saddle point structure of many of the relevant sub-matrices involved – we may then use this to construct representations of these sub-matrices based on good approximations of their $(1,1)$-block and Schur complement. We test our recommended iterative methods on a distributed control problem with Dirichlet boundary conditions, and on a time-periodic problem.

Keywords: PDE-constrained optimization, time-dependent Stokes control, preconditioning, saddle point system, Schur complement, commutator argument.

1. Introduction

Of late, the efficient numerical solution of PDE-constrained optimization problems has been an active area of research (see [30] for an introduction to PDE-constrained optimization). One of the main classes of such problems is that of Stokes control problems (which arise from flow control), and this is the class that we seek to examine in this paper. Many researchers have sought to develop solution strategies for the matrix systems resulting from finite element discretizations of time-independent [20, 29, 32] and time-dependent [7, 10, 11, 28] versions of these setups. It is a substantial challenge in particular to build solvers for the complex matrix systems that arise in the time-dependent case, and this is the problem on which we focus in this paper.

When discretized using finite elements, the resulting matrix systems are of very high dimension, even in comparison to equivalent time-independent formulations, and are also sparse. It is therefore extremely desirable to develop fast and robust iterative methods for their solution. We do this by exploiting the saddle point structure of the relevant matrices to construct effective preconditioners for the entire system. Previous research into this problem has also exploited the structure of the matrix systems: in [7] multigrid routines were developed using appropriate smoothers and prolongation/restriction operators, and in [28] saddle point preconditioners which exhibited mesh-independence were constructed and applied within MINRES. In this paper a preconditioned MINRES method is also proposed, but with the goal that the solver exhibits favourable convergence properties as both mesh-size and regularization parameter are modified.

In this paper we consider such problems with different types of conditions at initial (and final) time: both initial conditions and time-periodic conditions. We wish our solvers for these problems to be fast and effective for a range of parameter values, to involve the storage of small matrices compared with the dimension of the entire system, and to be parallelizable. To do this we build on work undertaken by the author in [20, 21] when solving matrix systems from time-independent Stokes and Navier-Stokes control problems, to tackle the larger and more complex systems arising when a time-dependent component is

© 2016. This manuscript version is made available under the CC-BY-NC-ND 4.0 license
http://creativecommons.org/licenses/by-nc-nd/4.0/

introduced. We find that this methodology can reasonably be applied to the time-dependent set-up, and we wish to present numerical results that validate this assertion.

This paper is structured as follows. In this section we state the problems that we wish to examine, and introduce some basic theory of saddle point systems. In Section 2 we derive the matrix systems of which the solution is required; in Section 3 we devise preconditioning strategies for these systems, which may be applied within a suitable iterative method. In Section 4 we present numerical results to demonstrate the performance of our methods, and finally in Section 5 we make some concluding comments.

1.1. Problem statement

The two problems on which we wish to focus in this paper are both time-dependent variants of the widely-considered Stokes control problem. The first is the following distributed control problem with Dirichlet boundary conditions:

$$\begin{aligned} \textbf{(P1)} \qquad \min_{\vec{v},\vec{u}} \quad & \frac{1}{2}\int_0^T \int_\Omega \|\vec{v}-\vec{v}_d\|^2 \ \mathrm{d}\Omega\mathrm{d}t + \frac{\beta}{2}\int_0^T \int_\Omega \|\vec{u}\|^2 \ \mathrm{d}\Omega\mathrm{d}t \\ \text{s.t.} \quad & \frac{\partial \vec{v}}{\partial t} - \nabla^2\vec{v} + \nabla p = \vec{u}, && \text{in } \Omega\times[0,T], \\ & -\nabla\cdot\vec{v} = 0, && \text{in } \Omega\times[0,T], \\ & \vec{v}(\mathbf{x},t) = \vec{f}(\mathbf{x},t), && \text{on } \partial\Omega\times[0,T], \\ & \vec{v}(\mathbf{x},0) = \vec{g}(\mathbf{x}), && \text{on } \Omega. \end{aligned}$$

This problem is solved for spatial coordinates given by $\mathbf{x}\in\mathbb{R}^d$, $d\in\{2,3\}$, and time t, on the space-time domain $\Omega\times[0,T]$ with boundary $\partial\Omega\times[0,T]$. The *state variables* for this problem are given by the velocity $\vec{v}$ (defined in d dimensions) and the pressure p. The *control variable* is denoted as $\vec{u}$ (in d dimensions), and $\vec{v}_d$ defines the *desired state*. The positive parameter β denotes the *regularization parameter* (or *Tikhonov parameter*), and indicates at what ratio one prioritizes the realization of a state variable close to the desired state as opposed to the minimization of the control. The functions $\vec{f}$ and $\vec{g}$ are defined over the spatial coordinates (and in the case of $\vec{f}$ over time as well), and correspond to boundary conditions and initial conditions respectively. We note that it would be equally feasible to include a natural boundary condition of the form $\frac{\partial\vec{v}(\mathbf{x},t)}{\partial\vec{n}} - p\vec{n} = \vec{f}(\mathbf{x},t)$ (where $\vec{n}$ is the outward facing normal vector to Ω, and $\frac{\partial}{\partial\vec{n}}$ denotes the normal derivative), or indeed a Neumann/mixed boundary condition, instead of a Dirichlet condition. The methodology introduced in this paper could also be tailored to these problems; however the performance of our preconditioner may change or degrade slightly, as has been observed for the forward problem.

The second problem is of a similar flavour to that stated above, but is a time-periodic problem. We write this as

$$\begin{aligned} \textbf{(P2)} \qquad \min_{\vec{v},\vec{u}} \quad & \frac{1}{2}\int_0^T \int_\Omega \|\vec{v}-\vec{v}_d\|^2 \ \mathrm{d}\Omega\mathrm{d}t + \frac{\beta}{2}\int_0^T \int_\Omega \|\vec{u}\|^2 \ \mathrm{d}\Omega\mathrm{d}t \\ \text{s.t.} \quad & \frac{\partial \vec{v}}{\partial t} - \nabla^2\vec{v} + \nabla p = \vec{u}, && \text{in } \Omega\times[0,T], \\ & -\nabla\cdot\vec{v} = 0, && \text{in } \Omega\times[0,T], \\ & \vec{v}(\mathbf{x},t) = \vec{f}(\mathbf{x},t), && \text{on } \partial\Omega\times[0,T], \\ & \vec{v}(\mathbf{x},0) = \vec{v}(\mathbf{x},T), && \text{on } \Omega. \end{aligned}$$

Here there is no longer an initial condition corresponding to a given function, but instead a restriction that the velocity profile must be the same at initial and final times. This creates different features when attempting to solve the problem at hand.

The problems, as well as the matrix systems arising from them, are of complex structure, and so finding ways to solve these problems efficiently is a highly non-trivial task. In the next section, we discuss the form of matrices which we may often consider when carrying out the solution process.

© 2016. This manuscript version is made available under the CC-BY-NC-ND 4.0 license
http://creativecommons.org/licenses/by-nc-nd/4.0/

1.2. Saddle point systems

When discretizing the problems **(P1)** and **(P2)** using a finite element method, the resulting matrix systems are of *saddle point* form. We therefore wish to briefly introduce such systems and some widely used approximations of them.

The general saddle point systems that we consider are of the form

$$\underbrace{\begin{bmatrix} \Phi & \Psi^T \\ \Psi & -\Theta \end{bmatrix}}_{\mathcal{A}} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}, \tag{1}$$

where $\Phi \in \mathbb{R}^{m\times m}$, $\Psi \in \mathbb{R}^{n\times m}$ (with $n \leq m$) has full row rank, and $\Theta \in \mathbb{R}^{n\times n}$. In Section 3.1, we describe how matrices with saddle point structure arise for the problems under consideration in this paper. For the matrix systems that we examine, Φ and Θ are symmetric (with Φ also invertible), and all of the matrices are large and sparse.

It is well-known (see [8, 12, 13]) that some preconditioners for the above matrix system, which are frequently very effective, are given by

$$\mathcal{P} = \underbrace{\begin{bmatrix} \Phi & 0 \\ 0 & \Theta + \Psi\Phi^{-1}\Psi^T \end{bmatrix}}_{\mathcal{P}_1} \quad \text{or} \quad \underbrace{\begin{bmatrix} \Phi & 0 \\ \Psi & \pm\left(\Theta + \Psi\Phi^{-1}\Psi^T\right) \end{bmatrix}}_{\mathcal{P}_2^+ \text{ or } \mathcal{P}_2^-}. \tag{2}$$

Specifically it is known that as long as the preconditioned matrix system is invertible,

$$\lambda\left(\left(\mathcal{P}_2^+\right)^{-1}\mathcal{A}\right) \in \{\pm 1\}, \quad \lambda\left(\left(\mathcal{P}_2^-\right)^{-1}\mathcal{A}\right) \in \{1\},$$

for all suitable Φ, Ψ and Θ, and

$$\lambda\left(\mathcal{P}_1^{-1}\mathcal{A}\right) \in \left\{1, \frac{1}{2}(1 \pm \sqrt{5})\right\},$$

if $\Theta = 0$. Furthermore, if Φ and Θ are positive definite (as we will often find within this paper), the eigenvalues of $\mathcal{P}_1^{-1}\mathcal{A}$ can be shown to be well clustered (see [1, 19, 27]).

We note that the two common components of the preconditioners $\mathcal{P}_1$, $\mathcal{P}_2^{\pm}$ are the inverses of the $(1,1)$-block Φ and the (negative) Schur complement $S := \Theta + \Psi\Phi^{-1}\Psi^T$. Furthermore, we note that for the preconditioners in (2) to be applicable in practice (in part because S is typically dense even if $\mathcal{A}$ is sparse), we need to consider approximations of Φ and S to create preconditioners of the form

$$\widehat{\mathcal{P}} = \begin{bmatrix} \widehat{\Phi} & 0 \\ 0 & \widehat{S} \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} \widehat{\Phi} & 0 \\ \Psi & \pm\widehat{S} \end{bmatrix}. \tag{3}$$

We will therefore return later to the theme of approximating the inverses of the $(1,1)$-block and Schur complement of matrices arising from time-dependent Stokes control problems. As we are preconditioning sparse matrix systems the goal is to develop accurate representations of Φ^{-1} and S^{-1}, which are themselves cheap to apply.

2. Matrix systems

We now consider the matrix systems obtained when discretizing Problems **(P1)** and **(P2)** using a finite element method. For this work we use the well-known *Taylor-Hood element*: that is we discretize the velocity $\vec{v}$ (and control $\vec{u}$) using piecewise quadratic basis functions, and discretize the pressure p using piecewise linear basis functions. This is far from the only option for the finite element discretization, but we point out that the methodology introduced in this paper may also be applied to alternative discretization strategies.

© 2016. This manuscript version is made available under the CC-BY-NC-ND 4.0 license
http://creativecommons.org/licenses/by-nc-nd/4.0/

We examine a *discretize-then-optimize* approach, where we build a discrete Lagrangian and then differentiate with respect to discrete variables to obtain optimality conditions.[1] Let us first consider discretizing the forward problem (that is the time-dependent Stokes equations) from **(P1)** or **(P2)** using a backward Euler method in time. This means that at each time-step $j = 1, ..., N_t$, we obtain equations of the form

$$\begin{aligned} \frac{\vec{v}_j - \vec{v}_{j-1}}{\tau} - \nabla^2 \vec{v}_j + \nabla p_j &= \vec{u}_j, \\ -\nabla \cdot \vec{v}_j &= 0, \end{aligned} \tag{4}$$

where $\vec{v}_j$, p_j and $\vec{u}_j$ denote the approximations of the velocity, pressure and control at the j-th time-step, and τ denotes the (constant) time-step used. We also introduce the notation h to represent the mesh-size used.

Discretizing the equations (4) using a finite element method, and compiling a matrix system at each time-step, gives for Problem **(P1)**:

$$\underbrace{\begin{bmatrix} \tau^{-1}\mathbf{M}+\mathbf{K} & B^T & & & & & & \\ B & 0 & & & & & & \\ -\tau^{-1}\mathbf{M} & 0 & \tau^{-1}\mathbf{M}+\mathbf{K} & B^T & & & & \\ 0 & 0 & B & 0 & \ddots & & & \\ & & & \ddots & \ddots & \ddots & & \\ & & & & \ddots & \ddots & \ddots & \\ & & & & -\tau^{-1}\mathbf{M} & 0 & \tau^{-1}\mathbf{M}+\mathbf{K} & B^T \\ & & & & 0 & 0 & B & 0 \end{bmatrix}}_{\mathcal{K}_1} \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{p}_1 \\ \mathbf{v}_2 \\ \mathbf{p}_2 \\ \vdots \\ \vdots \\ \mathbf{v}_{N_t} \\ \mathbf{p}_{N_t} \end{bmatrix}$$

$$- \underbrace{\begin{bmatrix} \mathbf{M} & & & \\ 0 & & & \\ & \mathbf{M} & & \\ & 0 & & \\ & & \ddots & \\ & & & \mathbf{M} \\ & & & 0 \end{bmatrix}}_{\mathcal{N}} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_{N_t} \end{bmatrix} = \underbrace{\begin{bmatrix} \tau^{-1}\widetilde{\mathbf{v}}_0 + \mathbf{c} \\ \mathbf{d} \\ \mathbf{c} \\ \mathbf{d} \\ \vdots \\ \vdots \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix}}_{\mathbf{f}}. \tag{5}$$

Here, the vectors $\mathbf{v}_j$, $\mathbf{p}_j$ and $\mathbf{u}_j$ relate to the values of $\vec{v}$, p and $\vec{u}$ at the j-th time-step (we take N_t time-steps, and so $j = 1, ..., N_t$). The matrices $\mathbf{M}$ and $\mathbf{K}$ are $d \times d$ block diagonal matrices containing finite element mass and stiffness matrices on the block diagonals (these are matrices containing entries of the form $\int_\Omega \phi_i \phi_j \, \mathrm{d}\Omega$ and $\int_\Omega \nabla\phi_i \cdot \nabla\phi_j \, \mathrm{d}\Omega$ respectively). The matrix B is given by $[B_{x_1} \; B_{x_2}]$ when $d = 2$ (and $\mathbf{x} = [x_1, x_2]^T$), and $[B_{x_1} \; B_{x_2} \; B_{x_3}]$ when $d = 3$ (and $\mathbf{x} = [x_1, x_2, x_3]^T$), with B_{x_k} containing entries of the form $\int_\Omega \psi_i \frac{\partial \phi_j}{\partial x_k} \, \mathrm{d}\Omega$. Further, $\widetilde{\mathbf{v}}_0$ contains terms of the form $\int_\Omega \vec{v}_0 \phi_i \, \mathrm{d}\Omega$. In all of these definitions, $\{\phi_i\}$ denote the (piecewise quadratic) **Q2** finite element basis functions that we use to discretize $\mathbf{v}$ and $\mathbf{u}$, with $\{\psi_i\}$ representing the (piecewise linear) **Q1** finite element basis functions used to discretize p. The vectors $\mathbf{c}$ and $\mathbf{d}$ arise from the boundary conditions imposed.

[1] The alternative *optimize-then-discretize* method, which will generate a matrix system of similar structure, involves finding optimality conditions from a continuous Lagrangian and then discretizing these conditions.

© 2016. This manuscript version is made available under the CC-BY-NC-ND 4.0 license
http://creativecommons.org/licenses/by-nc-nd/4.0/

For Problem **(P2)**, one obtains the same system as above, except with $\mathcal{K}_1$ replaced by

$$\mathcal{K}_2 = \begin{bmatrix} \tau^{-1}\mathbf{M}+\mathbf{K} & B^T & & & & & -\tau^{-1}\mathbf{M} & 0 \\ B & 0 & & & & & 0 & 0 \\ -\tau^{-1}\mathbf{M} & 0 & \tau^{-1}\mathbf{M}+\mathbf{K} & B^T & & & & \\ 0 & 0 & B & 0 & \ddots & & & \\ & & & \ddots & \ddots & \ddots & & \\ & & & & \ddots & \ddots & \ddots & \\ & & & & -\tau^{-1}\mathbf{M} & 0 & \tau^{-1}\mathbf{M}+\mathbf{K} & B^T \\ & & & & 0 & 0 & B & 0 \end{bmatrix},$$

and the right-hand side vector without the $\tau^{-1}\widetilde{\mathbf{v}}_0$ term, to take account of the periodic boundary condition.

Our next consideration is the cost functional

$$J(\vec{v},\vec{u}) = \frac{1}{2}\int_0^T \int_\Omega \|\vec{v}-\vec{v}_d\|^2 \ \mathrm{d}\Omega\mathrm{d}t + \frac{\beta}{2}\int_0^T \int_\Omega \|\vec{u}\|^2 \ \mathrm{d}\Omega\mathrm{d}t,$$

that we wish to minimize. We approximate this on the discrete space by

$$\mathcal{J}(\mathbf{v},\mathbf{u}) = \frac{\tau}{2}\sum_{j=1}^{N_t}\mathbf{v}_j^T\mathbf{M}\mathbf{v}_j - \tau\sum_{j=1}^{N_t}\mathbf{z}_j^T\mathbf{v}_j + \frac{\beta\tau}{2}\sum_{j=1}^{N_t}\mathbf{u}_j^T\mathbf{M}\mathbf{u}_j,$$

where we neglect additive constants independent of $\mathbf{v}$ and $\mathbf{u}$. Here $\mathbf{z}_j$ corresponds to integrals of the form $\int_\Omega \vec{v}_d\phi_i \ \mathrm{d}\Omega$ at the j-th time-step. Naturally the precise structure of $\mathcal{J}(\mathbf{v},\mathbf{u})$ will depend on the quadrature rule(s) employed, but we note that the methodology introduced in this paper may be tailored to any reasonable choices made.

At this point, the discrete Lagrangian is given by

$$\mathcal{L}(\mathbf{v},\mathbf{p},\mathbf{u},\boldsymbol{\lambda},\boldsymbol{\mu}) = \mathcal{J}(\mathbf{v},\mathbf{u}) + \mathbf{q}^T\left(\mathcal{K}_{1/2}\mathbf{y} - \mathcal{N}\mathbf{u} - \mathbf{f}\right),$$

where

$$\begin{aligned} \mathbf{y} &= \left[\begin{array}{ccccccc} \mathbf{v}_1^T & \mathbf{p}_1^T & \mathbf{v}_2^T & \mathbf{p}_2^T & \cdots & \mathbf{v}_{N_t}^T & \mathbf{p}_{N_t}^T \end{array}\right]^T, \\ \mathbf{u} &= \left[\begin{array}{cccc} \mathbf{u}_1^T & \mathbf{u}_2^T & \cdots & \mathbf{u}_{N_t}^T \end{array}\right]^T, \\ \mathbf{q} &= \left[\begin{array}{ccccccc} \boldsymbol{\lambda}_1^T & \boldsymbol{\mu}_1^T & \boldsymbol{\lambda}_2^T & \boldsymbol{\mu}_2^T & \cdots & \boldsymbol{\lambda}_{N_t}^T & \boldsymbol{\mu}_{N_t}^T \end{array}\right]^T, \end{aligned}$$

and $\mathcal{K}_{1/2}$ denotes $\mathcal{K}_1$ or $\mathcal{K}_2$ depending on whether Problem **(P1)** or **(P2)** is being solved.

The quantities $\boldsymbol{\lambda}_1,...,\boldsymbol{\lambda}_{N_t}$ and $\boldsymbol{\mu}_1,...,\boldsymbol{\mu}_{N_t}$ denote the discretized versions of the *adjoint variables* $\underline{\boldsymbol{\lambda}}$ and μ at each time-step. Formally we state that $\underline{\boldsymbol{\lambda}}$ is the adjoint variable to $\mathbf{v}$ (and we therefore discretize this using **Q2** basis functions), and that μ is the adjoint variable to p (and we therefore discretize this using **Q1** basis functions).

To obtain the discrete optimality conditions, we must now differentiate with respect to the adjoint variables $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$, the control variable $\mathbf{u}$, and the state variables $\mathbf{v}$ and $\mathbf{p}$. Doing so results in the following matrix system:

$$\begin{bmatrix} \tau\mathcal{M}_0 & 0 & \mathcal{K}_{1/2}^T \\ 0 & \beta\tau\mathcal{M} & -\mathcal{N}^T \\ \mathcal{K}_{1/2} & -\mathcal{N} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ \mathbf{u} \\ \mathbf{q} \end{bmatrix} = \begin{bmatrix} \tau\mathbf{z} \\ \mathbf{0} \\ \mathbf{f} \end{bmatrix}, \tag{6}$$

where

$$\begin{aligned} \mathcal{M}_0 &= \mathrm{blkdiag}\left(\mathbf{M},0,\mathbf{M},0,...,\mathbf{M},0\right), \\ \mathcal{M} &= \mathrm{blkdiag}\left(\mathbf{M},\mathbf{M},...,\mathbf{M}\right), \\ \mathbf{z} &= \left[\begin{array}{cccc} \mathbf{z}_1^T & \mathbf{z}_2^T & \cdots & \mathbf{z}_{N_t}^T \end{array}\right]^T. \end{aligned}$$

© 2016. This manuscript version is made available under the CC-BY-NC-ND 4.0 license
http://creativecommons.org/licenses/by-nc-nd/4.0/

The first line of the matrix system (6) relates to the *adjoint equations*, the second line to the *gradient equation*, and the third line to the *state equations*.

Eliminating the gradient equation from (6) enables us to reduce the system as follows:

$$\begin{bmatrix} \tau\mathcal{M}_0 & \mathcal{K}_{1/2}^T \\ \mathcal{K}_{1/2} & -\beta^{-1}\tau^{-1}\mathcal{M}_0 \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ \mathbf{q} \end{bmatrix} = \begin{bmatrix} \tau\mathbf{z} \\ \mathbf{f} \end{bmatrix}. \tag{7}$$

We now consider a rearrangement of the reduced matrix system (7) (with an initial focus on Problem **(P1)**) that is designed to make it easier to consider a preconditioner for the matrix system. We note that we may reorder the system to the form

$$\underbrace{\begin{bmatrix} \Delta & \Sigma^T & & & \\ \Sigma & \Delta & \Sigma^T & & \\ & \Sigma & \Delta & \ddots & \\ & & \ddots & \ddots & \Sigma^T \\ & & & \Sigma & \Delta \end{bmatrix}}_{\mathcal{A}} \mathbf{x} = \mathbf{b}, \tag{8}$$

where

$$\Delta = \begin{bmatrix} \tau\mathbf{M} & \tau^{-1}\mathbf{M}+\mathbf{K} & B^T & 0 \\ \tau^{-1}\mathbf{M}+\mathbf{K} & -\beta^{-1}\tau^{-1}\mathbf{M} & 0 & B^T \\ B & 0 & 0 & 0 \\ 0 & B & 0 & 0 \end{bmatrix}, \quad \Sigma = \begin{bmatrix} 0 & 0 & 0 & 0 \\ -\tau^{-1}\mathbf{M} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{9}$$

$$\mathbf{x} = \left[\mathbf{v}_1^T, \boldsymbol{\lambda}_1^T, \boldsymbol{\mu}_1^T, \mathbf{p}_1^T, \mathbf{v}_2^T, \boldsymbol{\lambda}_2^T, \boldsymbol{\mu}_2^T, \mathbf{p}_2^T, ..., \mathbf{v}_{N_t}^T, \boldsymbol{\lambda}_{N_t}^T, \boldsymbol{\mu}_{N_t}^T, \mathbf{p}_{N_t}^T\right]^T,$$

and $\mathbf{b}$ is the appropriate reordered right-hand side vector. We note that the matrix Δ corresponds, up to multiplicative constants, to the optimality conditions of the (generalized) time-independent Stokes control system. To demonstrate how the structure of our rearranged matrix system changes as N_t is increased, we present Matlab `spy` plots of $\mathcal{A}$ for various N_t in Figure 1.

At this stage, we make a very simple observation. This is that instead of minimizing $J(\vec{v}, \vec{u})$ subject to PDE constraints, we could equally minimize $\gamma J(\vec{v}, \vec{u})$ for any chosen (positive) factor γ. This would give the same solution as that for the original problem.

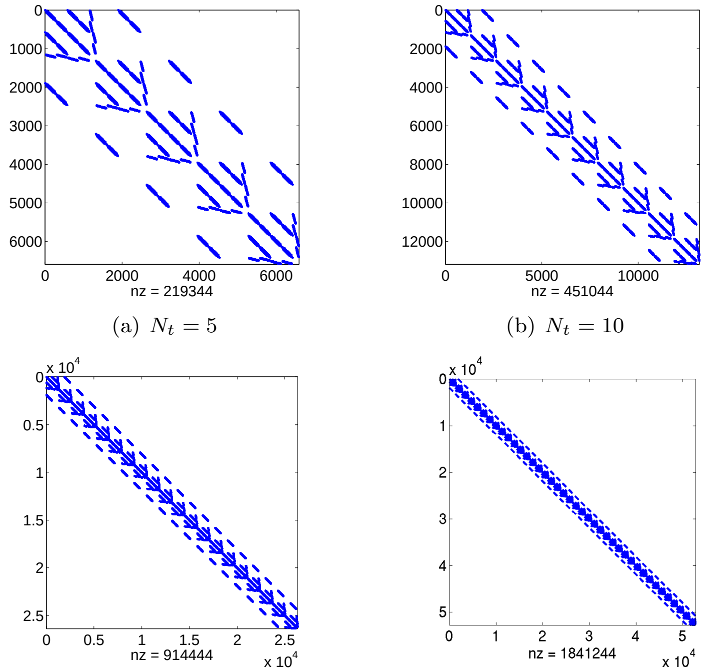
When solving Problem **(P1)** in this way on the discrete level, this leads to a matrix system of the form (8), with Σ as in (9) and

$$\Delta = \begin{bmatrix} \gamma\tau\mathbf{M} & \tau^{-1}\mathbf{M}+\mathbf{K} & B^T & 0 \\ \tau^{-1}\mathbf{M}+\mathbf{K} & -\gamma^{-1}\beta^{-1}\tau^{-1}\mathbf{M} & 0 & B^T \\ B & 0 & 0 & 0 \\ 0 & B & 0 & 0 \end{bmatrix}. \tag{10}$$

It is reasonable to consider modifying the value of γ taken when solving the problem (perhaps to improve the conditioning of the problem, or for another reason). We believe that three intuitive choices for this selection are as follows:

- $\gamma = 1$: This is a natural choice, as it is perhaps the one that is most faithful to the original problem formulation. This is also the choice of parameter made in [28] when the authors considered solvers for this problem.
- $\gamma = \tau^{-2}$: This selection of γ has the effect of "balancing" the $(1, 1)$-entry of Δ with the $(2, 1)$-entry of Σ. As we wish the block diagonal entries of $\mathcal{A}$ to dominate the character of the matrix in some sense, it is reasonable to ensure that the entries of Δ and Σ behave in this way.

© 2016. This manuscript version is made available under the CC-BY-NC-ND 4.0 license
http://creativecommons.org/licenses/by-nc-nd/4.0/



(a) $N_t = 5$ (b) $N_t = 10$

(c) $N_t = 20$ (d) $N_t = 40$

Figure 1: MATLAB **spy** plots of matrix systems of the form (8) for Problem **(P1)**, with $h = 2^{-3}$ and different values of N_t.

- $\gamma = \beta^{-1/2}\tau^{-1}$: By constrast this choice "balances" the $(1,1)$- and $(2,2)$-entries of Δ, so that when we seek effective approximations or analysis of Δ itself we are able to exploit this property.

Making such choices leads to systems of the form (8) with

$$\Delta = \begin{bmatrix} \tau\mathbf{M} & \tau^{-1}\mathbf{M}+\mathbf{K} & B^T & 0 \\ \tau^{-1}\mathbf{M}+\mathbf{K} & -\beta^{-1}\tau^{-1}\mathbf{M} & 0 & B^T \\ B & 0 & 0 & 0 \\ 0 & B & 0 & 0 \end{bmatrix}, \quad \text{for } \gamma = 1,$$

$$\Delta = \begin{bmatrix} \tau^{-1}\mathbf{M} & \tau^{-1}\mathbf{M}+\mathbf{K} & B^T & 0 \\ \tau^{-1}\mathbf{M}+\mathbf{K} & -\beta^{-1}\tau\mathbf{M} & 0 & B^T \\ B & 0 & 0 & 0 \\ 0 & B & 0 & 0 \end{bmatrix}, \quad \text{for } \gamma = \tau^{-2},$$

$$\Delta = \begin{bmatrix} \beta^{-1/2}\mathbf{M} & \tau^{-1}\mathbf{M}+\mathbf{K} & B^T & 0 \\ \tau^{-1}\mathbf{M}+\mathbf{K} & -\beta^{-1/2}\mathbf{M} & 0 & B^T \\ B & 0 & 0 & 0 \\ 0 & B & 0 & 0 \end{bmatrix}, \quad \text{for } \gamma = \beta^{-1/2}\tau^{-1}.$$

In each case Σ is as in (9).

We emphasize that, for each choice of γ, we are considering a matrix system that is of extremely high dimension (especially for finer discretizations in space and time), and one that contains sparse matrices $\mathbf{M}$, $\mathbf{K}$ and B. Our aim is for the solver to be robust with respect to the value of γ chosen, as well as h, N_t and β.

We also note at this stage that we could build a matrix system of exactly the same structure for a distributed control problem with other boundary conditions (Neumann or mixed conditions, for example).

For the time-periodic problem **(P2)**, applying the same working and reordering as above leads to the

© 2016. This manuscript version is made available under the CC-BY-NC-ND 4.0 license
http://creativecommons.org/licenses/by-nc-nd/4.0/

following matrix system which needs to be solved:

$$\underbrace{\begin{bmatrix} \Delta & \Sigma^T & & & \Sigma \\ \Sigma & \Delta & \Sigma^T & & \\ & \Sigma & \Delta & \ddots & \\ & & \ddots & \ddots & \Sigma^T \\ \Sigma^T & & & \Sigma & \Delta \end{bmatrix}}_{\mathcal{A}} \mathbf{x} = \mathbf{b}, \tag{11}$$

with the same definitions of Δ, Σ and $\mathbf{x}$ as for Problem **(P1)**.

When solving either (8) or (11), relating to Problem **(P1)** or **(P2)**, one is dealing with a matrix system of very high dimension when fine discretizations in space or time are taken. This renders a direct method inappropriate for solving such systems, and motivates the work presented in the next section, concerning the construction of preconditioned iterative methods for these systems.

3. Preconditioning and iterative solvers

Now that we have derived the relevant matrix systems which we need to solve when examining the problems **(P1)** and **(P2)**, we wish to construct effective preconditioners for the matrices which may be applied within a suitable iterative method. A particular focus when developing these preconditioners is the performance of our iterative method for smaller values of β, as such values will enable the velocity $\vec{v}$ to match the desired state $\vec{v}_d$ more closely.

Observe that in the previous section we have made specific choices of γ, with the goal that the properties of the matrix $\mathcal{A}$ (for either problem, **(P1)** or **(P2)**) are determined to a large extent by the matrices Δ appearing on the block diagonal of $\mathcal{A}$. That is to say, we assume that Σ makes a small contribution to the spectral properties of the matrix systems compared with Δ.

If this is the case, it is natural to consider the following approximation of $\mathcal{A}$ (for either problem):

$$\mathcal{A} \approx \text{blkdiag}\big(\Delta, \Delta, ..., \Delta, \Delta\big),$$

and use this to build (symmetric) preconditioners of the form

$$\widehat{\mathcal{P}} = \text{blkdiag}\big(\widehat{\Delta}, \widehat{\Delta}, ..., \widehat{\Delta}, \widehat{\Delta}\big), \tag{12}$$

where $\widehat{\Delta}$ denotes a suitable symmetric approximation of Δ.

Alternatively, if we are content to utilize nonsymmetric solvers, we may implement such a method alongside a preconditioner of the form

$$\widehat{\mathcal{P}} = \begin{bmatrix} \widehat{\Delta} & & & & \\ \Sigma & \widehat{\Delta} & & & \\ & \Sigma & \widehat{\Delta} & & \\ & & \ddots & \ddots & \\ & & & \Sigma & \widehat{\Delta} \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} \widehat{\Delta} & & & & \\ \Sigma & \widehat{\Delta} & & & \\ & \Sigma & \widehat{\Delta} & & \\ & & \ddots & \ddots & \\ \Sigma^T & & & \Sigma & \widehat{\Delta} \end{bmatrix},$$

depending on whether Problem **(P1)** or **(P2)** is being solved.

3.1. Approximating Δ

Therefore our next task is to construct a suitable approximation of the matrix Δ, as defined in (10). This approximation must take into account the parameter γ chosen, as well as β and τ.

© 2016. This manuscript version is made available under the CC-BY-NC-ND 4.0 license
http://creativecommons.org/licenses/by-nc-nd/4.0/

The matrix Δ is itself a saddle point system of the general form (1) (with $\Theta = 0$). We first wish to approximate the $(1,1)$-block

$$\Phi = \begin{bmatrix} \gamma\tau\mathbf{M} & \tau^{-1}\mathbf{M}+\mathbf{K} \\ \tau^{-1}\mathbf{M}+\mathbf{K} & -\gamma^{-1}\beta^{-1}\tau^{-1}\mathbf{M} \end{bmatrix}.$$

Note now that the $(1,1)$-block of the matrix Δ is itself a saddle point system. An 'ideal' approximation is therefore given by

$$\widetilde{\Phi} := \begin{bmatrix} \gamma\tau\mathbf{M} & 0 \\ 0 & \gamma^{-1}\tau^{-1}(\tau^{-1}\mathbf{M}+\mathbf{K})\mathbf{M}^{-1}(\tau^{-1}\mathbf{M}+\mathbf{K})+\gamma^{-1}\beta^{-1}\tau^{-1}\mathbf{M} \end{bmatrix},$$

using a block diagonal approximation of the form (3).

However, this is not yet an approximation that we can use, as we would need to compute the matrix $\mathbf{M}^{-1}$ (which is in general dense even though $\mathbf{M}$ is sparse) to evaluate the exact Schur complement $\gamma^{-1}\tau^{-1}(\tau^{-1}\mathbf{M}+\mathbf{K})\mathbf{M}^{-1}(\tau^{-1}\mathbf{M}+\mathbf{K})+\gamma^{-1}\beta^{-1}\tau^{-1}\mathbf{M}$.

We therefore need to construct an effective Schur complement approximation – we do this using a "matching strategy" discussed in [22, 23, 24, 25] to conclude that the matrix

$$\left[\gamma^{-1}\tau^{-1}\left((\tau^{-1}+\beta^{-1/2})\mathbf{M}+\mathbf{K}\right)\mathbf{M}^{-1}\left((\tau^{-1}+\beta^{-1/2})\mathbf{M}+\mathbf{K}\right)\right]^{-1} \left[\gamma^{-1}\tau^{-1}(\tau^{-1}\mathbf{M}+\mathbf{K})\mathbf{M}^{-1}(\tau^{-1}\mathbf{M}+\mathbf{K})+\gamma^{-1}\beta^{-1}\tau^{-1}\mathbf{M}\right] \tag{13}$$

has eigenvalues which are clustered within a tight range. From this statement, we can say that the Schur complement of Φ may be well approximated by $\gamma^{-1}\tau^{-1}\left((\tau^{-1}+\beta^{-1/2})\mathbf{M}+\mathbf{K}\right)\mathbf{M}^{-1}\left((\tau^{-1}+\beta^{-1/2})\mathbf{M}+\mathbf{K}\right)$. To demonstrate this, we may follow a very similar strategy as in [24]: as we work exclusively with positive definite matrices (a well known property of finite element mass and stiffness matrices [6]), we may bound the eigenvalues by considering the following Rayleigh quotient (for real $\mathbf{v}\neq\mathbf{0}$):

$$\begin{aligned} R &:= \frac{\mathbf{v}^T\left[\gamma^{-1}\tau^{-1}(\tau^{-1}\mathbf{M}+\mathbf{K})\mathbf{M}^{-1}(\tau^{-1}\mathbf{M}+\mathbf{K})+\gamma^{-1}\beta^{-1}\tau^{-1}\mathbf{M}\right]\mathbf{v}}{\mathbf{v}^T\left[\gamma^{-1}\tau^{-1}\left((\tau^{-1}+\beta^{-1/2})\mathbf{M}+\mathbf{K}\right)\mathbf{M}^{-1}\left((\tau^{-1}+\beta^{-1/2})\mathbf{M}+\mathbf{K}\right)\right]\mathbf{v}} \\ &= \frac{\mathbf{a}_1^T\mathbf{a}_1+\mathbf{a}_2^T\mathbf{a}_2}{(\mathbf{a}_1+\mathbf{a}_2)^T(\mathbf{a}_1+\mathbf{a}_2)}, \end{aligned}$$

where $\mathbf{a}_1 = \mathbf{M}^{-1/2}(\tau^{-1}\mathbf{M}+\mathbf{K})\mathbf{v}$ and $\mathbf{a}_2 = \beta^{-1/2}\mathbf{M}^{1/2}\mathbf{v}$. We may now use the observation that $\mathbf{a}_1^T\mathbf{a}_2 > 0$ to deduce that the Rayleigh quotient is less than 1 (by expanding out the terms of $(\mathbf{a}_1+\mathbf{a}_2)^T(\mathbf{a}_1+\mathbf{a}_2)$). We may also use straightforward algebraic manipulation to determine that the quotient is at least $\frac{1}{2}$ (as clearly $\mathbf{a}_2^T\mathbf{a}_2 > 0$, and it may be verified that $\mathbf{a}_1^T\mathbf{a}_1+\mathbf{a}_2^T\mathbf{a}_2 \geq \frac{1}{2}(\mathbf{a}_1+\mathbf{a}_2)^T(\mathbf{a}_1+\mathbf{a}_2)$). Therefore the eigenvalues of (13) are all contained in $[\frac{1}{2},1)$.

This leads to the following approximation of Φ:

$$\widehat{\Phi} = \begin{bmatrix} \gamma\tau\mathbf{M} & 0 \\ 0 & \gamma^{-1}\tau^{-1}\Big((\tau^{-1}+\beta^{-1/2})\mathbf{M}+\mathbf{K}\Big)\mathbf{M}^{-1}\Big((\tau^{-1}+\beta^{-1/2})\mathbf{M}+\mathbf{K}\Big) \end{bmatrix}.$$

To embed this approximation into a general saddle point preconditioner of the form (3), we now need to find an approximation of the Schur complement of Δ:

$$S = \begin{bmatrix} B & 0 \\ 0 & B \end{bmatrix} \begin{bmatrix} \gamma\tau\mathbf{M} & \tau^{-1}\mathbf{M}+\mathbf{K} \\ \tau^{-1}\mathbf{M}+\mathbf{K} & -\gamma^{-1}\beta^{-1}\tau^{-1}\mathbf{M} \end{bmatrix}^{-1} \begin{bmatrix} B^T & 0 \\ 0 & B^T \end{bmatrix}.$$

The heuristic argument used to tackle simpler time-independent Stokes and Navier-Stokes control problems [20, 21], which was found to be very effective for a range of examples, is to approximate the Schur complement

© 2016. This manuscript version is made available under the CC-BY-NC-ND 4.0 license
http://creativecommons.org/licenses/by-nc-nd/4.0/

by

$$\mathbf{B}_2 \begin{bmatrix} \gamma\tau\mathbf{M} & 0 \\ 0 & \gamma^{-1}\tau^{-1}\Big[(\tau^{-1}\mathbf{M}+\mathbf{K})\mathbf{M}^{-1}(\tau^{-1}\mathbf{M}+\mathbf{K})+\beta^{-1}\mathbf{M}\Big] \end{bmatrix}^{-1} \mathbf{B}_2^T$$
$$= \begin{bmatrix} \gamma^{-1}\tau^{-1}B\mathbf{M}^{-1}B^T & 0 \\ 0 & \gamma\tau B\Big(\mathbf{K}\mathbf{M}^{-1}\mathbf{K}+2\tau^{-1}\mathbf{K}+(\tau^{-2}+\beta^{-1})\mathbf{M}\Big)^{-1}B^T \end{bmatrix},$$

where $\mathbf{B}_2 = \begin{bmatrix} B & 0 \\ 0 & B \end{bmatrix}$. In other words, within the expression for the Schur complement we replace the inverse of Φ by the inverse of its saddle point approximation.

The question at this point is how $B\mathbf{M}^{-1}B^T$ and $B\left(\mathbf{K}\mathbf{M}^{-1}\mathbf{K}+2\tau^{-1}\mathbf{K}+(\tau^{-2}+\beta^{-1})\mathbf{M}\right)^{-1}B^T$ may be approximated (as B is a rectangular matrix and therefore not invertible). Happily, it is well known that $B\mathbf{M}^{-1}B^T$ may be well approximated by the stiffness matrix on the pressure space K_p (see [6, Chapter 8] for instance). This leaves the representation of $B\left(\mathbf{K}\mathbf{M}^{-1}\mathbf{K}+2\tau^{-1}\mathbf{K}+(\tau^{-2}+\beta^{-1})\mathbf{M}\right)^{-1}B^T$ as the next main challenge.

To do this we employ a *commutator argument*, which has been utilized by Cahouet and Chabard [3], and others. In more detail, we assume that the commutator

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

is approximately zero on the continuous space for some suitable operator $\mathcal{L}$, with corresponding operator $(\mathcal{L})_p$ on the pressure space. Furthermore we assume that its discrete representation

$$\mathcal{E}_h = (\mathbf{M}^{-1}\mathbf{L})\mathbf{M}^{-1}B^T - \mathbf{M}^{-1}B^T(M_p^{-1}L_p),$$

is also small in some sense. Here $\mathbf{L}$ is the discrete representation of the continuous operator $\mathcal{L}$. We observe that making the choices $\mathcal{L} = \nabla^4 - 2\tau^{-1}\nabla^2 + (\tau^{-2}+\beta^{-1})I$ and $\mathbf{L} = \mathbf{K}\mathbf{M}^{-1}\mathbf{K}+2\tau^{-1}\mathbf{K}+(\tau^{-2}+\beta^{-1})\mathbf{M}$ (with L_p, by which we denote the discrete representation of $(\mathcal{L})_p$, therefore given by $K_pM_p^{-1}K_p+2\tau^{-1}K_p+(\tau^{-2}+\beta^{-1})M_p$), may enable us to use these assumptions to approximate the matrix of interest. We do this by writing that

$$\begin{aligned} \mathcal{E}_h = {} & \Big(\mathbf{M}^{-1}\mathbf{K}\mathbf{M}^{-1}\mathbf{K}+2\tau^{-1}\mathbf{M}^{-1}\mathbf{K}+(\tau^{-2}+\beta^{-1})\mathbf{I}\Big)\mathbf{M}^{-1}B^T \\ & - \mathbf{M}^{-1}B^T\Big(M_p^{-1}K_pM_p^{-1}K_p+2\tau^{-1}M_p^{-1}K_p+(\tau^{-2}+\beta^{-1})I_p\Big) \approx 0, \end{aligned} \tag{14}$$

where I_p denotes the identity matrix on the pressure space. Pre-multiplying (14) by $B\mathbf{L}^{-1}\mathbf{M}$ and post-multiplying by $L_p^{-1}M_p$ gives that

$$\begin{aligned} & B\Big(\mathbf{K}\mathbf{M}^{-1}\mathbf{K}+2\tau^{-1}\mathbf{K}+(\tau^{-2}+\beta^{-1})\mathbf{M}\Big)^{-1}B^T \\ & \qquad \approx B\mathbf{M}^{-1}B^T\Big(K_pM_p^{-1}K_p+2\tau^{-1}K_p+(\tau^{-2}+\beta^{-1})M_p\Big)^{-1}M_p \\ & \qquad \approx K_p\Big(K_pM_p^{-1}K_p+2\tau^{-1}K_p+(\tau^{-2}+\beta^{-1})M_p\Big)^{-1}M_p \\ & \qquad = \Big(M_p^{-1}K_pM_p^{-1}+2\tau^{-1}M_p^{-1}+(\tau^{-2}+\beta^{-1})K_p^{-1}\Big)^{-1}, \end{aligned} \tag{15}$$

again using that $B\mathbf{M}^{-1}B^T \approx K_p$ to obtain (15). We emphasize that a commutator argument of this form was also applied in [20] to the time-independent Stokes control problem. The matrix under consideration was $B\big(\mathbf{K}\mathbf{M}^{-1}\mathbf{K}+\beta^{-1}\mathbf{M}\big)^{-1}B^T$ for the problem at hand: the added complexities that arise in the time-dependent setting are an additional term in $\mathbf{K}$ within the matrix $\mathbf{L}$, and the incorporation of the time-step τ into the commutator argument. Whereas this is a heuristic argument by nature, it has been found to be

© 2016. This manuscript version is made available under the CC-BY-NC-ND 4.0 license
http://creativecommons.org/licenses/by-nc-nd/4.0/

highly effective when approximating matrices of the form $B\mathbf{L}^{-1}B^T$ for a range of fluid dynamics problems (we refer to [3, 6, 9, 20, 21], for example).

We may therefore state a proposed approximation for the Schur complement S:

$$\widehat{S} = \begin{bmatrix} \gamma^{-1}\tau^{-1}K_p & 0 \\ 0 & \gamma\tau\Big(M_p^{-1}K_pM_p^{-1} + 2\tau^{-1}M_p^{-1} + (\tau^{-2}+\beta^{-1})K_p^{-1}\Big)^{-1} \end{bmatrix}.$$

Combining the approximations for the $(1,1)$-block and Schur complement of Δ, we may now write a block diagonal preconditioner for Δ of the form

$$\widehat{\Delta} = \mathrm{blkdiag}\left(\widehat{\Phi},\widehat{S}\right) = \begin{bmatrix} \gamma\tau\mathbf{M} & 0 & 0 & 0 \\ 0 & \mathcal{P}_{22} & 0 & 0 \\ 0 & 0 & \gamma^{-1}\tau^{-1}K_p & 0 \\ 0 & 0 & 0 & \mathcal{P}_{44} \end{bmatrix},$$

where

$$\mathcal{P}_{22} = \gamma^{-1}\tau^{-1}\Big((\tau^{-1}+\beta^{-1/2})\mathbf{M}+\mathbf{K}\Big)\mathbf{M}^{-1}\Big((\tau^{-1}+\beta^{-1/2})\mathbf{M}+\mathbf{K}\Big),$$
$$\mathcal{P}_{44} = \gamma\tau\Big(M_p^{-1}K_pM_p^{-1} + 2\tau^{-1}M_p^{-1} + (\tau^{-2}+\beta^{-1})K_p^{-1}\Big)^{-1}.$$

This approximation is symmetric and positive definite, which is highly desirable from an iterative solver point-of-view. We point out again that it is perfectly possible to construct block triangular preconditioners of similar form.

At each step of an iterative scheme, we will need to apply $\widehat{\Delta}^{-1}$ a number of times equal to the number of time-steps used in the problem formulation (see (12)).

When we apply $\widehat{\Delta}^{-1}$, we use Chebyshev semi-iteration [31] to approximate the inverse of a mass matrix, and the Harwell Subroutine Library (HSL) algebraic multigrid code `HSL_MI20` [2] or the Aggregation-Based Algebraic Multigrid (`AGMG`) code [14, 15, 16, 17] to similarly approximate the inverse of a stiffness matrix (or a sum of stiffness and mass matrices). Each application of $\widehat{\Delta}^{-1}$ therefore requires three Chebyshev semi-iteration processes (one for $\mathbf{M}$ and two for M_p), and four multigrid approximations (two for each of $\left(\tau^{-1}+\beta^{-1/2}\right)\mathbf{M}+\mathbf{K}$ and K_p).

We highlight two attractive features of the preconditioning strategy presented in this section:

1. The iterative solver used, whether a block diagonal or block triangular preconditioner is applied, requires the storage of matrices ($\mathbf{M}$, $\mathbf{K}$ and B) which are small in comparison to the dimension of the matrix $\mathcal{A}$. This enables the solution of much larger systems than would be possible when using a direct method.
2. In addition, the block diagonal preconditioner has a structure which would make it possible to implement the iterative method in parallel – this opens up the possibility of solving such complex problems over a large number of computational units.

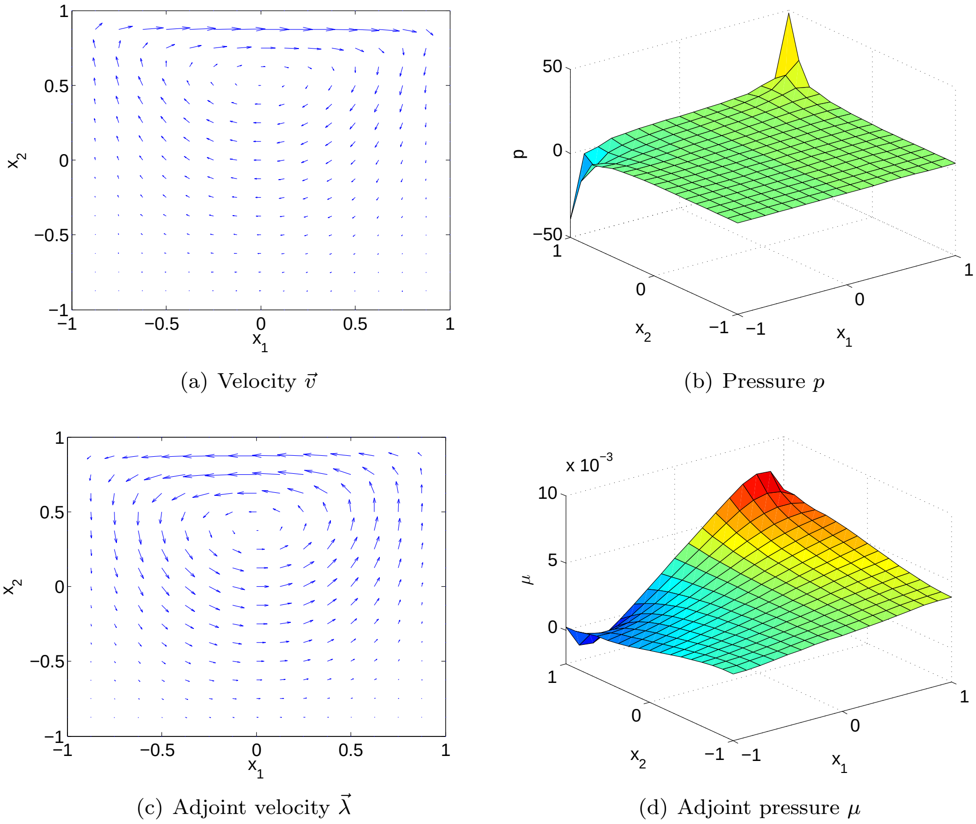
We also wish to briefly comment on the potential for solving Stokes control problems of different form to the distributed control problem considered in this section: for example, it would be reasonable to consider boundary control problems, or problems where the control variable is applied only on some subdomain of Ω. It is feasible to apply the methodology introduced in this paper to problems of these forms, although one will face two additional challenges. Firstly, the approximation of the Schur complement of Φ using a matching strategy becomes less rigorous in nature, i.e. an eigenvalue bound of the form $[\frac{1}{2},1)$ cannot be shown, due to the different types of mass matrices in Φ. However, promising results have nonetheless been observed using this strategy for problems of Poisson control and heat equation control form (see [19, Chapter 4] and [23]). Secondly, the commutator argument used to approximate S will need to be adjusted to take account of the structure of the new mass matrices arising in Φ. An additional alteration to the problem set-up would be to introduce non-uniform time-stepping, rather than using a uniform time-step τ as considered in this

© 2016. This manuscript version is made available under the CC-BY-NC-ND 4.0 license
http://creativecommons.org/licenses/by-nc-nd/4.0/

paper. If such a modification were introduced, the matrices Δ and Σ would change at each time-step, with new factors τ_i, $i = 1, ..., N_t$ replacing τ. However one would still be able to apply the same strategies for constructing $\widehat{\Delta}$ as for the uniform time-stepping routine, provided one incorporates the different scalings that arise at each time-step.

In the next section we demonstrate the practical performance of our (block diagonal) preconditioner when solving problems **(P1)** and **(P2)**. We implement our preconditioner within the MINRES algorithm [18], as this is the method of choice when solving symmetric indefinite systems with symmetric positive definite preconditioners.

4. Numerical results



(a) Velocity $\vec{v}$ (b) Pressure p

(c) Adjoint velocity $\vec{\lambda}$ (d) Adjoint pressure μ

Figure 2: Solution plots for velocity $\vec{v}$, pressure p, adjoint velocity $\vec{\lambda}$ and adjoint pressure μ for the first example of Problem **(P1)**, at the final time-step with $\beta = 10^{-2}$.

Having motivated our preconditioning strategy, we now wish to verify the potency of our iterative solver with a number of numerical experiments. These involve problems of the form **(P1)** (with Dirichlet boundary conditions) and **(P2)** (with periodic boundary conditions). Within these problem set-ups, we consider two examples on the domain $\Omega := [-1, 1]^2$, with zero initial conditions specified when a problem of the form **(P1)** is examined (although our method is of course general and does not require such a condition). The first example we test is of classical *lid-driven cavity* form (see [6, Chapters 6 & 8] for instance), with

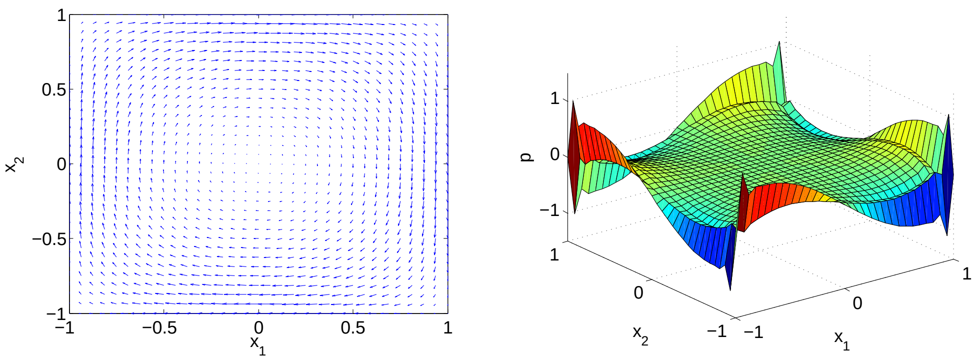
$$\vec{v}_d = 0, \quad \text{on } \Omega \times [0, T],$$

$$\vec{v} = \begin{cases} [1,\ 0]^T & \text{on } [-1, 1] \times \{1\} \times [0, T], \\ [0,\ 0]^T & \text{on } \partial\Omega \backslash \big([-1, 1] \times \{1\}\big) \times [0, T]. \end{cases}$$

© 2016. This manuscript version is made available under the CC-BY-NC-ND 4.0 license
http://creativecommons.org/licenses/by-nc-nd/4.0/

		h				
		2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
N_t	5	$1,870$	$6,590$	$24,670$	$95,390$	$\mathbf{375,090}$
	10	$3,740$	$13,180$	$49,340$	$190,780$	$\mathbf{750,180}$
	20	$7,480$	$26,360$	$98,680$	$\mathbf{381,560}$	$\mathbf{1,500,360}$
	40	$14,960$	$52,720$	$197,360$	$\mathbf{763,120}$	$\mathbf{3,000,720}$
	80	$29,920$	$105,440$	$\mathbf{394,720}$	$\mathbf{1,526,240}$	$\mathbf{6,001,440}$

Table 1: Dimensions of matrix systems when solving Problems **(P1)** and **(P2)** for different values of h and N_t. Problem dimensions in bold are those which could not be solved with a direct method on the processor used.



(a) Velocity $\vec{v}$

(b) Pressure p

Figure 3: Solution plots for velocity $\vec{v}$ and pressure p for the second example of Problem **(P1)**, at the final time-step with $\beta = 10^{-2}$.

The second example we examine is of a similar form to that considered by the author in [21], involving a recirculating wind close to $\partial\Omega$. In this case, we take

$$\vec{v}_d = \begin{cases} \left[\frac{1}{2}x_2(1-x_1^2),\ \ -\frac{1}{2}x_1(1-x_2^2)\right]^T & \text{if } x_1^2+x_2^2 \geq \frac{1}{2}, \\ [0,\ \ 0]^T & \text{otherwise}, \end{cases}$$
$$\vec{v} = \vec{v}_d, \quad \text{on } \partial\Omega \times [0,T],$$

where $\mathbf{x} := [x_1,\ \ x_2]^T$. Plots of solutions for each of these examples are provided in Figures 2 and 3 respectively. We highlight that the entries c_i, d_i of the vectors $\mathbf{c}$, $\mathbf{d}$ in (5) are of the following form:

$$c_i = -\sum_{j=n_u+1}^{n_u+n_\partial} V_j \int_\Omega \nabla\vec{\phi}_i : \nabla\vec{\phi}_j \ \mathrm{d}\Omega, \qquad d_i = \sum_{j=n_u+1}^{n_u+n_\partial} V_j \int_\Omega \psi_i \nabla\cdot\vec{\phi}_j \ \mathrm{d}\Omega,$$

where $V_{n_u+1}, ..., V_{n_u+n_\partial}$ denote the values of the components of $\vec{v}$ at the boundary nodes, and $\vec{\phi}_i$ is a d-vector function with entries ϕ_i. So for the first test problem, $\mathbf{c}$ and $\mathbf{d}$ contain contributions from the boundary conditions along $x_2 = 1$; for the second test problem, the vectors are terms containing the values of $\vec{v}_d$ along the boundary.

We compute numerical solutions to these problems in Matlab, using the Minres algorithm with the preconditioner derived in the previous section. The problems are solved to a preconditioned residual tolerance of 10^{-5}. In order to construct the relevant (stiffness, mass and divergence) matrices we make use of the Ifiss software package [4, 5, 26]. When approximating the inverse of a mass matrix we apply 20 steps of

© 2016. This manuscript version is made available under the CC-BY-NC-ND 4.0 license
http://creativecommons.org/licenses/by-nc-nd/4.0/

(P1) $\beta = 10^{-2}$		h				
		2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
N_t	5	63	73	77	81	87
		2.36	*3.47*	*5.94*	*14.4*	*59.5*
	10	64	74	79	80	84
		4.57	*7.05*	*12.7*	*29.3*	*114*
	20	88	92	100	101	107
		12.9	*18.4*	*33.3*	*75.4*	*298*
	40	174	181	186	190	188
		56.7	*68.1*	*121*	*276*	*1035*
	80	340	347	351	353	347
		206	*271*	*453*	*1002*	*3813*

(P1) $\beta = 10^{-4}$		h				
		2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
N_t	5	41	52	60	66	71
		1.45	*2.41*	*4.81*	*12.0*	*48.1*
	10	41	51	59	65	70
		3.10	*4.82*	*9.60*	*24.3*	*95.6*
	20	40	49	57	64	69
		5.71	*9.77*	*18.7*	*48.6*	*191*
	40	39	46	55	63	68
		11.9	*18.5*	*36.9*	*93.1*	*377*
	80	46	49	56	65	69
		28.1	*39.4*	*72.3*	*185*	*745*

(P1) $\beta = 10^{-6}$		h				
		2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
N_t	5	29	34	39	50	58
		0.533	*0.845*	*2.83*	*11.5*	*67.8*
	10	29	34	39	51	58
		1.07	*1.66*	*5.60*	*23.3*	*137*
	20	28	33	39	50	58
		2.05	*3.18*	*11.1*	*46.1*	*280*
	40	28	33	37	50	56
		4.04	*6.13*	*21.1*	*96.2*	*550*
	80	28	31	37	49	55
		8.00	*11.6*	*42.2*	*181*	*1066*

Table 2: Number of iterations and CPU times in seconds (emphasized) when applying MINRES, with our block diagonal preconditioner, to the first example for Problem **(P1)**. Results are given for a variety of h and N_t, for $\beta = 10^{-2}$, $\beta = 10^{-4}$ and $\beta = 10^{-6}$, and with $\gamma = 1$.

© 2016. This manuscript version is made available under the CC-BY-NC-ND 4.0 license
http://creativecommons.org/licenses/by-nc-nd/4.0/

(P1)		$\beta = 10^{-2}$				$\beta = 10^{-4}$			
		$h = 2^{-2}$	$h = 2^{-3}$	$h = 2^{-4}$	$h = 2^{-5}$	$h = 2^{-2}$	$h = 2^{-3}$	$h = 2^{-4}$	$h = 2^{-5}$
N_t	5	32	41	47	46	20	33	37	36
	10	39	44	47	46	22	32	37	36
	20	62	67	65	62	23	30	37	36
	40	129	132	128	123	26	32	37	36

Table 3: Number of iterations when applying Minres, with our block diagonal preconditioner, to the second example for Problem **(P1)**. Results are given for a variety of h and N_t, for $\beta = 10^{-2}$ and $\beta = 10^{-4}$. The value of γ is varied, and the same results are observed for $\gamma = 1$, $\gamma = \tau^{-2}$, and $\gamma = \beta^{-1/2}\tau^{-1}$.

the Chebyshev semi-iteration method [31]. For the algebraic multigrid routine to approximate matrices of the form $\left(\tau^{-1} + \beta^{-1/2}\right)\mathbf{M} + \mathbf{K}$ or K_p, we apply 2 V-cycles of the HSL code `HSL_MI20` [2] for $\beta \geq 10^{-4}$, and the `AGMG` software [14, 15, 16, 17] for $\beta < 10^{-4}$ (as we find that `AGMG` works especially well when applied to the matrix $\left(\tau^{-1} + \beta^{-1/2}\right)\mathbf{M} + \mathbf{K}$ for small β). All experiments are carried out on a quad-core 1.6 GHz processor.

We experiment with a range of values of mesh-size h, time-steps N_t and values of β – in particular altering the values of h and N_t changes the dimension of the matrix system. In Table 1 we present the dimensions of the systems that we carry out our tests on, all of which are solved to the required tolerance using our approach. In bold are the dimensions of the systems which cannot be solved using a direct method: the large number of such systems demonstrates the value and importance of developing effective iterative methods for these problems.

We initially consider the solution of the first (driven cavity) example above, in the form of Problem **(P1)**. In Table 2 we present, for a range of h, N_t and β (and with $\gamma = 1$), the number of Minres iterations and CPU time taken to solve the problem to a tolerance of 10^{-5}. We find that for smaller values of β the problem is solved in very few iterations considering the high complexity of the problem. Furthermore the iteration count exhibits only very benign dependence on problem dimension (as altered by changing h and N_t), meaning that we are able to solve problems of very high dimension. Given that we believe that our solver may be fully parallelized in time, this is a favourable property of our method. For larger values of β we observe that the increase in iteration numbers is no longer benign – in fact the count can increase by a factor of up to 2 as the problem dimension is increased by the same factor (when doubling N_t). Although these results are much less favourable, we note that the parallelizability of our method should mitigate this increase in iteration numbers, and leave the robustness in h unaffected. Moreover the fact that our solver is much more effective for the (often more physically realistic) cases of decreasing β, this is again a positive property of our method.

In Table 3 we apply our strategy to the second (recirculating wind) example of the form **(P1)**, and also verify the performance as γ is altered (which one may wish to do to improve the conditioning of the system, for instance). We tested our approach for $\beta = 10^{-2}$ and $\beta = 10^{-4}$, with the three natural values of γ motivated in Section 2 (that is $\gamma = 1$, $\gamma = \tau^{-2}$, and $\gamma = \beta^{-1/2}\tau^{-1}$). We observe from the results that our solver is completely robust with respect to γ, as might be expected as our preconditioner involves scaling the blocks with suitable factors of γ to handle changes in the parameter. Furthermore the solver exhibits a similar benign dependence on h and N_t, and hence problem dimension, as for the first example.

We also test our method on a problem of the form **(P2)** (with periodic boundary conditions). The problem is in the form of the first (driven cavity) example stated above. Results are presented in Table 4 for a range of h and N_t, for $\beta = 10^{-3}$ and $\beta = 10^{-5}$. As before we observe mild dependence of the iteration numbers on h and N_t, with improved performance as β decreases. As with other tests, we observe that the CPU time scales almost linearly with problem dimension, apart from the case of large β and increasing N_t. This is a property which could be further enhanced by exploiting the parallelizability of our preconditioner.

It is important to observe that the stopping criterion used for our numerical tests up to this point,

© 2016. This manuscript version is made available under the CC-BY-NC-ND 4.0 license
http://creativecommons.org/licenses/by-nc-nd/4.0/

(P2) $\beta = 10^{-3}$		h: 2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
N_t	5	44 *0.695*	57 *1.55*	66 *3.85*	72 *12.5*	78 *54.2*
	10	42 *1.34*	55 *2.68*	64 *7.29*	70 *24.4*	76 *105*
	20	42 *4.45*	55 *5.28*	64 *15.0*	70 *48.4*	76 *208*
	40	47 *10.4*	58 *12.5*	66 *30.9*	77 *106*	83 *456*
	80	61 *25.6*	80 *32.6*	90 *83.7*	99 *270*	103 *1139*

(P2) $\beta = 10^{-5}$		h: 2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
N_t	5	34 *0.629*	41 *0.822*	53 *3.28*	60 *17.1*	64 *78.8*
	10	33 *1.21*	41 *1.59*	51 *6.30*	59 *33.1*	62 *154*
	20	33 *2.40*	40 *3.06*	50 *12.2*	57 *64.1*	62 *320*
	40	31 *4.45*	38 *5.76*	48 *30.6*	54 *120*	61 *617*
	80	29 *8.46*	35 *10.6*	44 *55.4*	52 *226*	59 *1227*

Table 4: Number of iterations and CPU times in seconds (emphasized) when applying MINRES, with our block diagonal preconditioner, to the first example for Problem **(P2)** (with periodic boundary conditions). Results are given for a variety of h and N_t, for $\beta = 10^{-3}$ and $\beta = 10^{-5}$, and with $\gamma = 1$.

(P2)		$\beta = 10^{-3}$				$\beta = 10^{-5}$			
		$h = 2^{-2}$	$h = 2^{-3}$	$h = 2^{-4}$	$h = 2^{-5}$	$h = 2^{-2}$	$h = 2^{-3}$	$h = 2^{-4}$	$h = 2^{-5}$
N_t	5	34	45	49	64	26	32	43	51
	10	44	58	68	76	35	42	54	64
	20	48	62	75	—	39	45	59	—
	40	53	67	79	—	43	50	62	—
	80	74	95	—	—	44	51	—	—

Table 5: Number of iterations when applying MINRES, with our block diagonal preconditioner, to the first example for Problem **(P2)**. Results are given for a variety of h and N_t, for $\beta = 10^{-3}$ and $\beta = 10^{-5}$, and with $\gamma = 1$. The iteration is terminated when the (scaled) vector 2-norm of the difference between the 'iterative' and 'direct' solutions for the control variable is less than 10^{-5}, for problems which lead to a direct solution being achieved.

© 2016. This manuscript version is made available under the CC-BY-NC-ND 4.0 license
http://creativecommons.org/licenses/by-nc-nd/4.0/

that is the reduction of the preconditioned residual norm by a factor of 10^{-5}, is itself influenced by the preconditioner used, and hence the values of β, h and N_t. Whereas this is a widely used method for measuring the success of an iterative method, we also wish to carry out a numerical test using a stopping condition that does not depend on the preconditioner itself. In Table 5 we again present numerical results for Problem **(P2)** (with periodic boundary conditions) – the stopping criterion is now taken to be a measure of how 'closely' the MINRES solution resembles that obtained using a direct method, for such problems where a direct solution can be obtained. In more detail, we allow the method to terminate when the vector 2-norm of the difference between the solutions obtained with direct and iterative methods, scaled by the 2-norm of the direct solution, falls below 10^{-5}. We observe that, for smaller values of N_t in particular, this is a less restrictive stopping condition than the reduction of the preconditioned residual norm. For larger problems, however, we observe a more pronounced increase in the iteration counts when using this new stopping condition. We conclude that considering a '$\mathcal{P}$-independent' stopping condition makes this an even more challenging problem, however we nonetheless still observe satisfying convergence using our method.

5. Concluding remarks

In this article we have investigated the fast iterative solution of large and sparse matrix systems of complex structure that arise from time-dependent Stokes control problems. Our preconditioning strategy involved rearranging the matrix system to suitable form, then deriving saddle point preconditioners for the matrices resulting from each time-step of the problem.

Our preconditioner exhibits several key advantages over direct methods for such problems. Firstly, the storage requirements are very small, as opposed to the huge provisions required for a direct method, and we are therefore able to solve problems of very large dimension. Secondly, the way we have constructed our preconditioner means that we believe our solver to be fully parallelizable, further expanding the potential of our approach. Further, numerical tests indicate that the performance of our method improves as the regularization parameter β decreases, and appears to be robust with respect to N_t (apart from in the case of large β) and h.

This work presents substantial scope for future research. For instance problems of boundary control form, or more complex flow problems such as time-dependent Navier-Stokes control formulations, could be investigated using the methodology introduced in this paper. It would also be desirable to ascertain whether an alternative strategy can be derived for the case of large β, which is the parameter regime within which our method performs least well at present. A further natural step would involve developing a fully parallel implementation of our solver, to verify its effectiveness over a large number of processors.

Acknowledgements. The author would like to thank two anonymous referees for their helpful comments. He also gratefully acknowledges useful conversations with Martin Stoll, Andy Wathen and Walter Zulehner about this topic. This work was partially supported by the Engineering and Physical Sciences Research Council (EPSRC) Fellowship EP/M018857/1.

References

[1] O. Axelsson, M. Neytcheva, Eigenvalue estimates for preconditioned saddle point matrices, Numer. Linear Alg. Appl., 13 (2006), 339–360.
[2] J. Boyle, M. D. Mihajlovic, J. A. Scott, HSL_MI20: an efficient AMG preconditioner for finite element problems in 3D, Int. J. Numer. Meth. Eng., 82 (2010), 64–98.
[3] J. Cahouet, J.-P. Chabard, Some fast 3D finite element solvers for the generalized Stokes problem, Int. J. Numer. Meth. Fl., 8 (1988), 869–895.
[4] H. C. Elman, A. Ramage, D. J. Silvester, Algorithm 866: IFISS, a MATLAB toolbox for modelling incompressible flow, ACM T. Math. Software, 33 (2007), 2–14.
[5] H. C. Elman, A. Ramage, D. J. Silvester, IFISS: a computational laboratory for investigating incompressible flow problems, SIAM Review, 56 (2014), 261–273.
[6] H. C. Elman, D. J. Silvester, A. J. Wathen, Finite Elements and Fast Iterative Solvers: With Applications in Incompressible Fluid Dynamics, Numerical Mathematics and Scientific Computation Series, Oxford University Press: New York, 2005.
[7] M. Hinze, M. Köster, S. Turek, A hierarchical space-time solver for distributed control of the Stokes equation, Priority Programme 1253, Preprint Number SPP1253-16-01, 2008.

© 2016. This manuscript version is made available under the CC-BY-NC-ND 4.0 license
http://creativecommons.org/licenses/by-nc-nd/4.0/

[8] I. C. F. Ipsen, A note on preconditioning non-symmetric matrices, SIAM J. Sci. Comput., 23 (2001), 1050–1051.
[9] D. Kay, D. Loghin, A. Wathen, A preconditioner for the steady-state Navier-Stokes equations, SIAM J. Sci. Comput., 24 (2002), 237–256.
[10] W. Krendl, V. Simoncini, W. Zulehner, Efficient preconditioning for an optimal control problem with the time-periodic Stokes equations, Lect. Notes Comput. Sci. Eng., 103 (2015), 479–487.
[11] W. Krendl, V. Simoncini, W. Zulehner, Stability estimates and structural spectral properties of saddle point problems, Numer. Math., 124 (2013), 183–213.
[12] Y. A. Kuznetsov, Efficient iterative solvers for elliptic finite element problems on nonmatching grids, Russ. J. Numer. Anal. M., 10 (1995), 187–211.
[13] M. F. Murphy, G. H. Golub, A. J. Wathen, A note on preconditioning for indefinite linear systems, SIAM J. Sci. Comput., 21 (2000), 1969–1972.
[14] A. Napov, Y. Notay, An algebraic multigrid method with guaranteed convergence rate, SIAM J. Sci. Comput., 34 (2012), A1079–A1109.
[15] Y. Notay, An aggregation-based algebraic multigrid method, Electron. Trans. Numer. Anal., 37 (2010), 123–146.
[16] Y. Notay, Aggregation-based algebraic multigrid for convection-diffusion equations, SIAM J. Sci. Comput., 34 (2012), A2288–A2316.
[17] Y. Notay, AGMG software and documentation; see `http://homepages.ulb.ac.be/∼ynotay/AGMG`.
[18] C. C. Paige, M. A. Saunders, Solutions of sparse indefinite systems of linear equations, SIAM J. Numer. Anal., 12 (1975), 617–629.
[19] J. Pearson, Fast Iterative Solvers for PDE-Constrained Optimization Problems, DPhil Thesis, University of Oxford, 2013.
[20] J. W. Pearson, On the role of commutator arguments in the development of parameter-robust preconditioners for Stokes control problems, Electron. Trans. Numer. Anal., 44 (2015), 53–72.
[21] J. W. Pearson, Preconditioned iterative methods for Navier-Stokes control problems, J. Comp. Phys., 292 (2015), 194–207.
[22] J. W. Pearson, M. Stoll, Fast iterative solution of reaction-diffusion control problems arising from chemical processes, SIAM J. Sci. Comput., 35 (2013), B987–B1009.
[23] J. W. Pearson, M. Stoll, A. J. Wathen, Regularization-robust preconditioners for time-dependent PDE-constrained optimization problems, SIAM J. Matrix Anal. Appl., 33 (2012), 1126–1152.
[24] J. W. Pearson, A. J. Wathen, Fast iterative solvers for convection-diffusion control problems, Electron. Trans. Numer. Anal., 40 (2013), 294–310.
[25] J. W. Pearson, A. J. Wathen, A new approximation of the Schur complement in preconditioners for PDE-constrained optimization, Numer. Linear Alg. Appl., 19 (2012), 816–829.
[26] D. Silvester, H. Elman, A. Ramage, Incompressible Flow and Iterative Solver Software (IFISS), version 3.3, `http://www.manchester.ac.uk/ifiss`, 2014.
[27] D. Silvester, A. Wathen, Fast iterative solution of stabilised Stokes systems. Part II: using general block preconditioners, SIAM J. Numer. Anal., 31 (1994), 1352–1367.
[28] M. Stoll, A. Wathen, All-at-once solution of time-dependent Stokes control, J. Comp. Phys., 232 (2013), 498–515.
[29] S. Takacs, A robust all-at-once multigrid method for the Stokes control problem, Numer. Math., 130 (2015), 517–540.
[30] F. Tröltzsch, Optimal Control of Partial Differential Equations: Theory, Methods and Applications, Graduate Series in Mathematics, American Mathematical Society, 2010.
[31] A. J. Wathen, T. Rees, Chebyshev semi-iteration in preconditioning for problems including the mass matrix, Electron. Trans. Numer. Anal., 34 (2009), 125–135.
[32] W. Zulehner, Nonstandard norms and robust estimates for saddle point problems, SIAM J. Matrix Anal. Appl., 32 (2011), 536–560.