

A Hierarchical Space-Time Solver for Optimal Distributed Control of Fluid Flow

Michael Hinze ^{*} Michael Köster^{†‡} Stefan Turek [§]

December 9, 2008

Abstract

We present a space-time hierarchical solution concept for optimization problems governed by the time-dependent Stokes– and Navier–Stokes system. Discretisation is carried out with finite elements in space and a one-step- θ -scheme in time. By combining a Newton solver for the treatment of the nonlinearity with a space-time multigrid solver for linear subproblems, we obtain a robust solver whose convergence behaviour is independent of the number of unknowns of the discrete problem and robust with regard to the considered flow configuration. A set of numerical examples analyses the solver behaviour for various problem settings with respect to the efficiency of this approach.

1 Introduction

Active flow control plays a central role in many practical applications such as e.g. control of crystal growth processes [8, 12, 13], 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.

The underlying mathematical formulation is a minimisation problem with PDE constraints. By exploiting the special structure of the first order necessary optimality conditions, the so called Karush-Kuhn-Tucker (KKT)-system, we are able to develop a hierarchical solution approach for the optimisation of the Stokes– and Navier–Stokes equation which satisfies

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

^{*}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

for a constant $C > 0$ of moderate size, independent of the refinement level. Tests show a factor of 10 – 30 for the Navier–Stokes equation. Here, the effort for the simulation is assumed to be optimal in that sense, that a solver needs about $O(N)$ operations, $N \in \mathbb{N}$ denoting the total number of unknowns for a given computational mesh in space and time. This can be achieved by utilising appropriate multigrid techniques for the linear subproblems in space. Because of (1.1), the developed solution approach for the optimal control problem has also complexity $O(N)$; this is achieved by a combination of a space-time Newton approach for the nonlinearity and a space-time multigrid approach for linear subproblems. The complexity of this algorithm distinguishes our method from adjoint-based steepest descent methods used to solve optimisation problems in many practical applications, which in general do not satisfy this complexity requirement. 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* [9] is developed, compare also [7].

The paper is organised as follows: In Section 2 we describe the discretisation of a flow control problem and give an introduction to the ingredients needed to design a 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 for linear and a Newton solver for nonlinear problems. Finally, Section 4 is devoted to numerical examples which we present to confirm the predicted behaviour.

2 Problem formulation and discretisation

We consider the optimal control problem

$$\begin{aligned}
 J(y, u) := \frac{1}{2} \|y - z\|_{L^2(Q)}^2 + \frac{\alpha}{2} \|u\|_{L^2(Q)}^2 + \frac{\gamma}{2} \|y(T) - z(T)\|_{L^2(\Omega)}^2 &\longrightarrow \min! & (2.1) \\
 \text{s.t.} \quad y_t - \nu \Delta y + y \nabla y + \nabla p &= u & \text{in } Q, \\
 -\operatorname{div} y &= 0 & \text{in } Q, \\
 y(0, \cdot) &= y^0 & \text{in } \Omega, \\
 y &= g & \text{at } \Sigma,
 \end{aligned}$$

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 $g : \Sigma \rightarrow \mathbb{R}^d$ 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 .

The first order necessary optimality conditions are then given through the so called *Karush-Kuhn-Tucker* system

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

where λ denotes the dual velocity and ξ the dual pressure. We eliminate u in the KKT system, and (ignoring boundary conditions at the moment), we obtain

$$\begin{aligned}
y_t - \nu \Delta y + y \nabla y + \nabla p &= -\frac{1}{\alpha} \lambda, && (2.2) \\
-\operatorname{div} y &= 0, \\
y(0, \cdot) &= y_0,
\end{aligned}$$

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

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

Coupled discretisation in time

In the next step, we semi-discretise in time. For stability reasons (cf. [18]) we prefer implicit time stepping techniques that allow a large timestep and therefore lead to less unknowns in time. 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. For the time discretisation of (2.2) this yields

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

where $N \in \mathbb{N}$, $n = 0, \dots, N - 1$ and $\Delta t = 1/N$. To (2.2), (2.3) we apply the discretisation recipe from [2]. For this purpose, we define the following operators: $\mathcal{A}v := -\nu \Delta v$, $\mathcal{I}v := v$, $\mathcal{G}q := \nabla q$, $\mathcal{D}v := -\operatorname{div} v$, $\mathcal{K}_i v := \mathcal{K}(y_i)v := (y_i \nabla)v$, $\overline{\mathcal{K}}_i v := \overline{\mathcal{K}}(y_i)v := (v \nabla)y_i$ and $\mathcal{C}_i v := \mathcal{C}(y_i)v := \mathcal{A}v + \mathcal{K}(y_i)v$ for all velocity vectors v and and pressure functions q in space.

With $x := (y_0, p_0, y_1, p_1, \dots, y_n, p_n)$, this yields the nonlinear system of the primal equation,

$$\mathcal{H}x := \mathcal{H}(x)x = \begin{pmatrix} \frac{\mathcal{I}}{\Delta t} + \mathcal{C}_0 & \mathcal{G} & & \\ \mathcal{D} & & & \\ -\frac{\mathcal{I}}{\Delta t} & \frac{\mathcal{I}}{\Delta t} + \mathcal{C}_1 & \mathcal{G} & \\ \mathcal{D} & & & \\ & \ddots & \ddots & \ddots \\ & & -\frac{\mathcal{I}}{\Delta t} & \frac{\mathcal{I}}{\Delta t} + \mathcal{C}_n & \mathcal{G} \\ & & \mathcal{D} & & \end{pmatrix} \begin{pmatrix} y_0 \\ p_0 \\ y_1 \\ p_1 \\ \vdots \\ y_n \\ p_n \end{pmatrix} = \begin{pmatrix} (\frac{\mathcal{I}}{\Delta t} + \mathcal{C}_0)y^0 \\ 0 \\ -\frac{\lambda_1}{\alpha} \\ 0 \\ \vdots \\ -\frac{\lambda_n}{\alpha} \\ 0 \end{pmatrix}, \quad (2.5)$$

which is equivalent to (2.4) if y^0 is solenoidal. In the second step, we focus on the Fréchet derivative of the Navier–Stokes equation. For a vector (\bar{y}, \bar{p}) the Fréchet derivative in (y, p) reads

$$\mathcal{F}(y, p) \begin{pmatrix} \bar{y} \\ \bar{p} \end{pmatrix} := \begin{pmatrix} \bar{y}_t - \nu \Delta \bar{y} + (\bar{y} \nabla y + y \nabla \bar{y}) + \nabla \bar{p} \\ -\operatorname{div} \bar{y} \end{pmatrix}.$$

We again carry out the time discretisation as above. For vectors $x := (y_0, p_0, y_1, p_1, \dots, y_n, p_n)$ and $\bar{x} := (\bar{y}_0, \bar{p}_0, \bar{y}_1, \bar{p}_1, \dots, \bar{y}_n, \bar{p}_n)$ this results in the scheme

$$\mathcal{M}\bar{x} := \mathcal{M}(x)\bar{x} = \begin{pmatrix} \frac{\mathcal{I}}{\Delta t} + \mathcal{N}_0 & \mathcal{G} & & \\ \mathcal{D} & & & \\ -\frac{\mathcal{I}}{\Delta t} & \frac{\mathcal{I}}{\Delta t} + \mathcal{N}_1 & \mathcal{G} & \\ \mathcal{D} & & & \\ & \ddots & \ddots & \ddots \\ & & -\frac{\mathcal{I}}{\Delta t} & \frac{\mathcal{I}}{\Delta t} + \mathcal{N}_n & \mathcal{G} \\ & & \mathcal{D} & & \end{pmatrix} \begin{pmatrix} \bar{y}_0 \\ \bar{p}_0 \\ \bar{y}_1 \\ \bar{p}_1 \\ \vdots \\ \bar{y}_n \\ \bar{p}_n \end{pmatrix} \quad (2.6)$$

with the additional operator $\mathcal{N}_i := \mathcal{N}(y_i) := \mathcal{A} + \mathcal{K}(y_i) + \bar{\mathcal{K}}(y_i)$. The time discretisation of the dual equation corresponding to \mathcal{H} is now defined as the adjoint \mathcal{M}^* of \mathcal{M} ,

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

where $\lambda := (\lambda_0, \xi_0, \lambda_1, \xi_1, \dots, \lambda_n, \xi_n)$. With $\mathcal{N}_i^* := \mathcal{N}^*(y_i) = \mathcal{A} - \mathcal{K}(y_i) + \bar{\mathcal{K}}^*(y_i)$, $\bar{\mathcal{K}}^*(y_i)v = (\nabla y)^t v$ for all velocity vectors v , this reads

$$\mathcal{M}^*\lambda = \mathcal{M}^*(x)\lambda = \begin{pmatrix} \frac{\mathcal{I}}{\Delta t} + \mathcal{N}_0^* & \mathcal{G} & -\frac{\mathcal{I}}{\Delta t} & & \\ \mathcal{D} & & & & \\ & \frac{\mathcal{I}}{\Delta t} + \mathcal{N}_1^* & \mathcal{G} & -\frac{\mathcal{I}}{\Delta t} & \\ & \mathcal{D} & & & \\ & & \ddots & \ddots & \ddots \\ & & & \frac{\mathcal{I}}{\Delta t} + \mathcal{N}_n^* & \mathcal{G} \\ & & & \mathcal{D} & \end{pmatrix} \begin{pmatrix} \lambda_0 \\ \xi_0 \\ \lambda_1 \\ \xi_1 \\ \vdots \\ \lambda_n \\ \xi_n \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ y_1 - z_1 \\ 0 \\ \vdots \\ (1 + \frac{\gamma}{\Delta t})(y_n - z_n) \\ 0 \end{pmatrix} \quad (2.7)$$

3 The multigrid and the Newton 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 (linearised) KKT system as it is an ideal solver for elliptic PDEs.

We formally define the solution approach as outer nonlinear loop that has to solve a linear subproblem in each nonlinear step.

3.1 The outer defect correction loop

To treat the nonlinearity in the underlying Navier–Stokes equation, we use a standard nonlinear fixed point iteration as well as a space-time Newton iteration. Both algorithms can be written down as fully discrete defect correction loop,

$$w_{i+1}^h := w_i^h + C(w_i^h)^{-1}(f^h - G^h(w_i^h)w_i^h), \quad i \in \mathbb{N}.$$

For the fixed point method, we choose $C(w^h) := G^h(w^h)$ as preconditioner, while the space-time Newton method is characterised by $C(w^h) := F^h(w^h)$ with $F^h(w^h)$ being the discrete analogon to $F(w)$ from Section 2. $C(w^h)^{-1}$ is applied by the following space-time multigrid method.

3.2 The multigrid solver

To formulate the multigrid solver, let $\Omega_1, \dots, \Omega_k$ 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, \dots, V_k to refer to the different Finite Element spaces in space built upon these meshes. Furthermore, let T_1, \dots, 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

$$G^i w^i = f^i, \quad i = 1, \dots, k$$

of the form (2.10) 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 $I : W_i \rightarrow W_{i+1}$ the prolongation and by $R : W_i \rightarrow W_{i-1}$ the corresponding restriction. Furthermore, let $S : W_i \rightarrow W_i$ define a *smoothing* operator (see the following sections for a definition of these operators) and let us denote with $NSMpre, NSMpost \in \mathbb{N}$ the numbers of pre- and postsmoothing steps, respectively. With these components and definitions, Algorithm 1 implements a basic multigrid V-cycle; for variations of this algorithm which use the W- or F-cycle, see [1, 10, 22].

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
     $d \leftarrow R(f - G^k w)$  ▷ restriction 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

```

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 [11] and space prolongation/restriction. Let $k \in \mathbb{N}$ be the space level. Then, we denote by $I_S : V_k \rightarrow V_{k+1}$ the prolongation in space and $R_S : V_{k+1} \rightarrow V_k$ the corresponding restriction. The prolongation for a space-time vector $w^k = (w_0^k, \dots, w_N^k)$ can be written as:

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

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 $d^k = (d_0^k, \dots, d_{2N}^k)$ follows directly:

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

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 [14, 18].

3.4 Smoothing operators

The special matrix structure of the global space-time matrix (2.10) allows to define iterative smoothing operators for this algorithm based on defect correction. 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 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

```

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
     $x^{old} \leftarrow x$ 
    for  $i = N$  downto  $0$  do ▷ Backward in time
       $d_i \leftarrow r_i - \tilde{M}_i x_{i-1}^{old} - G_i^k x_i^{old} - \hat{M}_i (\omega x_{i+1} + (1 - \omega) x_{i+1}^{old})$ 
       $x_i \leftarrow x_i^{old} + (G_i^k)^{-1} d_i$ 
    end for
     $x^{old} \leftarrow x$ 
    for  $i = 0$  to  $N$  do ▷ Forward in time
       $d_i \leftarrow r_i - \tilde{M}_i (\omega x_{i-1} + (1 - \omega) x_{i-1}^{old}) - G_i^k x_i^{old} - \hat{M}_i x_{i+1}^{old}$ 
       $x_i \leftarrow x_i^{old} + (G_i^k)^{-1} d_i$ 
    end for
  end for
   $w \leftarrow w + x$  ▷ Correction
  return  $w$  ▷ Solution
end function

```

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.

Note that the key feature of both algorithms is the solution of the system $G_i^k c_i = d_i$ which means to solve fully coupled KKT system in one time step. The full space-time algorithm therefore reduces to an algorithm in space. It can be carried out without the necessity of setting up and saving the whole space time matrix in memory. The system $G_i^k c_i = d_i$ is a coupled saddle point problem for primal and dual velocity and pressure. For solving it, one can use e.g. direct solvers (as long as the number of unknowns in space is not too large) or sophisticated techniques from computational fluid dynamics, namely a space multigrid with Pressure-Schur-Complement based smoothers. A typical approach can be seen in the next section.

3.5 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. [1, 5, 10, 22, 18]), we restrict here to a short introduction into the pressure Schur complement (‘PSC’) approach for CFD problems (see also [17, 20, 21]) 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 [15].

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 $I(iel)$.

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

Algorithm 4 PSC-Smoothing 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

```

C_{iel}^{-1} by invoking a LU decomposition, e.g. with the LAPACK package [16]. 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 [18]), 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 $\text{diag}(\cdot)$ refers to the operator taking only the diagonal of a given matrix. The local preconditioner can then be formulated as $C_{iel} := \hat{A}_{I(iel), I(iel)}$. This approach decouples the primal from the dual variables in the local system. Applying $\hat{A}_{I(iel), I(iel)}^{-1}$ therefore decomposes into two independent subproblems which is much faster but leads to reduced stability. Most of the numerical tests in the later sections were carried out using PSCSMOOTHERDIAG except where noted. We note that it is even possible to increase the stability by applying this approach to patches of cells (cf. [17]) but we do not apply this approach here.

4 Numerical examples

In this section we numerically analyse the proposed solver strategy with respect to robustness and efficiency. The nonlinearity is captured by a space-time fixed point/Newton iteration, both preconditioned by the proposed space-time multigrid.

4.1 The *driven-cavity* example

4.1.1 Example (Driven-Cavity configuration). Let a domain $\Omega = [0, 1]^2$ be given. On the four boundary edges $\Gamma_1 := \{0\} \times (0, 1)$, $\Gamma_2 := [0, 1] \times \{0\}$, $\Gamma_3 := \{1\} \times (0, 1)$, $\Gamma_4 := [0, 1] \times \{1\}$ we describe Dirichlet boundary conditions as $u(x) = (0, 0)$ for $x \in \Gamma_1 \cup \Gamma_2 \cup \Gamma_3$ and $u(x) = (1, 0)$ for $x \in \Gamma_4$. The coarse grid consists of only one quadratic element. As time domain we define the interval $[0, T]$ with $T = 1$. The viscosity parameter is set to $\nu = 1/400$. On the whole space-time cylinder, we generate a target flow z by computing a simulation of the

nonstationary *Stokes* equation with right hand side $f := 0$, viscosity parameter $\nu = 1/400$ as well and the initial flow at $t = 0$ at rest.

A picture of the streamlines of the target flow, the uncontrolled flow and the controlled flow (for $\alpha = 0.01$, $\gamma = 0$) at time $t = 1$ can be seen in Figure 4.1. For better visualisation, the streamlines of the big vortex correspond to the value interval $[-0.1, 0]$ while the streamlines of the small vortices in the bottom corners highlight the value interval $[0, 1e-6]$ for the target and controlled flow, respectively, and $[0, 4e-4]$ for the uncontrolled flow. Table 4.1 shows the number of unknowns for a pure forward simulation and the optimisation for this problem.

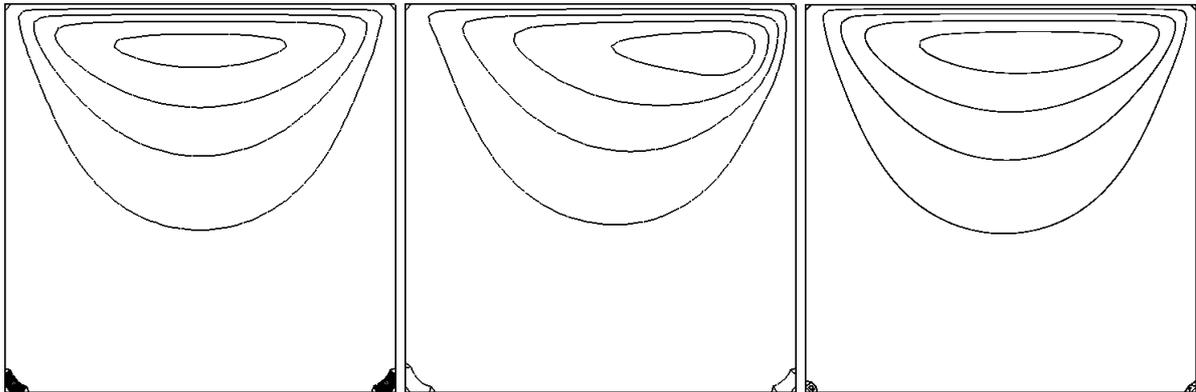


Figure 4.1: *Driven-cavity* example, Stokes (target-) flow. Streamlines of the (Stokes-) target flow (left), the uncontrolled (Navier–Stokes) flow (center) and the controlled (Navier–Stokes) flow (right) at time $t = 1$.

		simulation		optimisation	
Δt	h	#DOF space	#DOF total	#DOF space	#DOF total
1/8	1/8	352	3 168	704	6 336
1/16	1/16	1344	22 848	2 688	45 696
1/32	1/32	5248	173 184	10 496	346 368
1/64	1/64	20736	1 347 840	41 472	2 695 680

Table 4.1: Problem size we use in our numerical tests in the simulation and the optimisation on different refinement levels, driven cavity example. ‘#DOF space’ describes the number of degrees of freedom in space, i.e. in every timestep. ‘#DOF total’ refers to the total number of degrees of freedom on the whole space-time cylinder including the initial condition.

In the first couple of tests we analyse the behaviour of our solver when being applied to the driven cavity example. For our tests, we choose the regularisation parameters in the KKT-system to be $\alpha = 0.01$ and $\gamma = 0$ and start with defining a basic coarse mesh in space and time. For simplicity, we choose $\Delta t = h = 1.0$, although any other relation between Δt and h would be suitable as well. This mesh is simultaneously refined by regular refinement in space and time. On each space-time level, we perform the following tests: 1.) We calculate

a pure simulation with a fully implicit Navier–Stokes solver in time. In each timestep the norm of the residual was reduced by $\varepsilon_{\text{SimNL}} = 10^{-5}$. The linear multigrid subsolver in each nonlinear iteration reduces the norm of the residual by $\varepsilon_{\text{SimMG}}$. 2.) We calculate an optimal control problem with the target flow as specified above. The nonlinear space-time solver reduces the norm of the residual by $\varepsilon_{\text{OptNL}} = 10^{-5}$, the linear space-time multigrid in each nonlinear iteration by $\varepsilon_{\text{OptMG}}$. The convergence criterion of the innermost spatial multigrid solver in each timestep was set to reduce the norm of the residual by $\varepsilon_{\text{SpaceMG}}$.

General tests

In the first couple of tests we analyse the behaviour of the nonlinear space-time solver for optimal control. We fix the space-time mesh to $\Delta t = h = 1/16$, the convergence criterion of the innermost solver to $\varepsilon_{\text{SpaceMG}} = 10^{-2}$. The smoother in space is BiCGStab with PSC-SMOOTHERDIAG preconditioner, the space-time smoother FBSORSMOOTHER($\omega = 0.9$). From Table 4.2 one can see linear convergence for the fixed point iteration and quadratic convergence of the Newton iteration. Note that because of $\varepsilon_{\text{OptNL}} = 10^{-5}$ the impact of the parameter $\varepsilon_{\text{OptMG}}$ to Newton is rather low, it does not (yet) influence the number of iterations.

The next test analyses the influence of the innermost stopping criterion $\varepsilon_{\text{SpaceMG}}$. For the two space-time smoothers JACSMOOTHER($\omega = 0.7$, NSM = 4) and FBSORSMOOTHER($\omega = 0.9$, NSM = 1) we fix the convergence criterion of the space-time multigrid to $\varepsilon_{\text{OptMG}} = 10^{-6}$ and calculate the fixed point and Newton iteration for different settings of $\varepsilon_{\text{SpaceMG}}$. As one can see from Table 4.3, the solver behaves very robust against this parameter. We choose $\varepsilon_{\text{SpaceMG}} = 10^{-2}$ for all later tests. Furthermore one can see, that JACSMOOTHER is by far less efficient than FBSORSMOOTHER: Although there are 4 times more smoothing steps used, the iteration needs more iterations and more computational time. Therefore, we drop JACSMOOTHER in our future tests.

Step \ $\varepsilon_{\text{OptMG}}$	fixed point		Newton	
	10^{-2}	10^{-6}	10^{-2}	10^{-6}
0	5.59E-04	5.59E-04	5.59E-04	5.59E-04
1	1.35E-04	1.35E-04	9.90E-05	9.81E-05
2	3.43E-05	3.34E-05	3.18E-06	2.99E-06
3	9.80E-06	9.10E-06	7.95E-09	2.70E-09
4	3.01E-06	2.69E-06	3.91E-11	
5	9.91E-07	8.51E-07		
6	3.33E-07	2.73E-07		
7	1.09E-07	8.52E-08		
8	3.43E-08	2.54E-08		
9	1.03E-08	7.25E-09		
10	2.98E-09	1.98E-09		

Table 4.2: *Driven-cavity* example. Behaviour of the space-time fixed point and Newton solver for different settings of $\varepsilon_{\text{OptMG}}$ for the linear space-time subproblems.

		JACSMOOTHER(NSM=4)			FBSORSMOOTHER(NSM=1)		
Nonl. Solv.	$\varepsilon_{\text{SpaceMG}}$	#NL	#MG	Time	#NL	#MG	Time
fixed point	10^{-1}	10	110	590.0	10	69	285.9
	10^{-2}	10	110	617.0	10	68	296.3
	10^{-3}	10	110	735.8	10	68	349.5
	10^{-6}	10	110	1207.3	10	68	528.3
Newton	10^{-1}	3	34	219.8	3	23	99.8
	10^{-2}	3	34	228.1	3	23	102.7
	10^{-3}	3	34	322.7	3	23	123.6
	10^{-6}	3	34	541.7	3	23	179.0

Table 4.3: The global nonlinear solver is not influenced by the convergence criterion of the multigrid solver in space. ‘#NL’ describes the number of iterations in the nonlinear solver, ‘#MG’ the number of iterations in the space-time multigrid solver, ‘Time’ the computational time in seconds; *Driven-cavity* example.

$\varepsilon_{\text{OptMG}}$:		10^{-2}				10^{-6}			
Nonl. Solv.:		fixed point		Newton		fixed point		Newton	
Δt	h	#NL	#MG	#NL	#MG	#NL	#MG	#NL	#MG
1/8	1/8	6	13	4	12	6	46	3	27
1/16	1/16	10	20	4	11	10	68	3	23
1/32	1/32	8	16	4	10	8	53	3	21
1/64	1/64	6	12	4	8	6	40	3	18

Table 4.4: Number of iterations in the nonlinear (#NL) and linear (#MG) space-time solver, for various levels of refinement and different settings for $\varepsilon_{\text{OptMG}}$. The smoother is FBSOR-SMOOTHER($\omega = 0.9$); *Driven-cavity* example.

Table 4.4 reveals the dependence of the nonlinear space-time solver on the convergence criterion of the space-time multigrid solver. The number of linear and nonlinear iterations stays constant upon increasing the resolution of the space-time mesh which shows the linear complexity of the algorithm. More precisely, the number of iterations even reduces with increasing space-time level – an effect which was also observed and proven in [6].

Furthermore, for a convergence criterion of $\varepsilon_{\text{OptNL}} = 10^{-5}$, a reduction of $\varepsilon_{\text{OptMG}}$ from 10^{-2} to 10^{-6} has no visible influence to the nonlinear solver but increases the number of linear iterations by a factor of approx. 3. Thus we take $\varepsilon_{\text{OptMG}} = 10^{-2}$ for all further numerical tests.

Optimisation vs. Simulation

In the following tests we want to compare the behaviour of the solver for the optimal control problem with a pure simulation. As convergence criterion for the solver we choose

$\varepsilon_{\text{SimNL}} = \varepsilon_{\text{OptNL}} = 10^{-5}$ and $\varepsilon_{\text{SimMG}} = \varepsilon_{\text{OptMG}} = 10^{-2}$. Table 4.5 depicts the result of a set of forward simulations for various settings of Δh and t . As nonlinear solver in each time step of the simulation, we use on the one hand a simple nonlinear fixed point iteration, on the other hand a Newton iteration. The linear solver used here was multigrid in space with a PSCSMOOTHERDIAG-type smoother.

Nonl. Solv.	Δt	h	#NL	#MG	\circ #NL	\circ #MG	T_{sim}	T_{opt}	$\frac{T_{\text{opt}}}{T_{\text{sim}}}$
fixed point	1/8	1/8	51	165	6	21	0.46	14.21	31.0
	1/16	1/16	94	525	6	33	2.20	74.83	34.0
	1/32	1/32	159	1009	5	32	15.76	351.21	22.3
	1/64	1/64	265	1925	4	30	132.43	1568.97	11.8
Newton	1/8	1/8	25	79	3	10	0.36	14.25	39.2
	1/16	1/16	50	302	3	19	2.27	48.14	21.2
	1/32	1/32	99	762	3	24	18.73	301.99	16.1
	1/64	1/64	196	1669	3	26	233.34	1348.12	5.8

Table 4.5: *Driven-cavity*-example: Total and mean number of nonlinear and linear iterations per timestep for a forward simulation on different levels of refinement. Comparison of the execution time for a simulation and a corresponding optimisation

The columns \circ #NL and \circ #MG in this table describe the average number of linear/non-linear iterations per time step which are comparable to the number of nonlinear/linear iterations in Table 4.4. The total number iterations of the space-time multigrid solver in the optimisation solver differs to the total number of iterations of the space multigrid solver by a factor of approx. 2–3, which means that the effort for both, the simulation and the optimisation, grows with same complexity when increasing the problem size.

Table 4.4 as well compares the different execution times of the simulation and optimisation solver. While the fraction of the execution times is still inaccurate for ‘small’ mesh resolutions ($\Delta t, h \leq 1/16$) in conjunction with the fixed point method, using the Newton method clearly shows that the execution time of the optimisation is a bounded multiple of the execution time of the simulation. One can see a factor of $C \approx 10 - 30$, even being better for higher mesh resolutions. We note here that the high factors for small meshes are merely an effect of acceleration due to cache effects and can be expected: A simulation with only some hundreds of unknowns is able to rather completely run in the cache, whereas the optimisation of a fully nonstationary PDE in space and time usually does not fit into the computer cache anymore. A more detailed analysis and exploitation of the computer cache and its effects can be found in [3, 19].

4.2 Tests with the *flow-around-cylinder* example

In a similar way as above, we now carry out a set of tests for the more complicated *flow-around-cylinder* problem:

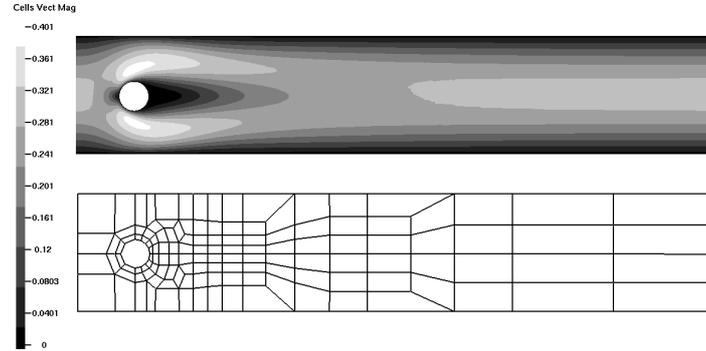


Figure 4.2: Top: Uncontrolled Navier–Stokes flow in the *flow-around-cylinder* example. Velocity magnitude, fully developed stationary flow. Bottom: Underlying mesh, coarse grid.

Flow-around-cylinder:		simulation		optimisation	
Δt	Space-Lv.	#DOF space	#DOF total	#DOF space	#DOF total
1/10	2	2704	29 744	5 408	59 488
1/20	3	10608	222 768	21 216	445 536
1/40	4	42016	1 722 656	84 032	3 445 312
1/80	5	167232	13 545 792	334 464	27 091 584

Table 4.6: Problem size we use in our numerical tests in the simulation and the optimisation on different refinement levels. ‘#DOF space’ describes the number of degrees of freedom in space, i.e. in every timestep. ‘#DOF total’ refers to the total number of degrees of freedom on the whole space-time cylinder including the initial condition.

Nonl. Solv.			Simulation				Optimisation	
	Δt	Space-Lv.	#NL	#MG	\circ #NL	\circ #MG	#NL	#MG
fixed point	1/20	3	100	486	5	24	5	24
	1/40	4	167	946	4	24	5	15
	1/80	5	309	1900	4	24	4	11
Newton	1/20	3	63	312	3	16	4	24
	1/40	4	123	709	3	18	4	14
	1/80	5	246	1589	3	20	4	13

Table 4.7: Total and mean number of nonlinear and linear iterations per timestep for a forward simulation and a corresponding optimisation on different levels of refinement; *flow-around-cylinder* example.

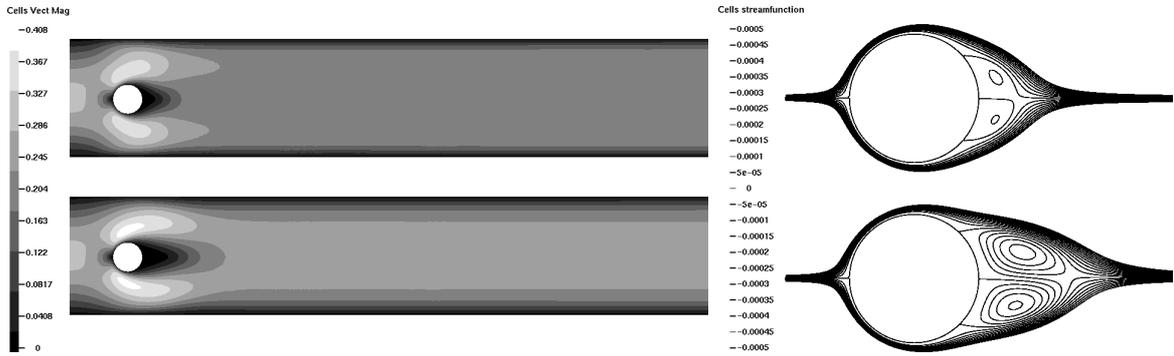


Figure 4.3: *Flow-around-cylinder* example. Uncontrolled Navier–Stokes flow at time $t = 0.5$ (top) and $t = 1$ (bottom). Left: Velocity magnitude field. Right: Streamlines around the cylinder. Note: The nonsymmetry stems from the nonsymmetric position of the cylinder.

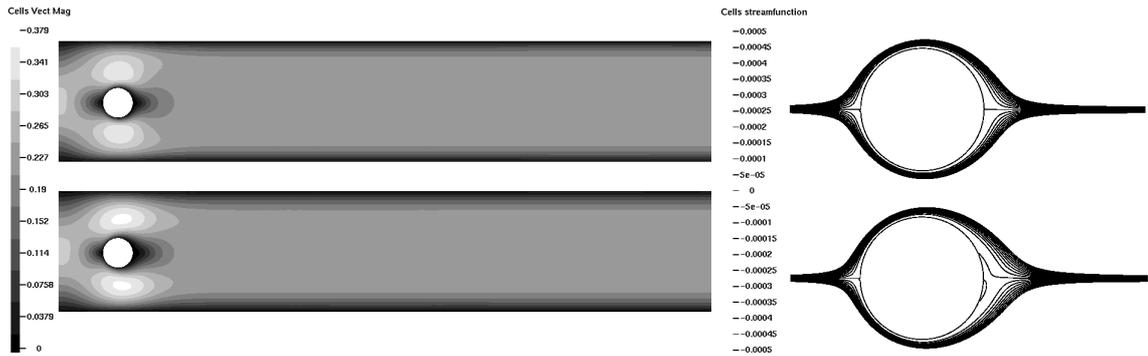


Figure 4.4: *Flow-around-cylinder* example. Velocity magnitude (left) and streamlines around the cylinder (right) controlled solution at $t = 0.5$ (top) and $t = 1$ (bottom).

Nonl. Solv.	Δt	Space-Lv.	T_{sim}	T_{opt}	$\frac{T_{\text{opt}}}{T_{\text{sim}}}$
fixed point	1/20	3	22.1	739.27	33.5
	1/40	4	167.9	3509.51	20.9
	1/80	5	1721.7	26741.82	15.5
Newton	1/20	3	27.0	852.33	31.6
	1/40	4	209.6	4037.21	19.3
	1/80	5	2227.1	37217.81	16.7

Table 4.8: Comparison of the execution time for a simulation and a corresponding optimisation. *Flow-around-cylinder* example.

		fixed point		Newton	
Δt	Space-Lv.	#NL	#MG	#NL	#MG
1/20	4	5	13	3	11
1/40	4	5	21	4	21
1/80	4	5	28	4	27

Table 4.9: *Flow-around-cylinder* example. Convergence behaviour of the solver for fixed space-level and different Δt .

α	0.05		0.01		0.005	
γ	#NL	#MG	#NL	#MG	#NL	#MG
0	5	15	5	15	5	15
0.1	5	15	5	15	5	15
0.5	5	15	5	40	5	79

Table 4.10: *Flow-around-cylinder* example. Number of nonlinear and linear steps of the solver when modifying α and γ .

4.2.1 Example (Flow-around-cylinder configuration). As spatial domain, we prescribe a rectangle without an inner cylinder $\Omega := [0, 2.2] \times [0, 0.41] \setminus B_r(0.2, 0.2)$, $r = 0.05$, cf. [18]. We decompose the boundary of this domain into five parts: $\Gamma_1 := \{0\} \times [0, 0.41]$, $\Gamma_2 := (0, 2.2] \times \{0\}$, $\Gamma_3 := \{2.2\} \times (0, 0.41)$, $\Gamma_4 := [0, 2.2] \times \{0.41\}$ and $\Gamma_5 := \partial B_r(0.2, 0.2)$. As boundary conditions, we define $u(x) := (0, 0)$ for $x \in \Gamma_2 \cup \Gamma_4 \cup \Gamma_5$. On Γ_3 we prescribe do-nothing boundary conditions while on Γ_1 a parabolic inflow profile with maximum velocity $U_{\max} := 0.3$ is used. The time interval for this test case we define as $[0, T]$ with $T = 1$. To generate a target flow z , we compute a nonstationary simulation using the *Stokes equation*, the initial flow at $t = 0$ at rest. The right hand side is set to $f := 0$ and viscosity parameter to $\nu = 1/1000$ (resulting in $\text{Re}=20$).

Figure 4.2 depicts the basic mesh and the uncontrolled flow in the stationary limit. Figure 4.3 visualises the velocity field and streamlines of the uncontrolled Navier–Stokes equation at time $t = 0.5$ and $t = 1$. The fully developed stationary Navier–Stokes flow as well as the underlying mesh can be seen in Figure 4.2. For the time discretisation, we choose $N = 5$ timesteps on time level 1 which leads to 10, 20, 40 and 80 timesteps space-time level 2..5. In Table 4.6 the corresponding problem size for the simulation and optimisation can be seen. The convergence criteria for the solvers are defined as $\varepsilon_{\text{SimNL}} = \varepsilon_{\text{OptNL}} = 10^{-5}$, $\varepsilon_{\text{SimMG}} = \varepsilon_{\text{OptMG}} = \varepsilon_{\text{SpaceMG}} = 10^{-2}$ and we again focus on the difference in the execution time between the simulation and a corresponding optimisation. Figure 4.4 shows a picture of the velocity and streamline field of the controlled flow at $t = 0.5$ and $t = 1.0$ which are similar to the target flow.

Table 4.7 depicts the total number of nonlinear (#NL) and linear (#MG) iterations as well as the mean number of nonlinear and linear iterations per timestep for both, the optimisation and the simulation. Like in the driven-cavity example, the number of nonlinear iterations for

the optimisation is comparable to the mean number of nonlinear iterations in the simulation, and so it does for the number of linear iterations. Table 4.8 again confirms that the execution time of the simulation and the optimisation¹ differs by only a constant factor, which is better for higher levels of refinement. The table indicates a factor $C \approx 15 - 30$.

Table 4.9 now analyses the effect to the solver when modifying Δt against h . We fixed the space level to 4 and used 20, 40 and 80 timesteps. The nonlinear solver is obviously not influenced by this modification, but the number of iterations of the linear solver increases with smaller timesteps. Therefore, higher order schemes using large timesteps are preferable.

Finally, Table 4.10 visualises the stability of the solver with respect to α and γ , here for $\Delta t = h = 1/16$. The solver behaves very stable for a large range of α and γ . Very small values of α with increasing values of γ nevertheless lead to a reduced stability, as the number of linear iterations increases. For $\alpha = 0.005$ and $\gamma = 0.5$ we even had to use the stronger PSCSMOOTHERFULL smoother in space due to convergence problems of the spatial problems.

Both effects indicate the way the solver is influenced by the parameter settings. A deeper analysis of these effects is a subject of future research and will help to improve the efficiency and robustness of the solver components.

4.2.2 Example (Flow-around-cylinder with pulsating inflow). The spatial domain is the same as in Example 4.2.1. We use the time cylinder $[0, T]$ with $T = 8$ and a viscosity parameter $\nu = 1/1000$. As boundary conditions, we again define $u(x) := (0, 0)$ for $x \in \Gamma_2 \cup \Gamma_4 \cup \Gamma_5$ and do-nothing boundary conditions on $\Gamma_1 \cup \Gamma_3$. To generate a target flow z , we define on Γ_1 an oscillating Dirichlet inflow boundary condition as parabolic profile with maximum inflow velocity $U_{\max}(t) := 0.15(1 + \sin(\frac{t+3}{2}\pi))/2$ and compute a full Navier–Stokes simulation with $f := 0$ as right hand side, the initial flow at $t = 0$ at rest. For the coarsest time level, we choose $N = 20$ timesteps, leading to 40, 80, 160 and 320 timesteps on space-time level 2, 3, 4 and 5, respectively. The regularisation parameters are chosen as $\alpha = 0.01$ and $\gamma = 0.1$.

Table 4.11 shows the convergence behaviour of the Newton optimisation solver. For all levels of refinement, the number of nonlinear iterations stays constant at a value of 3 and the number of linear iterations at 9, which means that we have about 3 multigrid iterations per nonlinear iteration.

We check the solution quality by measuring the drag coefficients

$$C_D = \frac{2}{(\frac{2}{3}\bar{U}_{\max})^2 \cdot 0.1} \int_{\Gamma_5} \nu \frac{\partial y_t}{\partial n} n_y - p n_x \, dS, \quad \bar{U}_{\max} := 0.15,$$

(v_t being the tangential velocity, cf. [18]) on the circular boundary component Γ_5 . Figure 4.5 visualises the drag coefficients around the cylinder for the time interval under consideration. The black line belongs to the drag coefficients in the simulation while the dashed and dotted lines depict the results from optimisation at different space/time refinement levels. The maximum drag coefficient increases with increasing refinement level and has a relative error

¹In the table, the execution time using the fixed point algorithm is usually lower than the execution time with the Newton algorithm. This stems from the fact that when using Newton, the effort for solving the spatial system in every timestep is much higher, and due to the convergence criterion $\varepsilon_{\text{OptNL}} < 10^{-5}$ the Newton and the fixed point method need approximately the same number of linear/nonlinear iterations.

to the maximum reference drag of approx. 25%. The frequencies of both curves, those from the simulation and those from the optimisation, are identical, although the curve obtained by optimisation is slightly time-shifted by a time difference of $\Delta t \approx 0.5$. Nevertheless, the optimisation solver successfully controlled the fully nonstationary flow by volume force to imitate pulsating Dirichlet inflow boundary conditions.

Nonl. Solv.	Δt	Space-Lv.	#NL	#MG
Newton	1/10	2	3	7
	1/20	3	3	9
	1/40	4	3	9
	1/80	5	3	8

Table 4.11: *Flow-around-cylinder* example with oscillating flow. Convergence behaviour of the solver.

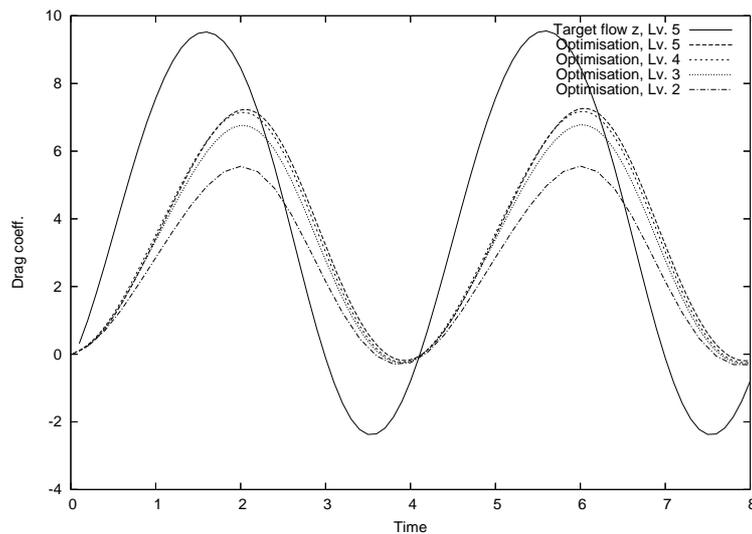


Figure 4.5: Drag coefficients in the *flow-around-cylinder* example with oscillating inflow boundary conditions. The solid black line shows the target solution obtained by simulation.

5 Conclusions

Optimal control of the time-dependent Navier–Stokes equation can be carried out with iterative nonlinear and linear solution methods that act on the whole space-time cylinder. A nonlinear Newton approach allows fast convergence of the global solution. Because of the special structure of the system matrix a space-time multigrid algorithm can be formulated for the linear subproblems in the Newton iteration. All matrix vector multiplications and smoothing operations can be reduced to local operations in space, thus avoiding the necessity of storing

the whole space-time matrix in memory. Problems in space can be tackled by efficient space-multigrid and Pressure-Schur-Complement techniques from Computational Fluid Dynamics. The overall solver works with optimal complexity, the numerical effort growing linearly with the problem size. The execution time necessary for the optimisation is a bounded multiple of the execution time necessary for a ‘similar’ simulation, where numerical tests indicate a factor of $C \approx 10 - 30$. Being based on finite elements, the solver can be applied to rather general computational meshes.

This article concentrated on the basic ingredients of the solver. For simplicity, we restricted to first order implicit Euler discretisation in time and ignored any restrictions on the controls. In the next steps, we will concentrate on higher order schemes like Crank-Nicolson for larger timesteps, bounds on the control, other finite element spaces for higher accuracy and stabilisation techniques which are necessary to compute with higher Reynolds numbers.

References

- [1] R. E. Bank and T. F. Dupond. An optimal order process for solving finite element equations. *Math. Comput.*, 36(153):35–51, 1981.
- [2] G. Bärwolff and M. Hinze. Optimization of semiconductor melts. *Zeitschrift für Angewandte Mathematik und Mechanik*, 86:423–437, 2006.
- [3] Ch. Becker, S. Kilian, and S. Turek. Hardware-oriented numerics and concepts for PDE software. In *FUTURE 1095*, pages 1–23. Elsevier, 2003. International Conference on Computational Science ICCS2002, Amsterdam.
- [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, E. Ozugurlu, J. Turner, and H. Zhang. Controlling transport phenomena in the czochralski crystal growth process. *Journal of Crystal Growth*, 234:47–62, 2002.
- [9] W. Hackbusch. Fast solution of elliptic optimal control problems. *J. Opt. Theory and Appl.*, 31(4):565–581, 1980.
- [10] W. Hackbusch. *Multi-Grid Methods and Applications*. Springer, Berlin, 1985.

- [11] W. Hackbusch. Multigrid methods for FEM and BEM applications. In E. Stein, R. de Borst, and Th. J. R. Hughes, editors, *Encyclopedia of Computational Mechanics*, chapter 20. John Wiley & Sons Ltd., 2004.
- [12] 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.
- [13] 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).
- [14] M. Köster. Robuste Mehrgitter-Krylowraum-Techniken für FEM-Verfahren, 2007. Diplomarbeit, Universität Dortmund, Diplomarbeit, http://www.mathematik.tu-dortmund.de/lisiii/static/schriften_eng.html.
- [15] M. Köster. *A Parallel High Performance Flow Solver for Optimisation with PDE Constraints*. PhD thesis, TU Dortmund, To appear 2009.
- [16] NETLIB. LAPACK – Linear Algebra PACKage, 1992. <http://www.netlib.org/lapack/>.
- [17] 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/lisiii/static/schriften_eng.html.
- [18] S. Turek. *Efficient Solvers for Incompressible Flow Problems: An Algorithmic and Computational Approach*. Springer, Berlin, 1999.
- [19] S. Turek, D. Göddeke, Ch. Becker, S. H. M. Buijssen, and H. Wobker. FEAST - realisation of hardware-oriented numerics for HPC simulations with finite elements. *CCPE*, 2008. Submitted.
- [20] S. P. Vanka. Block-implicit multigrid solution of Navier-Stokes equations in primitive variables. *J. Comput. Phys.*, 65:138–158, 1986.
- [21] H. Wobker and S. Turek. Numerical studies of Vanka-type smoothers in computational structural mechanics. *Adv. Appl. Math. Mech. (AAMM)*, 2008. Submitted.
- [22] H. Yserentant. Old and new convergence proofs for multigrid methods. *Acta Numerica*, pages 1–44, 1992.