A Hierarchical Space-Time Solver for Distributed Control of the Stokes Equation

Michael Hinze* Michael Köster†‡ Stefan Turek §

December 9, 2008

Abstract

We present a space-time hierarchical multigrid solution concept for optimisation problems governed by the time-dependent Stokes system. Discretisation is carried out with finite elements in space and a one-step θ-scheme in time. It is a key-feature of our multigrid solver that it shows a convergence behaviour which is independent of the degrees of freedom of the discrete problem, and that the solver performs robust with regard to the considered flow configuration. A set of numerical tests confirms this expectation and shows the efficiency of this approach for various problem settings.

1 Introduction

Active flow control plays a central role in many practical applications such as e.g. control of crystal growth processes [9, 14, 15], where the flow in the melt has a significant impact on the quality of the crystal. Optimal control of the flow by electro-magnetic fields and/or boundary temperatures leads to optimisation problems with PDE constraints, which are frequently governed by the time-dependent Navier-Stokes system. In the present work, we focus on the development of the hierarchical solution algorithm and therefore in the first instance consider flow governed by the time-dependent Stokes equation. The extension of the numerical solution concept to the fully nonlinear Navier-Stokes system is addressed in [18].

The structure of an optimisation problem with PDE constraints is condensated in the so called Karush-Kuhn-Tucker system (KKT), which describes the first order necessary optimality conditions of the underlying optimisation problem. It couples the state equation, which is the PDE to be controlled, with an adjoint equation and an optimality condition for the

*Department of Mathematics, University of Hamburg, Bundesstrasse 55, 20146 Hamburg, Germany, michael.hinze@uni-hamburg.de

†corresponding author

‡Institute of Applied Mathematics, Technische Universität Dortmund, Vogelpothsweg 87, D-44227 Dortmund, Germany, michael.koester@mathematik.tu-dortmund.de

§Institute of Applied Mathematics, Technische Universität Dortmund, Vogelpothsweg 87, D-44227 Dortmund, Germany, stefan.turek@mathematik.tu-dortmund.de

control input. The KKT system inherits a lot of structure and, as is shown in the present work, allows to develop numerical solution approaches which satisfy

$$\frac{\text{effort for optimisation}}{\text{effort for simulation}} \leq C, \tag{1.1}$$

with a constant $C > 0$ if moderate size. Here, the effort needed for the simulation of the PDE should be *optimal*, which typically means that the solver calculates an approximate solution with $O(N)$ operations, $N \in \mathbb{N}$ denoting the total number of unknowns for a given computational mesh – for a nonstationary simulation, in space and time. This can be achieved e.g. by implicit timestepping methods for the discretisation in time, multigrid methods for linear systems in space and Newton methods for treating nonlinearities. Requiring (1.1) then means that a solver for an optimal control problem also should have optimal complexity $O(N)$. As an example, in many practical applications adjoint-based steepest descent methods are used to solve optimisation problems, which in general do not satisfy this complexity requirement.

In the present work, we propose an hierarchical solution algorithm for distributed control of time-dependent Stokes flow with $O(N)$ complexity. It is based on a space-time multigrid approach applied to a space-time boundary value problem which is defined through the KKT system. First numerical results indicate that solving the KKT-system with this approach is about $C \approx 8-10$ times more expensive than the simulation. A related approach can be found, e.g. in [4] where multigrid methods for the numerical solution of optimal control problems for parabolic PDEs are developed based on Finite Difference techniques for the discretisation. In [6] a space-time multigrid method for Hackbusch's *integral equation approach* [10] is developed, compare also [7].

The paper is organised as follows: In Section 2, we describe the discretisation of the flow control problem and focus on the ingredients needed to design our multigrid solver. The discretisation is carried out with Finite Elements in space and Finite Differences in time. In Section 3, we propose the basic algorithms that are necessary to construct our multigrid solver. Finally, Section 4 is devoted to numerical examples based on the Stokes equation which we present to confirm the predicted behaviour.

An extension of our solution concept to flow governed by the nonstationary Navier-Stokes system is presented in [18].

2 Problem formulation and discretisation

We consider the optimal control problem

$$J(y,u) := \frac{1}{2}||y-z||^2_{L^2(Q)} + \frac{\alpha}{2}||u||^2_{L^2(Q)} + \frac{\gamma}{2}||y(T)-z(T)||^2_{L^2(\Omega)} \quad \longrightarrow \quad \min! \tag{2.1}$$

$$\begin{aligned} \text{s.t.} \qquad y_t - \nu\Delta y + \nabla p &= u \quad \text{in } Q, \\ -\text{div } y &= 0 \quad \text{in } Q, \\ y(0,\cdot) &= y^0 \quad \text{in } \Omega, \\ y &= g \quad \text{at } \Sigma, \end{aligned}$$

Here, $\Omega \subset \mathbb{R}^d$ $(d = 2, 3)$ denotes an open bounded domain, $\Gamma := \partial\Omega$, $T > 0$ defines the time horizon, and $Q = (0, T) \times \Omega$ denotes the corresponding space-time cylinder with space-time boundary $\Sigma := (0, T) \times \Gamma$. The function $g : \Sigma \to \mathbb{R}^d$ specifies some Dirichlet boundary conditions, u denotes the control, y the velocity vector, p the pressure and z a given target velocity field for y. Finally, $\gamma \geq 0$, $\alpha > 0$ denote constants. For simplicity, we do not assume any restrictions on the control u.

It is well known that problem (2.1) admits a unique solution (y, u), see e.g. [1, 8, 13], where also appropriate functional analytic settings for the optimisation problem can be found.

The first order necessary (and here also sufficient) optimality conditions for problem (2.1) are given by the so called *Karush-Kuhn-Tucker* system

$$\begin{aligned} y_t - \nu\Delta y + \nabla p &= u && \text{in } Q \\ -\text{div } y &= 0 && \text{in } Q \\ y(t,\cdot) &= g(t,\cdot) && \text{on } \Gamma \text{ for all } t \in [0,T] \\ y(0,\cdot) &= y^0 && \text{in } \Omega \\ \\ -\lambda_t - \nu\Delta\lambda + \nabla\xi &= y - z && \text{in } Q \\ -\text{div } \lambda &= 0 && \text{in } Q \\ \lambda(t,\cdot) &= 0 && \text{at } \Gamma \text{ for all } t \in [0,T] \\ \lambda(T) &= \gamma(y(T) - z(T)) && \text{in } \Omega \\ \\ u &= -\tfrac{1}{\alpha}\lambda, \end{aligned}$$

where λ denotes the dual velocity and ξ the dual pressure. Eliminating u in the KKT system yields (omitting boundary conditions at the moment)

$$\begin{aligned} y_t - \nu\Delta y + \nabla p &= -\frac{1}{\alpha}\lambda, \\ -\text{div } y &= 0, \\ y(0,\cdot) &= y^0, \end{aligned} \tag{2.2}$$

$$\begin{aligned} -\lambda_t - \nu\Delta\lambda + \nabla\xi &= y - z, \\ -\text{div } \lambda &= 0, \\ \lambda(T) &= \gamma(y(T) - z(T)), \end{aligned} \tag{2.3}$$

where we call (2.2) the *primal* and (2.3) the *dual* equation.

In the next step, we semi-discretise in time. As space-time systems are usually large, it is important to use a timestep as large as possible while guaranteeing accuracy and robustness.

This requires the usage of implicit time-stepping algorithms (cf. [21]) for stiff problems. Here, implicit schemes like the Crank-Nicolson or the Fractional-Step-θ scheme are favourable because they are of 2nd order accurate. However, for the sake of simplicity, we restrict to the standard 1st order backward Euler scheme as a representative of implicit schemes. Schemes of higher order and non-equidistant time stepping will be investigated in a forthcoming paper.

Using the implicit Euler scheme for the time discretisation of (2.2) yields

$$\begin{aligned} \frac{y_{n+1}-y_n}{\Delta t} - \nu\Delta y_{n+1} + \nabla p_{n+1} &= -\frac{1}{\alpha}\lambda_{n+1}, \\ -\text{div } y_{n+1} &= 0, \end{aligned} \tag{2.4}$$

$n = 0, ..., N-1$, where $N \in \mathbb{N}$ and $\Delta t := 1/N$. To the system (2.2), (2.3) we apply the discretisation recipe from [3]. For this purpose, we set $\mathcal{A} := -\nu\Delta$, $\mathcal{I} := \text{id}$, $\mathcal{G} := \nabla$, $\mathcal{D} := -\text{div}$. With $x := (y_0, p_0, y_1, p_1, ..., y_n, p_n)$ this yields

$$\mathcal{H}x := \left(\begin{array}{c|c|c|c} \frac{\mathcal{I}}{\Delta t}+\mathcal{A} \quad \mathcal{G} & & & \\ \mathcal{D} & & & \\ \hline -\frac{\mathcal{I}}{\Delta t} & \frac{\mathcal{I}}{\Delta t}+\mathcal{A} \quad \mathcal{G} & & \\ & \mathcal{D} & & \\ \hline & \ddots & \ddots \quad \ddots & \\ \hline & & -\frac{\mathcal{I}}{\Delta t} & \frac{\mathcal{I}}{\Delta t}+\mathcal{A} \quad \mathcal{G} \\ & & & \mathcal{D} \end{array}\right) \left(\begin{array}{c} y_0 \\ p_0 \\ \hline y_1 \\ p_1 \\ \hline \vdots \\ \hline y_n \\ p_n \end{array}\right) = \left(\begin{array}{c} (\frac{\mathcal{I}}{\Delta t}+\mathcal{A})y^0 \\ 0 \\ \hline -\frac{\lambda_1}{\alpha} \\ 0 \\ \hline \vdots \\ \hline -\frac{\lambda_n}{\alpha} \\ 0 \end{array}\right),$$

which is equivalent to (2.4) if y^0 is solenoidal. The time-integration scheme for the adjoint equation is determined by the adjoint $\mathcal{H}^*$ of $\mathcal{H}$ defined by

$$(\mathcal{H}x, \lambda) = (x, \mathcal{H}^*\lambda),$$

where $\lambda := (\lambda_0, \xi_0, \lambda_1, \xi_1, ..., \lambda_n, \xi_n)$. This yields

$$\mathcal{H}^*\lambda = \left(\begin{array}{c|c|c|c} \frac{\mathcal{I}}{\Delta t}+\mathcal{A} \quad \mathcal{G} & -\frac{\mathcal{I}}{\Delta t} & & \\ \mathcal{D} & & & \\ \hline & \frac{\mathcal{I}}{\Delta t}+\mathcal{A} \quad \mathcal{G} & -\frac{\mathcal{I}}{\Delta t} & \\ & \mathcal{D} & & \\ \hline & & \ddots \quad \ddots & \ddots \\ \hline & & & \frac{\mathcal{I}}{\Delta t}+\mathcal{A} \quad \mathcal{G} \\ & & & \mathcal{D} \end{array}\right) \left(\begin{array}{c} \lambda_0 \\ \xi_0 \\ \hline \lambda_1 \\ \xi_1 \\ \hline \vdots \\ \hline \lambda_n \\ \xi_n \end{array}\right) = \left(\begin{array}{c} 0 \\ 0 \\ \hline y_1 - z_1 \\ 0 \\ \hline \vdots \\ \hline (1+\frac{\gamma}{\Delta t})(y_n - z_n) \\ 0 \end{array}\right) \tag{2.5}$$

as the time discretisation scheme for the dual equation (2.3). Here we have used $\mathcal{D}^* = \mathcal{G}$ and $\mathcal{A}^* = \mathcal{A}$. Now let us define $w_i := (y_i,\ p_i,\ \lambda_i,\ \xi_i)$, $w := (w_0, w_1, ...) := (y_0, p_0, \lambda_0, \xi_0, y_1, p_1, \lambda_1, \xi_1, y_2, p_2, \lambda_2, \xi_2, ...)$. After shifting the terms with λ_{n+1} and y_n in (2.4) and (2.5)

from the right hand side to the left hand side and mixing the two matrices stemming from $\mathcal{H}$ and $\mathcal{H}^*$, we obtain a semi-discrete system

$$Gw = f. \tag{2.6}$$

Here,

$$G = \left(\begin{array}{c|c|c|c} \frac{\mathcal{I}}{\Delta t}+\mathcal{A} \quad \mathcal{G} & & & \\ \mathcal{D} & & & \\ \frac{\mathcal{I}}{\Delta t}+\mathcal{A} \quad \mathcal{G} & -\frac{\mathcal{I}}{\Delta t} & & \\ \mathcal{D} & & & \\ \hline -\frac{\mathcal{I}}{\Delta t} & \frac{\mathcal{I}}{\Delta t}+\mathcal{A} \quad \mathcal{G} \quad \frac{\mathcal{I}}{\alpha} & & \\ & \mathcal{D} & & \\ & -\mathcal{I} \quad \frac{\mathcal{I}}{\Delta t}+\mathcal{A} \quad \mathcal{G} & -\frac{\mathcal{I}}{\Delta t} & \\ & \mathcal{D} & & \\ \hline & \ddots & \ddots & \ddots \\ \hline & & -\frac{\mathcal{I}}{\Delta t} & \frac{\mathcal{I}}{\Delta t}+\mathcal{A} \quad \mathcal{G} \quad \frac{\mathcal{I}}{\alpha} \\ & & & \mathcal{D} \\ & & & -(1+\frac{\gamma}{\Delta t})\mathcal{I} \quad \frac{\mathcal{I}}{\Delta t}+\mathcal{A} \quad \mathcal{G} \\ & & & \mathcal{D} \end{array}\right)$$

and the right-hand-side of the system is given by

$$f = \Big(\underbrace{(\mathcal{I}/\Delta t + \mathcal{A})y^0, 0, 0, 0}, \underbrace{0, 0, -z_1, 0}, ..., \underbrace{0, 0, -z_{n-1}, 0}, \underbrace{0, 0, -(1+\gamma/\Delta t)z_n, 0} \Big).$$

At this point, we discretise in space with a Finite Element approach. The fully discrete version of the KKT system is defined by replacing the operators $\mathcal{I}$, $\mathcal{A}$, $\mathcal{D}$ and $\mathcal{G}$ by their Finite Element versions $\mathcal{A}^h$, $\mathcal{I}^h$, G^h and $\mathcal{D}^h$ and by incorporating boundary conditions into the right hand side f. We finally end up with the linear system

$$G^h w^h = f^h \tag{2.7}$$

with the vector $w^h := (w_0^h, w_1^h, ...)$ and $w_i^h := (y_i^h,\ p_i^h,\ \lambda_i^h,\ \xi_i^h)$. Note that G^h is a block tridiagonal matrix of the form

$$G^h = \left(\begin{array}{c|c|c|c} G_0 & \hat{M}_0 & & \\ \hline \tilde{M}_1 & G_1 & \hat{M}_1 & \\ \hline & \ddots & \ddots & \ddots \\ \hline & & \tilde{M}_N & G_N \end{array}\right)$$

where $N \in \mathbb{N}$ denotes the number of timesteps, and thus the solver for optimal control problem reduces to a solver for a sparse block tridiagonal system where the diagonal blocks G_n correspond to the timesteps of the fully coupled KKT system. This system does not have to be set up in memory in its complete form: Utilising defect correction algorithms reduces the

solution process to a sequence of matrix vector multiplications in space and time. A matrix-vector multiplication of a vector w^h with the space-time matrix G^h on the other hand reduces to N local matrix-vector multiplication sequences, one in each timestep with subsequent $\tilde{M}_n$, G_n and $\hat{M}_n$. So for solving (2.7), it is sufficient to perform matrix-vector multiplications in space with $\tilde{M}_n$, G_n and $\hat{M}_n$ – as long as it is possible to design correspondingly suitable space-time preconditioners to accelerate the iterative solution algorithm. This is explained in more detail in the next section.

3 The multigrid solver

The KKT-system represents a boundary value problem in the space-time cylinder. It is shown e.g. in [6] that, assuming sufficient regularity on the state (y, p) and the adjoint state (λ, ξ), it can equivalently be rewritten as higher-order elliptic equation in the space-time cylinder for either the state or the adjoint state. This indicates that multigrid could be used to solve the KKT system as it is an ideal solver for elliptic PDEs.

To formulate the multigrid solver, let Ω_1, ..., Ω_k for $k \in \mathbb{N}$ be a conformal hierarchy of triangulations of the domain Ω, with Ω_{i+1} stemming from a regular refinement of Ω_i (i.e. new vertices, cells and edges are generated by connecting opposite midpoints of edges). We use V_1, ..., V_k to refer to the different Finite Element spaces in space built upon these meshes. Furthermore, let T_1, ..., T_k be a hierarchy of decompositions of the time interval $[0, T]$, where each T_{i+1} stems from T_i by bisecting each time interval. For each i, the above discretisation in space and time yields a solution space $W_i = V_i \times T_i$ and a space-time system

$$G^i w^i = f^i, \qquad i = 1, ..., k$$

of the form (2.7) with $f^k = f^h$, $w^k = w^h$ and $G^k = G^h$ identifying the discrete right hand side, the solution and system operator on the finest level, respectively.

To describe the multigrid solver, we need prolongation and restriction operators. Let us denote by $I : W_i \to W_{i+1}$ the prolongation and by $R : W_i \to W_{i-1}$ the corresponding restriction. Furthermore, let $S : W_i \to W_i$ define a *smoothing* operator (see the following sections for a definiton of these operators) and let us denote with $NSMpre$, $NSMpost \in \mathbb{N}$ the numbers of pre- and postsmoothing steps, respectively.

This algorithm implements a basic multigrid V-cycle; for variations of this algorithm which use the W- or F-cycle, see [2, 11, 24].

3.1 Smoothing operators

The special matrix structure of the global space-time matrix (2.7) allows to define iterative smoothing operators for this algorithm. Note that every smoother usually can also be used as coarse grid solver to solve the equation $(G^k)^{-1} f$ in the first step of the algorithm by replacing the fixed number of iterations by a terminating condition depending on the residuum.

We introduce two basic iterative block smoothing algorithms. Let $\omega \in \mathbb{R}$ be a damping parameter. Then, the special matrix structure suggests the use of a Block-Jacobi method of the following form, see Algorithm 2.

Algorithm 1 Space-time multigrid

function SpaceTimeMultigrid(w;f;k)
  if ($k = 1$) **then**
    return $(G^k)^{-1} f$ ▷ coarse grid solver
  end if
  while (not converged) **do**
    $w \leftarrow S(G^k, w, f, NSMpre)$ ▷ presmoothing
    $d \leftarrow R(f - G^k w)$ ▷ restricion of the defect
    $w \leftarrow w + I(\text{SpaceTimeMultigrid}(0; d; k-1))$ ▷ coarse grid correction
    $w \leftarrow S(G^k, w, f, NSMpost)$ ▷ postsmoothing
  end while
  return w ▷ solution
end function

Algorithm 2 Space-time Block-Jacobi smoother

function JacSmoother(G^k,w,f,NSM)
  for $j = 0$ to NSM **do**
    $d \leftarrow f - G^k w$ ▷ Defect
    for $i = 0$ to N **do**
      $d_i \leftarrow (G_i^k)^{-1} d_i$ ▷ Block-Jacobi preconditioning
    end for
    $w \leftarrow w + \omega d$
  end for
  return w ▷ Solution
end function

Note that the key of the Block-Jacobi smoother lies in solving the system $G_i^k c_i = d_i$. This step means to solve the fully coupled KKT system in one time step and thus reduces the full space time algorithm to an algorithm in space; time coupling is controlled by the defect correction. One step of the Block-Jacobi algorithm therefore corresponds to one sweep through the whole time domain.

Similar to a Block-Jacobi algorithm, it is possible to design a forward-backward block SOR algorithm for smoothing, see Algorithm 3. (For the sake of notation, we define $x_{-1} := x_{N+1} := 0$, $\tilde{M}_0 := \hat{M}_N := 0$.) Again, let $\omega \in \mathbb{R}$ be a damping parameter; for $\omega = 1$ the algorithm reduces to a standard block Gauß-Seidel algorithm.

In contrast to Block-Jacobi, this algorithm respects the solutions backward and forward in time, so a stronger time coupling is reached without more additional costs, just some multiplications with a mass matrix. Like the Block-Jacobi method, this smoother reduces the whole space-time iteration to solving the spatial system $G_i^k c_i = d_i$ in each time step.

Note that our proposed multigrid is an iterative algorithm based on defect correction which, as we already mentioned, can be implemented without setting up the complete space-time matrix in memory. Both proposed smoothers reduce their space-time subproblems to a sequence of problems in space as well. The complete algorithm can therefore be implemented

Algorithm 3 Forward-Backward Block-SOR smoother

function FBSORSmoother(G^k,w,f,NSM)
  $r \leftarrow f - G^k w$ ▷ Defect
  $x \leftarrow 0$ ▷ Initial correction vector
  for istep = 1 to NSM **do**
    $x^{\text{old}} \leftarrow x$
    for $i = N$ downto 0 **do** ▷ Backward in time
      $d_i \leftarrow r_i - \tilde{M}_i x_{i-1}^{\text{old}} - G_i^k x_i^{\text{old}} - \hat{M}_i\big(\omega x_{i+1} + (1-\omega)x_{i+1}^{\text{old}}\big)$
      $x_i \leftarrow x_i^{\text{old}} + (G_i^k)^{-1} d_i$
    end for
    $x^{\text{old}} \leftarrow x$
    for $i = 0$ to N **do** ▷ Forward in time
      $d_i \leftarrow r_i - \tilde{M}_i\big(\omega x_{i-1} + (1-\omega)x_{i-1}^{\text{old}}\big) - G_i^k x_i^{\text{old}} - \hat{M}_i x_{i+1}^{\text{old}}$
      $x_i \leftarrow x_i^{\text{old}} + (G_i^k)^{-1} d_i$
    end for
  end for
  $w \leftarrow w + x$ ▷ Correction
  return w ▷ Solution
end function

without the necessity of setting up and saving the whole space time matrix in memory.

Solving a system of the form $G_i^k c_i = d_i$ in space looks more like a standard task. This system can be solved e.g. with direct solvers as long as the number of unknowns in space is not too large. For larger systems however, sophisticated techniques from computational fluid mechanics must be used. The system is a coupled saddle point problem for primal and dual velocity and pressure. Such problems can be solved by invoking a special multigrid method in space.

3.2 Coupled multigrid solvers in space

As mentioned above, in each time step a system of the form $G_i^k c_i = d_i$ must be solved, e.g. with a multigrid solver in space. Since the used prolongation and restriction operators based on the applied Finite Element spaces are standard and well known (see e.g. [2, 11, 24, 5, 21]), we restrict here to a short introduction into the pressure Schur complement ('PSC') approach for CFD problems (see also [20, 22, 23]) which is used as a smoother, acting simultaneously on the primal and dual variables. A complete overview and in-depth description of the operators and smoothers will be given in [17].

To formulate the corresponding algorithm, we first introduce some notations. Let $iel \in \mathbb{N}$ denote the number of an arbitrary element in the mesh. On this mesh, a linear system $Ax = b$

is to be solved; in our case, this system can be written in the form

$$\begin{pmatrix} A^{\text{primal}} & M^{\text{dual}} & B & 0 \\ M^{\text{primal}} & A^{\text{dual}} & 0 & B \\ B^T & 0 & 0 & 0 \\ 0 & B^T & 0 & 0 \end{pmatrix} \begin{pmatrix} y \\ \lambda \\ p \\ \xi \end{pmatrix} = \begin{pmatrix} b_y \\ b_\lambda \\ b_p \\ b_\xi \end{pmatrix}$$

which is a typical saddle point problem for primal and dual variables, with A^{primal}, A^{dual} velocity submatrices, M^{primal} and M^{dual} coupling matrices between the primal and dual velocity and B and B^T clustering the gradient/divergence matrices.

Now let $I(iel)$ identify a list of all degrees of freedom that can be found on element iel, containing numbers for the primal and dual velocity vectors in all spatial dimensions and the primal and dual pressure. With this index set, we define $A_{I(iel)}$ to be a (rectangular) matrix containing only those rows from A identified by the index set $I(iel)$. In the same way, let $x_{I(iel)}$ and $b_{I(iel)}$ define the subvectors of x and b containing only the entries identified by $I(iel)$. Furthermore we define $A_{I(iel),I(iel)}$ to be the (square) matrix that stems from extracting only those rows and columns from A identified by $I(iel)$.

Algorithm 4 PSC-Smoother for smoothing an approximate solution to $Ax = b$

function PSCSmoother(A,x,b,NSM)
  for ism = 1,NSM **do** ▷ NSM smoothing sweeps
    for iel = 1 to NEL **do** ▷ Loop over the elements
      $x_{I(iel)} \leftarrow x_{I(iel)} + \omega C_{iel}^{-1}(b_{I(iel)} - A_{I(iel)}x)$ ▷ Local Correction
    end for
  end for
  return x ▷ Solution
end function

This notation allows to formulate the basic PSC smoother in space, providing $\omega \in \mathbb{R}$ to be a damping parameter, see Algorithm 4. Of course, this formulation is not yet complete, as it is lacking a proper definition of the local preconditioner C_{iel}^{-1} which is a small square matrix with as many unknowns as indices in $I(iel)$.

There are two basic approaches for this preconditioner. The first approach, which we entitle by PSCSmootherFull, results in the simple choice of $C_{iel} := A_{I(iel),I(iel)}$ and calculating C_{iel}^{-1} by invoking a LU decomposition, e.g. with the LAPACK package [19]. That approach is rather robust and still feasible as the system is small; for the $\tilde{Q}_1/Q_0$ space that is used in our discretisation (see [21]), the system has 18 unknowns.

The second approach, which we call PSCSmootherDiag, results in taking a different subset of the matrix A for forming $C_{I(iel)}$. To describe this approach, we define

$$\hat{A} := \begin{pmatrix} \text{diag}(A^{\text{primal}}) & 0 & B & 0 \\ 0 & \text{diag}(A^{\text{dual}}) & 0 & B \\ B^T & 0 & 0 & 0 \\ 0 & B^T & 0 & 0 \end{pmatrix}$$

where *diag(·)* refers to the operator taking only the diagonal of a given matrix. The local preconditioner can then be formulated as $C_{iel} := \hat{A}_{I(iel),I(iel)}$. Note that this matrix decouples the primal variables from the dual variables, so that applying $\hat{A}^{-1}_{I(iel),I(iel)}$ decomposes into two independent subproblems for the primal variables (y, p) and the dual variables (λ, ξ). The special feature that the velocity blocks are diagonal allows to efficiently use Schur complement techniques by what this smoother is numerically cheaper than PSCSmootherFull. Nevertheless, the disadvantage of this smoother is the reduced stability. We note that the PSC approach in general also allows to increase the stability by taking larger local systems. Such an approach was carried out e.g. in [20] where the degrees of freedom of multiple adjacent elements were clustered together to form a linear subsystem. For our numerical tests however, such an approach was not necessary. Most of the numerical tests in the later sections were carried out using PSCSmootherDiag except where noted.

3.3 Prolongation/Restriction

Our discretisation is based on Finite Differences in time and Finite Elements in space. The operators for exchanging solutions and right hand side vectors between the different levels therefore decompose into a time prolongation/restriction [12] and space prolongation/restriction. Let $k \in \mathbb{N}$ be the space level. Then, we denote by $I_S : V_k \to V_{k+1}$ the prolongation in space and $R_S : V_{k+1} \to V_k$ the corresponding restriction. The prolongation for a space-time vector $w^k = (w_0^k, ..., w_N^k)$ can be written as:

$$P(w^k) := \left(P_S(w_0^k),\ \frac{P_S(w_0^k) + P_S(w_1^k)}{2},\ P_S(w_1^k),\ \frac{P_S(w_1^k) + P_S(w_2^k)}{2},\ ...,\ P_S(w_N^k) \right)$$

and is a composition of the usual Finite Difference prolongation in time and Finite Element prolongation in space. The corresponding restriction for a defect vector $d^k = (d_0^k, ..., d_{2N}^k)$ follows directly:

$$R(d^k) := \left(R_S(\frac{1}{4}(2d_0^k + d_1^k)),\ R_S(\frac{1}{4}(d_1^k + 2d_2^k + d_3^k)),\ ...,\ R_S(\frac{1}{4}(d_{2N-1}^k + 2d_{2N}^k)) \right)$$

Our numerical tests in Section 4 are carried out with the nonconforming $\tilde{Q}_1/Q_0$ Finite Element pair in space. For these elements, we use the standard prolongation/restriction operators which can be found e.g. in [16, 21].

4 Numerical examples

In this section we numerically analyse the proposed multigrid method. We start with examining the dependence of the convergence behaviour on the two smoothers suggested above.

Basic tests: Convergence properties of the solver

For our investigations, we set up a test example with an analytical solution. As domain, consider a unit square $\Omega = [0, 1]^2$ in $\mathbb{R}^2$ on the time domain $[0, T]$ with $T = 1$. The optimal

control problem reads:

$$J(y,u) := \frac{1}{2}||y-z||^2_{L^2(Q)} + \frac{\alpha}{2}||u||^2_{L^2(Q)} + \frac{\gamma}{2}||y(T)-z(T)||^2_{L^2(\Omega)} \quad \longrightarrow \quad \text{min!} \tag{4.1}$$

$$\begin{aligned} \text{s.t.} \qquad y_t - \nu\Delta y + \nabla p &= f+u \quad \text{in } \Omega \\ -\text{div } y &= 0 \quad \text{in } \Omega \\ y(0,\cdot) &= y^0 \quad \text{in } \Omega \\ y(\cdot,0) &= 0 \quad \text{at } \Gamma := \partial\Omega \end{aligned}$$

with a slight modification in the right hand side of the Stokes equation that allows us to set up an analytical solution. The corresponding KKT system reads after the elimination of u:

$$\begin{aligned} y_t - \nu\Delta y + \nabla p + \tfrac{1}{\alpha}\lambda &= f && \text{in } \Omega \\ -\text{div } y &= 0 && \text{in } \Omega \\ y(t,\cdot) &= g(t,\cdot) && \text{at } \Gamma \text{ for all } t\in[0,T] \\ y(0,\cdot) &= 0 && \text{in } \Omega \\ \\ -\lambda_t - \nu\Delta\lambda + \nabla\xi - y &= -z && \text{in } \Omega \\ -\text{div } \lambda &= 0 && \text{in } \Omega \\ \lambda(t,\cdot) &= 0 && \text{at } \Gamma \text{ for all } t\in[0,T] \\ \lambda(T) - \gamma y(T) &= -\gamma z(T) && \text{in } \Omega \end{aligned}$$

We choose $\gamma = 0$, $\alpha = 0.01$, $\nu = 1$. As analytical solution, we choose for the primal velocity $\bar{y} := \bar{y}(x,y,t) := (y_1,y_2)\cdot s(t)$ with $y_1 := y_1(x_1,x_2,t) := \sin^2(\pi x_1)\sin(\pi x_2)\cos(\pi x_2)$, $y_2 := y_2(x_1,x_2,t) := -\sin^2(\pi x_2)\sin(\pi x_1)\cos(\pi x_1)$ and $\bar{p} := \bar{p}(x_1,x_2,t) := \sin(2\pi x_1)\sin(2\pi x_2)s(t)$. Here we use $s(t) := 1-4(t-\frac{1}{2})^2$. As dual velocity/pressure, we use the same as the primal, i.e. $\bar{\lambda} := \bar{y}$, $\bar{\xi} := \bar{p}$. This choice of primal and dual solution leads to a right hand side $f = \bar{y}_t - \Delta\bar{y} + \nabla\bar{p} + 100\bar{\lambda}$ and a target flow $z = \bar{\lambda}_t + \bar{y} + \Delta\bar{\lambda} - \nabla\bar{\xi}$. Figure 4.1 depicts y_1 at time $t = 0.5$.

Δt	1/4	1/8	1/16	1/32	1/64
h	1/4	1/8	1/16	1/32	1/64
$\|\|y-\bar{y}\|\|_{L^2(Q)}$	2.69E-02	1.16E-02	6.14E-03	3.34E-03	1.76E-03
$\|\|p-\bar{p}\|\|_{L^2(Q)}$	2.08E-01	1.16E-01	5.90E-02	3.00E-02	1.51E-02
$\|\|\lambda-\bar{\lambda}\|\|_{L^2(Q)}$	2.49E-02	9.12E-03	4.61E-03	2.62E-03	1.43E-03
$\|\|\xi-\bar{\xi}\|_{L^2(Q)}$	1.98E-01	1.11E-01	5.79E-02	2.95E-02	1.49E-02

Table 4.1: L^2 error reduction in the different velocity/pressure components upon refinement in space and time, for the test problem with the analytical solution.

The next step is to set up a mesh in space and time. We choose as coarse grid the once refined unit cube in space-time, i.e. $\Delta t = h = 1/2$. The space-time multigrid approach solves the system up to a relative residual of $\varepsilon_{\text{OptMG}} = 10^{-10}$, the coarse grid solver solves

Δt	1/4	1/16	1/64	1/256
h	1/4	1/8	1/16	1/32
$\|\|y - \bar{y}\|\|_{L^2(Q)}$	2.69E-02	8.68E-03	2.41E-03	6.16E-04
$\|\|p - \bar{p}\|\|_{L^2(Q)}$	2.08E-01	1.18E-01	5.91E-02	2.94E-02
$\|\|\lambda - \bar{\lambda}\|\|_{L^2(Q)}$	2.49E-02	7.77E-03	2.19E-03	5.52E-04
$\|\|\xi - \bar{\xi}\|_{L^2(Q)}$	1.98E-01	1.14E-01	5.83E-02	2.92E-02

Table 4.2: L^2 error reduction in the different velocity/pressure components with 2 refinement steps in time upon one refinement step in space, for the test problem with the analytical solution.

exactly. As smoother, we use only postsmoothing with JacSmoother($\omega = 0.7$,NSM=2) and FBSORSmoother($\omega = 1.0$,NSM=1). With this setting, the effort for one multigrid iteration is roughly the same for both types of smoothers.

The discretisation in space is carried out with the nonconforming Rannacher-Turek element $\tilde{Q}_1$ for the velocity and the piecewise constant element Q_0 for the pressure. Discretisation and the solution of the linear system $G_i^k c_i = d_i$ in each timestep is realised with the described modified flow solver based on the FeatFlow package (`http://www.featflow.de`). A multigrid solver in space solves the system up to a relative residual of $\varepsilon_{\text{SpaceMG}} = 10^{-5}$ using a BiCGStab smoother with PSC preconditioner in accordance to the algorithm described in section 3.2. We note here that the solver behaviour is relatively insensitive to the stopping criterion $\varepsilon_{\text{SpaceMG}}$.

Table 4.1 depicts the L^2-error over the space-time cylinder in the primal and dual velocity

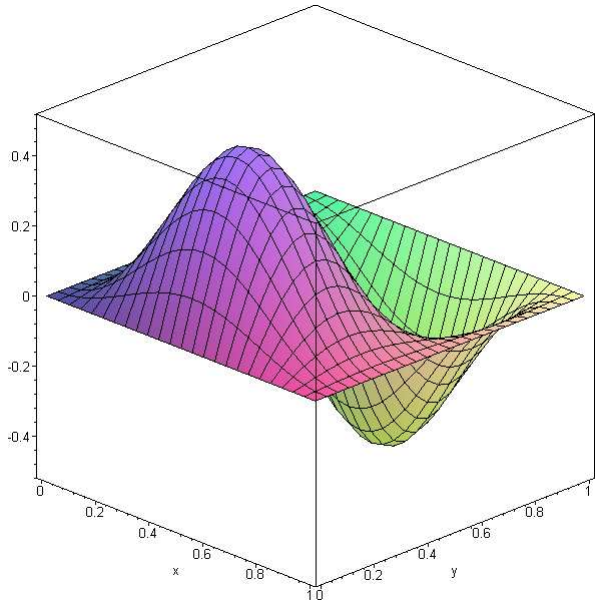

Figure 4.1: The analytical test solution y_1 at time $t = 0.5$.

			JACSmoother		FBSORSmoother	
Δt	h	#DOF	#ite	ρ	#ite	ρ
1/4	1/4	960	10	9.18E-02	3	4.76E-05
1/8	1/8	6336	10	9.27E-02	3	1.03E-04
1/16	1/16	45696	10	8.65E-02	3	1.04E-04
1/32	1/32	346368	10	8.46E-02	3	2.91E-04
1/64	1/64	3244800	10	9.64E-02	3	2.93E-04
1/4	1/8	2112	10	8.65E-02	3	3.89E-05
1/8	1/16	24192	10	8.51E-02	3	1.04E-04
1/16	1/32	178432	10	8.42E-02	3	2.05E-04
1/32	1/64	1647360	10	8.40E-02	3	2.91E-04
1/4	1/16	8064	10	8.45E-02	3	3.91E-05
1/8	1/32	94464	10	8.41E-02	3	1.04E-04
1/16	1/64	848640	10	8.39E-02	3	2.04E-04
1/32	1/128	5440512	10	8.48E-02	3	2.96E-04
1/4	1/32	31488	10	8.37E-02	3	4.13E-05
1/8	1/64	449280	10	8.38E-02	3	1.04E-04
1/16	1/128	2802688	10	8.47E-02	3	2.04E-04

Table 4.3: Convergence rates of the time and the space-time multigrid for the two smoothers. *#DOF* indicates the number of degrees of freedom of the complete space-time system.

as well as in the primal and dual pressure upon refinement of the space-time mesh. The error of all these four solution components reduce nicely with a factor of at least 2 on simultaneous decreasing of h and Δt; a higher factor for the error reduction in the velocity cannot be expected due to the first order discretisation in time. This behaviour changes in Table 4.2 where the time resolution is twice increased per refinement in space. Here, the L^2 error reduction factor of 4 for the velocity and 2 for the pressure can be seen, which is the expected behaviour for the Stokes equation.

We continue with the numerical analysis of the multigrid solver itself. Table 4.3 lists the number of iterations as well as the convergence rate of the multigrid solver for different settings for Δt and h on the finest space-time mesh.

The convergence rates are obviously independent of the space-time level and do not deteriorate if Δt is much larger than the space resolution h. This behaviour is of course highly favourable, as higher order implicit timestepping methods are planned to be used in future to reduce the number of timesteps. The forward-backward block SOR smoother shows a much better convergence behaviour than the Block-Jacobi method as expected from the additional time coupling.

Numerical effort: Optimisation vs. Simulation

As already indicated in the introduction, a key point in the design of solvers for optimal control problems is the numerical effort of the algorithm in comparison to a 'similar' simulation:

Starting from an efficient flow solver that solves a CFD problem in reasonable time, an optimisation algorithm using these flow solver techniques is intended to do the optimisation with a work amount which is a (preferably small) multiple of the costs that were necessary for one simulation. It is important that this factor is independent of the problem size, thus making the optimisation approach feasible also for large problems.

To illustrate the amount of additional effort of the optimisation algorithm in comparison to a simulation, we define the following test configurations:

1.) We solve a distributed control problem for the Stokes equation up to a relative residual of $\varepsilon_{\text{OptMG}} = 10^{-10}$. In each timestep of the smoother, the linear system in space is solved to a relative residuum of $\varepsilon_{\text{SpaceMG}} = 10^{-5}$. The settings utilised here reflect the configuration that was used to calculate the values in Table 4.3.

2.) We solve a Stokes flow of the form

$$\begin{aligned} y_t - \Delta y + \nabla p &= f \quad \text{in } Q \\ -\text{div } y &= 0 \quad \text{in } Q \\ y &= 0 \quad \text{at } \Sigma \\ y(\cdot, 0) &= y^0 := 0 \quad \text{in } \Omega \end{aligned} \tag{4.2}$$

with f being the same as in the case of the optimisation. Each timestep was solved up to a relative residuum of $\varepsilon_{\text{SpaceMG}} = 10^{-10}$ with a multigrid solver in space using 4 postsmoothing steps of a 'diagonal' PSC smoother.

Based on this configuration, we now define the *performance measure*

$$\mu := \frac{\text{time for the optimisation solver}}{\text{time for the simulation solver}}$$

and use this to compare the execution time, see Table 4.4. Using the Block-Jacobi smoother, the optimisation algorithm is about $20-25$ times more expensive, while the forward-backward block SOR algorithm reduces the costs to about $7-9$ times the costs of the simulation, independent of the problem size. Of course, the value of the performance measure is largely influenced by the setting of the linear solver in each time step of the optimisation algorithm. Another reasonable setting would be to reduce the stopping criterion of the spatial solver in each time step to $\varepsilon_{\text{SpaceMG}} = 10^{-1}$ instead of $\varepsilon_{\text{SpaceMG}} = 10^{-5}$, so the 'inner' solvers are intended to gain one digit per iteration, while the 'outer' solver solves for 10 digits. As can be seen in the Table, this modification is indeed possible and reduces e.g. the factor μ for the FBSORSmoother on reasonable refinement levels by its half. One can also see that this modification has only a minor impact to the global space-time multigrid solver: The number of iterations stays nearly constant.

Flow Control for a Driven Cavity configuration

We proceed with an optimal control problem in a nonstationary case without a given analytical solution. For that purpose, we set up a destination flow by using the Driven Cavity problem:

		$\varepsilon_{\text{SpaceMG}} = 10^{-5}$				$\varepsilon_{\text{SpaceMG}} = 10^{-1}$			
		JacSmoother		FBSORSm.		JacSmoother		FBSORSm.	
Δt	h	$\#\text{ite}_{\text{OptMG}}$	μ	$\#\text{ite}_{\text{OptMG}}$	μ	$\#\text{ite}_{\text{OptMG}}$	μ	$\#\text{ite}_{\text{OptMG}}$	μ
1/4	1/4	10	16	3	7	10	19	3	8
1/8	1/8	10	23	3	9	10	23	4	10
1/16	1/16	10	22	3	9	10	20	4	7
1/32	1/32	10	20	3	8	10	10	4	5
1/64	1/64	10	25	3	8	10	9	4	4

Table 4.4: Number of iterations of the space-time multigrid and quotient of the CPU time of the optimisation algorithm against a simulation for the different smoothers.

For $(x_1, x_2) \in \Gamma$, we define $y(x_1, 1, t) := (1, 0)$ and $y(x_1, x_2, t) := (0, 0)$ for all $(x_1, x_2) \in \Gamma$ with $x_2 \neq 1$. By simulating the flow for $t \in [0, 1]$ using the nonstationary Stokes equation (4.2) with $f := 0$ and $\nu = 1$, we generate a flow z which is to be used as target flow in the above optimisation setting.

The optimal control problem aims now at simultaneously tracking this flow and reducing fluctuations introduced by boundary conditions. We set the right hand side $f := 0$ and $y(x_1, x_2, t) := (0, 0)$ for all $(x_1, x_2) \in \Gamma$ with $x_2 \neq 1$. The boundary condition at $x_2 = 1$ is now defined by $y(x_1, 1, t) := (1 + \frac{1}{2}\cos(4\pi t - \pi), 0)$ which introduces a fluctuation. The calculation itself is carried out with a viscosity of $\nu = 1/100$.

		$\varepsilon_{\text{SpaceMG}} = 10^{-10}$		$\varepsilon_{\text{SpaceMG}} = 10^{-5}$		$\varepsilon_{\text{SpaceMG}} = 10^{-2}$		$\varepsilon_{\text{SpaceMG}} = 10^{-1}$	
Δt	h	#MG	time	#MG	time	#MG	time	#MG	time
1/4	1/4	10	6.1	10	5.4	10	4.9	10	4.9
1/8	1/8	12	24.2	12	17.5	12	14.5	12	14.4
1/16	1/16	11	102.5	11	58.5	11	37.1	11	37.3
1/32	1/32	8	528.8	8	219.7	8	111.8	8	113.1
1/64	1/64	7	2914.5	7	1241.0	7	659.0	7	667.8

Table 4.5: Number of iterations and CPU time for different space and time resolutions and different stopping criteria of the multigrid in space; Driven cavity configuration.

Figure 4.2 to 4.4 illustrate the target flow as well as the uncontrolled and the controlled flow at $t = 0.5$ and $t = 0.75$. Figure 4.4 visualises the reference flow. From Figure 4.3 one can see, that the optimiser successfully calculated a controlled flow that shows only minor visual difference to the reference flow.

Table 4.5 and 4.6 show the convergence rates and number of iterations for different mesh sizes in space and time as well as for different stopping criteria of the inner linear solver. The smoother used in these tests was FBSORSmoother with 1 postsmoothing step, damped by $\omega = 0.9$. From Table 4.5 one can see that the convergence of the global solver barely depends on the convergence criterion of the linear solver in space. The solver always converges with the

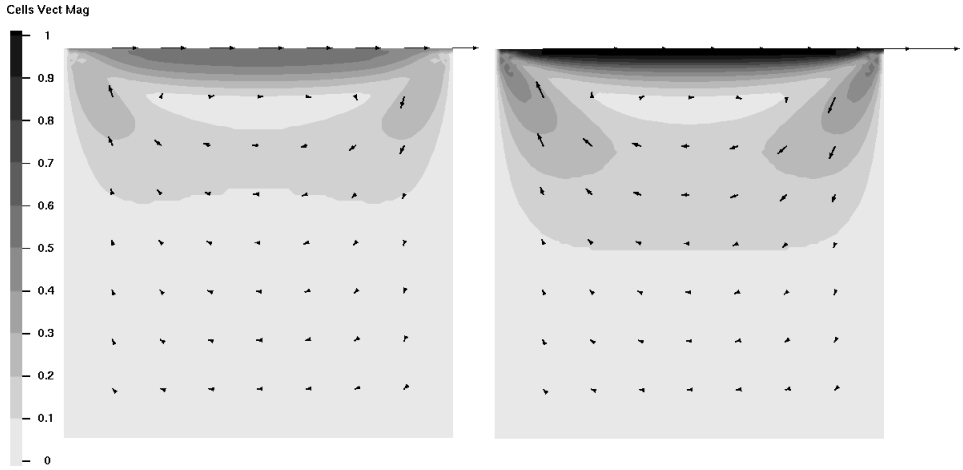

Figure 4.2: Uncontrolled Stokes flow at $t = 0.5$ (left) and $t = 0.75$ (right)

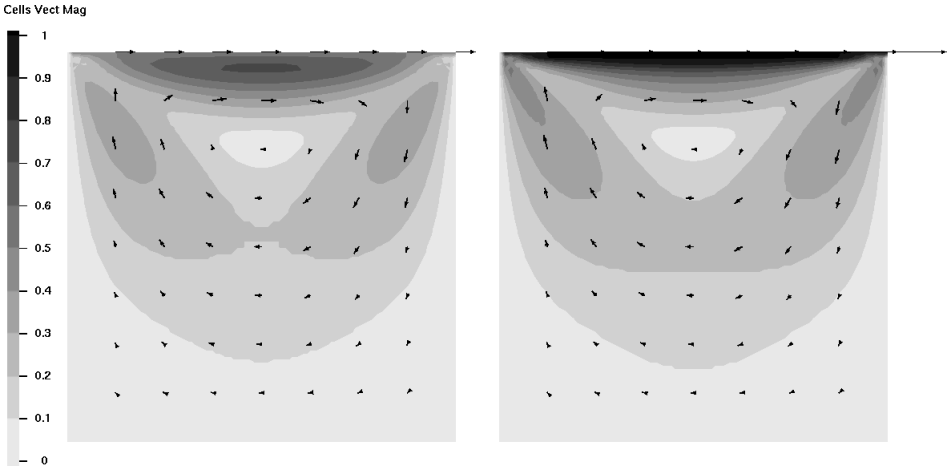

Figure 4.3: Controlled Stokes flow at $t = 0.5$ (left) and $t = 0.75$ (right)

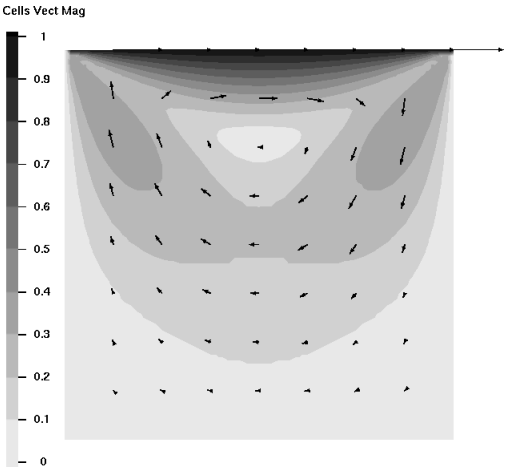

Figure 4.4: Target Stokes flow at $t = 0.5$.

		FBSORSmoother ($\omega = 0.9$)		JacSmoother ($\omega = 0.7$)	
Δt	h	#ite	ρ	#ite	ρ
1/4	1/4	2	2.46E-06	23	3.54E-01
1/8	1/8	12	1.45E-01	21	3.30E-01
1/16	1/16	11	1.10E-01	21	3.28E-01
1/32	1/32	8	5.01E-02	22	3.48E-01
1/64	1/64	7	3.05E-02	23	3.60E-01
1/4	1/8	9	7.58E-02	22	3.48E-01
1/8	1/16	12	1.33E-01	25	3.85E-01
1/16	1/32	10	9.47E-02	21	3.32E-01
1/32	1/64	8	4.55E-02	22	3.42E-01
1/4	1/16	9	6.78E-02	22	3.42E-01
1/8	1/32	11	1.17E-01	22	3.50E-01
1/16	1/64	10	8.67E-02	20	3.14E-01
1/4	1/32	9	6.04E-02	21	3.22E-01
1/8	1/64	11	1.08E-01	21	3.30E-01

Table 4.6: Convergence rate and number of iterations for different space and time resolutions for the Driven cavity configuration.

same number of iterations[1], only the absolute CPU time is different. This clearly indicates the robustness of our space-time MG solver. In the following numerical tests we choose $\varepsilon_{\text{SpaceMG}} = 10^{-2}$.

Table 4.6 now depicts the convergence rates for FBSORSmoother($\omega = 0.9, \text{NSM} = 1$) JacSmoother($\omega = 0.7, \text{NSM} = 2$) – so one multigrid iteration with Block-Jacobi smoother is roughly as expensive as one iteration with our forward-backward block SOR smoother. Similar to the case of the test problem with the prescribed analytical solution above, the convergence rates remain constant when refining the mesh simultaneously in space and time, clearly visible when FBSORSmoother is used. The convergence rates are slightly worse than in the analytical test problem but still in most cases < 0.1. When reducing h, the convergence rates of the solver slowly reduce (except for very coarse grids), thus a finer spatial mesh leads to better convergence rates of the solver. This behaviour is independent of whether Δt is changed with h or kept fixed which was also observed in [6] for the case of the optimal control of a heat equation with a space-time multigrid solver.

Optimisation in a complex spatial domain

In the last test we want to exploit the flexibility of Finite Elements to apply optimal control onto a more complex problem of 'flow around cylinder' type. In this test case, our domain as

[1]We note that the CPU times of $\varepsilon_{\text{SpaceMG}} = 10^{-1}$ were a little bit higher than those of $\varepsilon_{\text{SpaceMG}} = 10^{-2}$ because of a slightly worse behaviour of the space-time coarse grid solver. This had no influence to the overall iteration.

defined as a rectangle without an inner cylinder: $\Omega = [0, 2.2] \times [0.0.41] \setminus B_r(0.2, 0.2)$, $r = 0.05$. On the left edge of the domain Γ_4 we induce an oscillating parabolic inflow profile with maximum velocity $U_{\max} = U_{\max}(t) = 0.3(1 + \sin\big((t-1)\pi/2\big)$. On the right edge Γ_2 we define natural boundary conditions and the boundary conditions at the circle are defined as no-slip. As problem, we consider again the Stokes problem with a viscosity parameter $\nu = 1/250$ and a time horizon of $[0, T] = [0, 10]$. The target flow z is generated by a simulation of the corresponding Navier–Stokes equation, also with $\nu = 1/250$. A proper optimisation will therefore 'emulate' the effect of the nonlinear term $y\nabla y$ with the right hand side and produce the same flow field as in the Navier–Stokes case.

		FBSORSmoother($\omega = 0.9$)			
Δt	space-lv.	#ite	ρ	μ	$\|\|y - z\|\|_{L^2(Q)}$
0.625	2	3	2.13E-02	11.84	1.19E-02
0.3125	3	4	3.43E-02	11.00	9.72E-03
0.15625	4	5	7.17E-02	10.70	6.57E-03
0.078125	5	6	9.32E-02	7.99	3.73E-03
0.625	3	4	2.51E-02	30.47	9.85E-03
0.3125	4	4	3.91E-02	20.39	6.65E-03
0.15625	5	5	6.71E-02	11.83	3.52E-03

Table 4.7: Convergence rate ρ and number of iterations *#ite* of the space-time multigrid as well as the space multigrid for the 'flow around cylinder' problem.

The discrete KKT system in this test is solved up to a relative residual of 10^{-5} utilising a V-cycle with 1 postsmoothing step with FBSORSmoother. The spatial solver in each time step is again a multigrid algorithm that solves up to a relative residual of 10^{-1} and uses a PSC like smoother as described above.

In Figure 4.5, we entitle the coarse grid by 'space level 1', each higher space level is generated by regular refinement. Figure 4.6 shows the target and uncontrolled flow, Figure 4.7 the controlled flow field which can hardly be distinguished from the destination flow field in Figure 4.6. Figure 4.8 finally visualises the control u; it can be seen that, as expected, the nonlinearity is most active around the object while there is no influence in the region far behind where the flow turns to Poiseuille flow. Table 4.7 depicts the results of the convergence of the multigrid solver for different levels of refinement in space and time. The full space-time multigrid solver works well for all refinement levels in space and time, the error $||y - z||_{L^2(Q)}$ to the target flow reduces with increasing level. The column entitled by μ again shows the additional effort of the optimisation in comparison to a simulation. For the simulation, we used in each timestep a multigrid solver with 4 smoothing steps of the 'diagonal' PSC smoother, reducing the residuum in each timestep by 10^{-5}. As can be seen, the additional effort of the optimisation is roughly $10 - 12$ times the effort for the simulation on reasonable refinement levels.

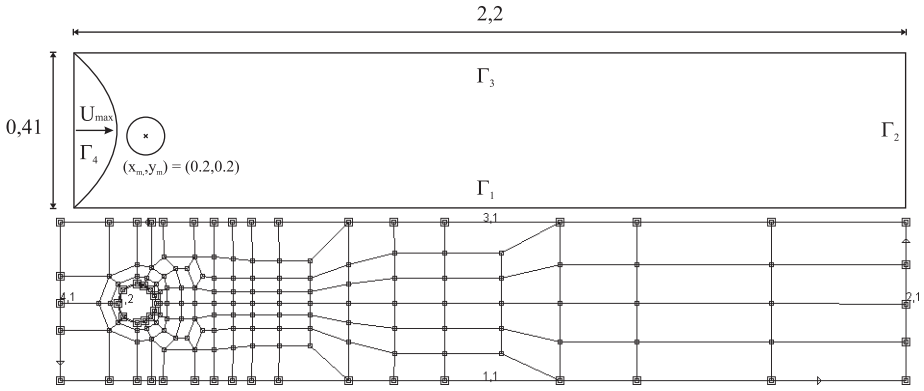

Figure 4.5: Domain and basic coarse mesh, Flow around Cylinder configuration

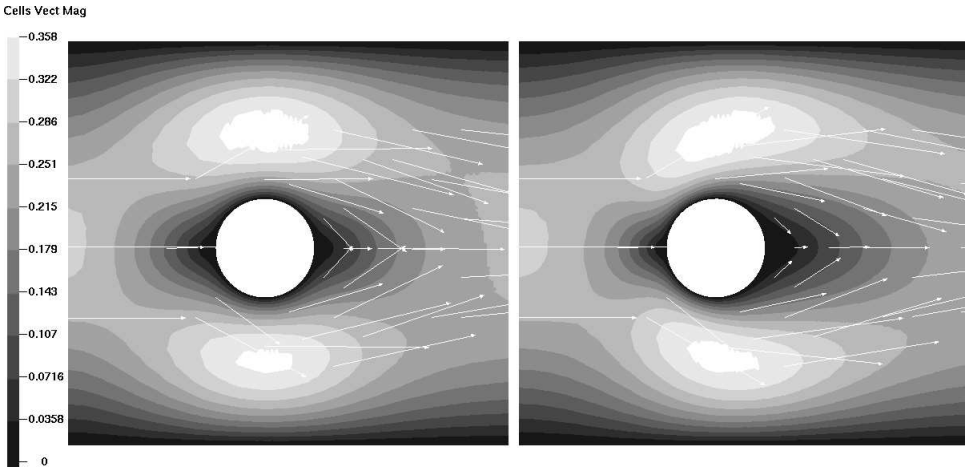

Figure 4.6: Velocity field of the (Navier–Stokes) target flow z (left) and the uncontrolled Stokes flow at $t = 10$ (right).

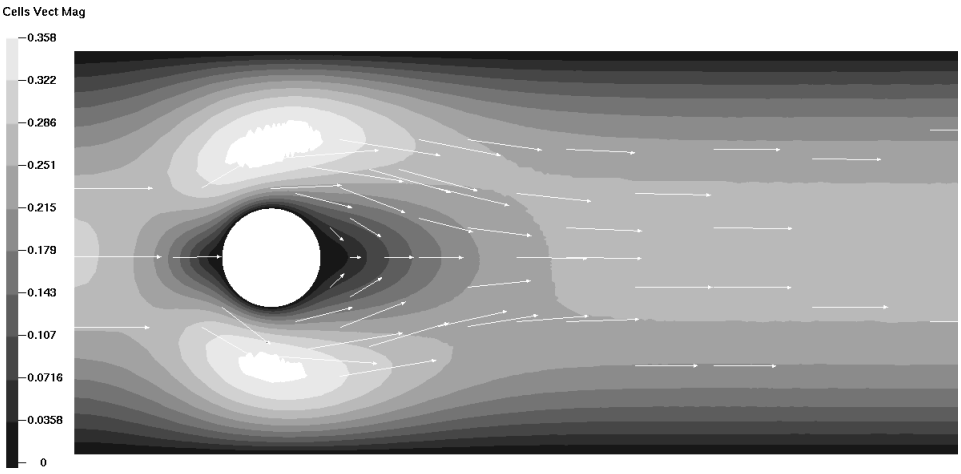

Figure 4.7: Controlled Stokes flow; primal velocity field at $t = 10.0$.

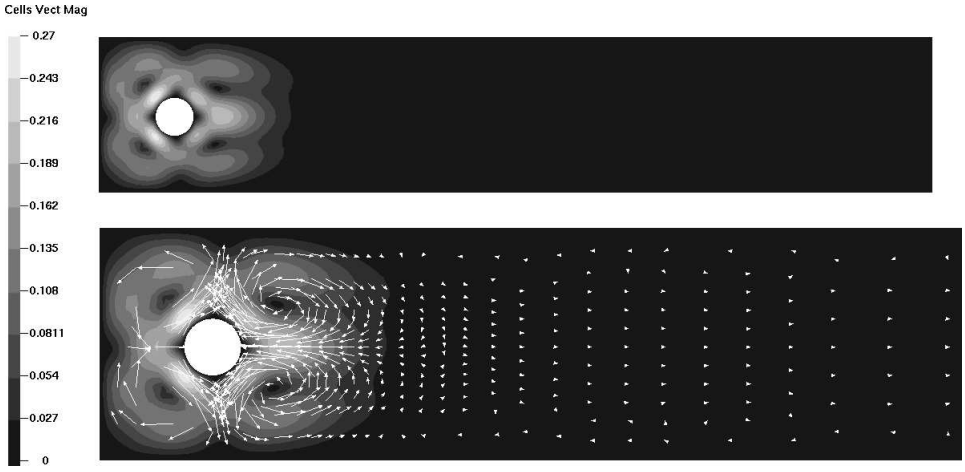

Figure 4.8: Controlled Stokes flow; control u at $t = 10.0$.

5 Conclusions

Optimal control of the time dependent Stokes equation can be carried out with iterative solution methods that act on the whole space-time cylinder. Because of the special structure of the space-time matrix, matrix-vector multiplications and preconditioners can be reduced to local operations in space, thus avoiding the necessity of storing the whole space-time matrix in memory. For preconditioning in space, the Pressure-Schur-Complement concept implemented in the FeatFlow package (`http://www.featflow.de`) can be extended to a coupled primal-dual problem and leads in combination with block-oriented solvers to efficient space-time preconditioners. Using such preconditioners as smoothers in a multigrid algorithm allows to formulate a solver with optimal efficiency, as the convergence rates are independent of the refinement of the space-time mesh. This is one of the key ingredients in the design of an optimisation-type solver where the effort for solving an optimisation problem is only a small multiple of the effort necessary for the simulation of a flow problem.

This article focuses on the basic ingredients of such a multigrid solver and illustrates the feasibility using a simple Stokes equation. The realisation of this solver as preconditioner in a nonlinear Newton-type solver for the full Navier–Stokes equations and its numerical analysis in combination with higher order time-stepping schemes will be analysed in [18, 17].

References

[1] F. Abergel and R. Temam. On some control problems in fluid mechanics. *Theoret. Comput. Fluid Dynamics*, 1:303–325, 1990.

[2] R. E. Bank and T. F. Dupond. An optimal order process for solving finite element equations. *Math. Comput.*, 36(153):35–51, 1981.

[3] G. Bärwolff and M. Hinze. Optimization of semiconductor melts. *Zeitschrift für Angewandte Mathematik und Mechanik*, 86:423–437, 2006.

[4] A. Borzi. Multigrid methods for parabolic distributed optimal control problems. *J. Comput. Appl. Math.*, 157:365–382, 2003.

[5] S. C. Brenner. An optimal-order multigrid method for P_1 nonconforming finite elements. *Math. Comput.*, 52(185):1–15, 1989.

[6] G. Büttner. *Ein Mehrgitterverfahren zur optimalen Steuerung parabolischer Probleme*. PhD thesis, Fakultät II – Mathematik und Naturwissenschaften der Technischen Universität Berlin, 2004. `http://edocs.tu-berlin.de/diss/2004/buettner_guido.pdf`.

[7] H. Goldberg and F. Tröltzsch. On a SQP–multigrid technique for nonlinear parabolic boundary control problems. In W. W. Hager and P. M. Pardalos, editors, *Optimal Control: Theory, Algorithms, and Applications*, pages 154–174. Kluwer, 1998.

[8] M. Gunzburger and S. Manservisi. Analysis and approximation of the velocity tracking problem for navier-stokes flows with distributed control. *SIAM J. Numer. Anal.*, 40:1481–1512, 2001.

[9] M. D. Gunzburger. *Perspectives in Flow Control and Optimization*. SIAM, Dec. 2002.

[10] W. Hackbusch. Fast solution of elliptic optimal control problems. *J. Opt. Theory and Appl.*, 31(4):565–581, 1980.

[11] W. Hackbusch. *Multi-Grid Methods and Applications*. Springer, Berlin, 1985.

[12] W. Hackbusch. Multigrid methods for FEM and BEM applications. In E. Stein, R. de Borst, and T. J. R. Hughes, editors, *Encyclopedia of Computational Mechanics*, chapter 20. John Wiley & Sons Ltd., 2004.

[13] M. Hinze and K. Kunisch. Second order methods for optimal control of time–dependent fluid flow. *SIAM J. Control and Optimization*, 37:925–946, 2000.

[14] M. Hinze and S. Ziegenbalg. Optimal control of the free boundary in a two-phase stefan problem with flow driven by convection. *Z. Angew. Math. Mech.*, 87:430–448, 2007.

[15] M. Hinze and S. Ziegenbalg. Optimal control of the phase interface during solidification of a gaas melt. Proceedings of the iciam 2007, zürich, INRIA Sophia-Antipolis, Laboratoire Odyssee, 2007. to appear in PAMM (2008).

[16] M. Köster. Robuste Mehrgitter-Krylowraum-Techniken für FEM-Verfahren. TU Dortmund, Diplomarbeit, 2004.

[17] M. Köster. *A Parallel High Performance Flow Solver for Optimisation with PDE Constraints*. PhD thesis, TU Dortmund, to appear.

[18] M. Köster, M. Hinze, and S. Turek. A hierarchical space-time solver for distributed control of the Navier–Stokes equation based on one-shot techniques. Preprint series DFG-SPP 1253, TU Dortmund, 2008, to appear.

[19] NETLIB. LAPACK – Linear Algebra PACKage, 1992. http://www.netlib.org/lapack/.

[20] R. Schmachtel. *Robuste lineare und nichtlineare Lösungsverfahren für die inkompressiblen Navier–Stokes-Gleichungen*. PhD thesis, TU Dortmund, June 2003. http://www.mathematik.tu-dortmund.de/lsiii/static/schriften_eng.html.

[21] S. Turek. *Efficient Solvers for Incompressible Flow Problems: An Algorithmic and Computational Approach*. Springer, Berlin, 1999.

[22] S. P. Vanka. Implicit multigrid solutions of Navier-Stokes equations in primitive variables. *J. Comput. Phys.*, 65:138–158, 1985.

[23] H. Wobker and S. Turek. Numerical studies of Vanka-type smoothers in computational structural mechanics. *Comp. Meth. Appl. Math. (CMAM)*, 2008, submitted.

[24] H. Yserentant. Old and new convergence proofs for multigrid methods. *Acta Numerica*, pages 1–44, 1992.