# Algorithms for PDE-Constrained Optimization

**Roland Herzog**$^{*}$ [1] and **Karl Kunisch**$^{**}$[2]

[1] Chemnitz University of Technology, Faculty of Mathematics, Reichenhainer Straße 41, D–09126 Chemnitz, Germany
[1] Karl-Franzens University Graz, Heinrichstraße 36, A–8010 Graz, Austria

Received 15 November 2004, revised 30 November 2004, accepted 2 December 2004
Published online 3 December 2004

**Key words** PDE-Constrained Optimization, semi-smooth Newton method, control and state constraints, Navier-Stokes equations.
**MSC (2000)** 49-M05, 49-M37, 76-D55, 90-C06,

Some first and second order algorithmic approaches for the solution of PDE-constrained optimization problems are reviewed. An optimal control problem for the stationary Navier-Stokes system with pointwise control constraints serves as an illustrative example. Some issues in treating inequality constraints for the state variable and alternative objective functions are also discussed.

## 1 Introduction

In this paper we review a number of algorithmic approaches for solving optimization problems with PDE constraints. Most of these methods were originally developed for finite dimensional problems. When applied to optimization problems with PDE constraints, new aspects become important. For instance, (discretized) PDE-constrained problems are inherently large-scale. Some aspects, like mesh independent convergence behavior, can only be explained by incorporating the infinite dimensional point of view, which is not present in finite dimensional problems. Moreover, discretization and solution of PDE-constrained optimization problems should not be viewed as independent. Rather, they must be intertwined to yield efficient algorithms. Discretization and specifically adaptivity are an important aspect which will not be convered within this contribution.[1]

In this article, we provide a unified treatment of algorithms, which proceeds on a formal basis and does not distinguish between infinite dimensional and discretized problem settings. To make the presentation more concrete, we consider the solution of an optimal control problem for the stationary Navier-Stokes system as a prototypical example. The volume force acts as the control function, and the corresponding state is given by the velocity and pressure components of the solution to the Navier-Stokes equation. Clearly, this is a rather artificial setting from an engineering point of view, although a certain possibility to control the Lorentz

$^{*}$ e-mail: roland.herzog@mathematik.tu-chemnitz.de, Phone: +49 371 531 22530, Fax: +49 361 531 22509
$^{**}$ Corresponding author: e-mail: karl.kunisch@uni-graz.at, Phone: +43 316 380 5162, Fax: +43 316 380 9827
[1] see the article by Rannacher and Vexler in this issue

forces exists in electrically conducting fluids (see [1, 2]). All algorithms will be explained in this setting.

### 1.1 Black-Box versus All-at-Once Methods

When applied to PDE-constrained problems, the distinction of *black-box* vs. *all-at-once* optimization methods becomes important. The first class of methods builds upon existing solvers for the (discretized) PDE and is also known as simulation-based optimization. In other words, an existing algorithm for the solution of the state equation is embedded into an optimization loop. Since efficient optimization requires gradients, the solver for the state equation has to be augmented with a routine which provides the gradient of the state w.r.t. the optimization variables. Employing a finite difference approximation of the gradient may seem attractive due to its ease of implementation, but it has limited accuracy and it is costly in the presence of many design variables. And thus it is usually preferable to calculate the gradient using a sensitivity or adjoint approach, which may even be available more or less automatically by Algorithmic Differentiation (AD).[2] Note, however, that while *black-box* approaches are conceptually appealing, their main drawback is that they require the repeated costly solution of the (nonlinear) state equation, even in the initial stages when the design variables are still far from their optimal value. This drawback can be partially overcome by carrying out the early optimization steps with a coarsely discretized PDE, by using reduced-order models[3], or by the use of semi-implicit discretization strategies in the case of time-dependent equations (see the discussion in Remark 3.1).

By contrast, *all-at-once* methods treat the design and state variables as independent optimization variables, which are coupled through the PDE constraint. From the optimization perspective, this constraint need only be satisfied at the final iterate. The obvious advantage of this point of view is that the repeated solution of the (nonlinear) state equation is avoided. Typically, optimization algorithms of this class still require the solution of linearized state equations.

The algorithms presented in this paper can be categorized as follows:

| | black-box | all-at-once |
|---|---|---|
| 1st order | steepest descent<br>nonlinear CG | |
| 2nd order | Newton | SQP |

For brevity of presentation, we address only the very basic variants of these methods. We refer to the general literature on numerical optimization, such as [3, 4], for more details. We also refer to [5] for an overview on second-order methods for a time-dependent version of our model problem presented in Section 1.3 below, albeit in the absence of inequality constraints.

### 1.2 Outline of the Paper

In Section 2 we address gradient-type methods. These methods are attractive in view of their simplicity of implementation and global convergence properties. In particular, we discuss

[2] see the article by Gauger and Özkaya in this issue
[3] see the article by Sachs and Volkwein in this issue

the steepest descent and nonlinear conjugate gradient methods. As a drawback, gradient-type methods exhibit only linear convergence. Therefore, we turn to higher-order methods in Section 3 and discuss Newton and SQP approaches. Most of the aforementioned methods deal rather naturally with inequality constraints on the control (design) variables, which are present in most practical applications. In Section 4, we comment on possibilities of treating other, more involved types of inequality constraints and objective functions.

### 1.3 Model Problem

As a prototypical example, we consider the following optimal control problem for the stationary Navier-Stokes system

$$\left.\begin{aligned} -\nu\triangle \boldsymbol{y} + (\boldsymbol{y}\cdot\nabla)\,\boldsymbol{y} + \nabla p &= \chi_\omega \boldsymbol{u} && \text{in } \Omega \\ \nabla\cdot\boldsymbol{y} &= 0 && \text{in } \Omega \\ \boldsymbol{y} &= \boldsymbol{0} && \text{on } \Gamma = \partial\Omega \end{aligned}\right\} \tag{1.1}$$

on a given bounded domain $\Omega \subset \mathbb{R}^d$, $d \in \{2,3\}$, occupied by a fluid:

$$\begin{aligned} \text{Minimize} \quad & J(\boldsymbol{y},\boldsymbol{u}) := \frac{1}{2}\int_\Omega |\boldsymbol{y}-\boldsymbol{y}_d|^2\,\mathrm{d}x + \frac{\alpha}{2}\int_\omega |\boldsymbol{u}|^2\,\mathrm{d}x \\ \text{subject to} \quad & (1.1) \\ \text{and} \quad & \boldsymbol{u}_a \le \boldsymbol{u} \le \boldsymbol{u}_b \quad \text{in } \Omega. \end{aligned} \tag{OCP$_{\text{cc}}$}$$

The velocity $\boldsymbol{y}$ and the pressure $p$ of the fluid are the state variables of the problem, while the right hand side $\boldsymbol{u}$ (volume force on a subdomain $\omega \subseteq \Omega$) acts as the control (design) variable. The control constraints are understood in a component-wise sense. From an optimization point of view, the unknown optimization variables in (OCP$_{\text{cc}}$) are $(\boldsymbol{y}, p, \boldsymbol{u})$. Here and throughout, vector valued quantities are denoted by boldface symbols, and $\chi_\omega$ denotes the characteristic function of $\omega$.

While we proceed on a completely formal basis, we refer to [5–7] for a mathematical analysis of a time-dependent version of this problem. Note that we have assumed no-slip boundary conditions on the boundary $\Gamma$ only for simplicity of the presentation. More general optimal control problems for the Navier-Stokes system are considered, for instance, in [6, 8].

The objective function, or performance measure, in (OCP$_{\text{cc}}$) is typical and it contains two terms. The first term measures the departure of the obtained velocity profile from a desired profile $y_d$ in a least-squares sense. The second term involves a parameter $\alpha \ge 0$. It describes the cost of the control. In the presence of upper and lower bounds $\boldsymbol{u}_b$ and $\boldsymbol{u}_a$, the choice $\alpha = 0$ is an admissible and does not jeopardize the existence of a solution. Positive values of $\alpha$ stabilize the solution and therefore also increase the stability of numerical algorithms.

As was already mentioned, black-box algorithms make use of the fact that for every given control function $\boldsymbol{u}$, there exists a unique solution of the state equation. In the context of the stationary Navier-Stokes system (1.1), a unique pair of state variables $(\boldsymbol{y}, p)$ exists at least for sufficiently small control functions $\boldsymbol{u}$, see [9, Theorem IV.2.2]. Since we do not consider pressure dependent objective functions here, we use the notation $\boldsymbol{y} = \mathcal{S}(\boldsymbol{u})$ to denote the unique velocity associated with the control function $\boldsymbol{u}$. We can thus formally eliminate the state equation from our problem (OCP$_{\text{cc}}$) and obtain the following reduced optimal control problem.

$$\begin{aligned} &\text{Minimize} \quad \widehat{J}(\boldsymbol{u}) := \frac{1}{2}\int_\Omega |\mathcal{S}(\boldsymbol{u}) - \boldsymbol{y}_d|^2 \, \mathrm{d}x + \frac{\alpha}{2}\int_\omega |\boldsymbol{u}|^2 \, \mathrm{d}x \\ &\text{subject to} \quad \boldsymbol{u}_a \le \boldsymbol{u} \le \boldsymbol{u}_b \quad \text{in } \Omega. \end{aligned} \tag{ROCP$_{\text{cc}}$}$$

Note that in the reduced problem, only the control function $\boldsymbol{u}$ appears as the optimization variable. Black-box algorithms will consider the reduced problem (ROCP$_{\text{cc}}$), while all-at-once methods address (OCP$_{\text{cc}}$) where the PDE constraint is kept as an explicit side constraint to the optimization problem.

## 2 Gradient-Type Methods

Gradient-type methods are black-box methods. They are among the simplest deterministic optimization methods for the solution of (ROCP$_{\text{cc}}$) and rely solely on gradient information for the reduced objective $\widehat{J}$. We begin by computing the directional derivative of $\widehat{J}$ at $\boldsymbol{u}$

$$\begin{aligned} \langle \widehat{J}'(\boldsymbol{u}), \delta\boldsymbol{u}\rangle &= \big(\mathcal{S}(\boldsymbol{u}) - \boldsymbol{y}_d, \mathcal{S}'(\boldsymbol{u})\,\delta\boldsymbol{u}\big)_\Omega + \alpha\,\big(\boldsymbol{u}, \delta\boldsymbol{u}\big)_\omega \\ &= \big(\mathcal{S}'(\boldsymbol{u})^\star(\mathcal{S}(\boldsymbol{u}) - \boldsymbol{y}_d) + \alpha\boldsymbol{u}, \delta\boldsymbol{u}\big)_\omega \end{aligned} \tag{2.1}$$

for arbitrary directions $\delta\boldsymbol{u}$. Here and throughout, $(\boldsymbol{u}, \boldsymbol{v})_\Omega$ denotes the inner product $\int_\Omega \boldsymbol{u}\cdot\boldsymbol{v}\,\mathrm{d}x$ and similarly for $(\boldsymbol{u}, \boldsymbol{v})_\omega$. From (2.1) we infer the gradient of the reduced objective, which involves the characteristic function $\chi_\omega$ of $\omega$ and is given by

$$\nabla\widehat{J}(\boldsymbol{u}) = \chi_\omega\,\mathcal{S}'(\boldsymbol{u})^\star(\mathcal{S}(\boldsymbol{u}) - \boldsymbol{y}_d) + \alpha\boldsymbol{u}. \tag{2.2}$$

Above, the linearized solution map $\mathcal{S}'(\boldsymbol{u})$ and its adjoint $\mathcal{S}'(\boldsymbol{u})^\star$ appear. The evaluation of $\delta\boldsymbol{y} = \mathcal{S}'(\boldsymbol{u})\,\delta\boldsymbol{u}$ can be realized by solving the linearized Navier-Stokes system,

$$\left.\begin{aligned} -\nu\triangle\delta\boldsymbol{y} + (\delta\boldsymbol{y}\cdot\nabla)\,\boldsymbol{y} + (\boldsymbol{y}\cdot\nabla)\,\delta\boldsymbol{y} + \nabla\delta p &= \chi_\omega\delta\boldsymbol{u} && \text{in } \Omega \\ \nabla\cdot\delta\boldsymbol{y} &= 0 && \text{in } \Omega \\ \delta\boldsymbol{y} &= \boldsymbol{0} && \text{on } \Gamma = \partial\Omega \end{aligned}\right\} \tag{2.3}$$

with coefficients $\boldsymbol{y} = \mathcal{S}(\boldsymbol{u})$. The adjoint can be evaluated by solving the corresponding adjoint system, e.g., $\boldsymbol{z} = \mathcal{S}'(\boldsymbol{u})^\star(\boldsymbol{y} - \boldsymbol{y}_d)$ is given by the velocity component (restricted to $\omega$) of

$$\left.\begin{aligned} -\nu\triangle\boldsymbol{z} + (\nabla\boldsymbol{y})^\top\boldsymbol{z} - (\boldsymbol{y}\cdot\nabla)\,\boldsymbol{z} + \nabla q &= \boldsymbol{y} - \boldsymbol{y}_d && \text{in } \Omega \\ \nabla\cdot\boldsymbol{z} &= 0 && \text{in } \Omega \\ \boldsymbol{z} &= \boldsymbol{0} && \text{on } \Gamma = \partial\Omega. \end{aligned}\right\} \tag{2.4}$$

The evaluation of the gradient $\nabla\widehat{J}'(\boldsymbol{u})$ for a given control function $\boldsymbol{u}$ according to the representation (2.2) is described in Algorithm 2.1.

**Algorithm 2.1** Evaluation of the reduced gradient $\nabla \widehat{J}(\boldsymbol{u})$

```
1: Solve the Navier-Stokes system (1.1) for $(\boldsymbol{y}, p)$
2: Solve the adjoint Navier-Stokes system (2.4) for $(\boldsymbol{z}, q)$
3: Return $\nabla \widehat{J}(\boldsymbol{u}) := \chi_\omega \boldsymbol{z} + \alpha \boldsymbol{u}$
```

## 2.1 Projected Gradient Method

The projected gradient, or steepest descent method, uses the negative reduced gradient as a search direction and then computes a step length in this direction (Algorithm 2.2). Due to the presence of bounds $\boldsymbol{u}_a$ and $\boldsymbol{u}_b$ for the control variable, a projection onto the set of admissible controls is needed, which is given by the cut-off function

$$\mathcal{P}_{[\boldsymbol{u}_a, \boldsymbol{u}_b]}(\boldsymbol{v}) = \max\{\boldsymbol{u}_a, \min\{\boldsymbol{u}_b, \boldsymbol{v}\}\},$$

understood component-wise.

**Algorithm 2.2** Projected gradient algorithm

```
Require: initial guess $\boldsymbol{u}_0$ satisfying $\boldsymbol{u}_a \le \boldsymbol{u}_0 \le \boldsymbol{u}_b$
1: Set $k := 0$
2: while stopping criteria are violated do
3:     Evaluate $\boldsymbol{d}_k := -\nabla \widehat{J}(\boldsymbol{u}_k)$
4:     Obtain step length $t_k$, e.g., from Armijo's rule
5:     Set $\boldsymbol{u}_{k+1} := \mathcal{P}_{[\boldsymbol{u}_a, \boldsymbol{u}_b]}(\boldsymbol{u}_k + t_k\, \boldsymbol{d}_k)$ and increase $k$
6: end while
```

The step length $t_k$ is obtained from a suitable line search strategy such as Armijo's rule: for given parameters $\beta, \sigma \in (0,1)$, we accept the first step length $t \in \{\beta^\ell : \ell = 0, 1, \dots\}$ which satisfies the descent criterion

$$\varphi(t) \le \varphi(0) + \sigma\, t\, \varphi'(0).$$

The line search function $\varphi$ and its derivative are given by

$$\varphi(t) = \widehat{J}\big(\mathcal{P}_{[\boldsymbol{u}_a, \boldsymbol{u}_b]}(\boldsymbol{u}_k + t\, \boldsymbol{d}_k)\big), \qquad \varphi'(0) = \big(\nabla \widehat{J}(\boldsymbol{u}_k), \widetilde{\boldsymbol{d}}_k\big)_\omega,$$

where $\widetilde{\boldsymbol{d}}_k$ is a modification of the negative reduced gradient $\boldsymbol{d}_k$ (in a component-wise sense):

$$\widetilde{\boldsymbol{d}}_k = \begin{cases} \boldsymbol{0} & \text{where } \boldsymbol{u}_k \notin [\boldsymbol{u}_a, \boldsymbol{u}_b] \text{ or } (\boldsymbol{u}_k = \boldsymbol{u}_b \text{ and } \boldsymbol{d}_k \ge \boldsymbol{0}) \text{ or } (\boldsymbol{u}_k = \boldsymbol{u}_a \text{ and } \boldsymbol{d}_k \le \boldsymbol{0}) \\ \boldsymbol{d}_k & \text{elsewhere.} \end{cases}$$

A typical feature of gradient based methods such as Algorithm 2.2 is their good progress in reducing the objective in the initial iterations, while they slow down significantly in later iterations.

## 2.2 Nonlinear Conjugate Gradient Method

The conjugate gradient (CG) method is known as an iterative method with attractive convergence properties for the solution of linear systems of equations with symmetric and positive definite coefficient matrix. A nonlinear version can also be employed to find minimizers of unconstrained optimization problems. In this section, we neglect the control bounds $\boldsymbol{u}_a$ and $\boldsymbol{u}_b$ in (ROCP$_{\text{cc}}$).

A nonlinear CG method is stated as Algorithm 2.3. In step 4, a search procedure is needed to determine an appropriate step length which minimizes $\varphi(t) = \widehat{J}(\boldsymbol{u}_k + t\,\boldsymbol{d}_k)$, or finds a zero of $\varphi'(t) = \big(\nabla\widehat{J}(\boldsymbol{u}_k + t\,\boldsymbol{d}_k), \boldsymbol{d}_k\big)_\omega$. In principle, Newton's method can be used for this purpose, but it requires the repeated evaluation of the Hessian of $\widehat{J}$, which is prohibitive in the context of nonlinear CG methods (but see Section 3.1). As a remedy, one can resort to a secant method in which $\varphi(t)$ is approximated by

$$\varphi(t) \approx \varphi(0) + \varphi'(0)\,t + \frac{\varphi'(\sigma) - \varphi'(0)}{2\sigma}\,t^2$$

with some $\sigma > 0$. Minimization of the right hand side then leads to

$$t = -\frac{\varphi'(0)}{\varphi'(\sigma) - \varphi'(0)} = -\frac{\big(\nabla\widehat{J}(\boldsymbol{u}_k), \boldsymbol{d}_k\big)_\omega}{\big(\nabla\widehat{J}(\boldsymbol{u}_k) + \sigma\,\boldsymbol{d}_k, \boldsymbol{d}_k\big)_\omega - \big(\nabla\widehat{J}(\boldsymbol{u}_k), \boldsymbol{d}_k\big)_\omega}. \tag{2.5}$$

Typically, only few iterations of (2.5) are carried out which generate a sequence of step lengths $t_i$ and linearization points $\sigma_{i+1} = -t_i$, starting from an arbitrary initial value $\sigma_0 > 0$. Such inexact line searches, however, may lead to search directions which are not descent directions for the objective $\widehat{J}$. A common solution is to restart the method by setting $\boldsymbol{d}$ to the negative reduced gradient whenever $(\boldsymbol{r}, \boldsymbol{d})_\omega \le 0$ holds. This is the purpose of step 8 in Algorithm 2.3. An alternative step length selection strategy in step 4 based on Wolfe conditions is given in [3, eq. (5.42)].

**Algorithm 2.3** Nonlinear conjugate gradient algorithm

```
Require: initial guess $\boldsymbol{u}_0$
 1: Set $k := 0$
 2: Evaluate $\boldsymbol{d}_k := \boldsymbol{r}_k := -\nabla\widehat{J}(\boldsymbol{u}_k)$
 3: while stopping criteria are violated do
 4:     Obtain step length $t_k$ satisfying $\big(\nabla\widehat{J}(\boldsymbol{u}_k + t_k\,\boldsymbol{d}_k), \boldsymbol{d}_k\big)_\omega = 0$
 5:     Set $\boldsymbol{u}_{k+1} := \boldsymbol{u}_k + t_k\,\boldsymbol{d}_k$
 6:     Set $\boldsymbol{r}_{k+1} := -\nabla\widehat{J}(\boldsymbol{u}_{k+1})$
 7:     Determine step length $\beta_{k+1}$
 8:     Set $\boldsymbol{d}_{k+1} := \boldsymbol{r}_{k+1} + \beta_{k+1}\,\boldsymbol{d}_k$ and increase $k$
 9:     if $(\boldsymbol{r}_k, \boldsymbol{d}_k)_\omega \le 0$ then
10:         Set $\boldsymbol{d}_k := \boldsymbol{r}_k$
11:     end if
12: end while
```

Several common choices exist for the selection of the step length $\beta_{k+1}$ in step 7. Among them are the Fletcher-Reeves and the Polak-Ribière formulas

$$\beta_{k+1}^{\text{FR}} := \frac{(\boldsymbol{r}_{k+1}, \boldsymbol{r}_{k+1})_\omega}{(\boldsymbol{r}_k, \boldsymbol{r}_k)_\omega}, \quad \beta_{k+1}^{\text{PR}} := \frac{(\boldsymbol{r}_{k+1}, \boldsymbol{r}_{k+1} - \boldsymbol{r}_k)_\omega}{(\boldsymbol{r}_k, \boldsymbol{r}_k)_\omega}, \quad \beta_{k+1}^{\text{PR+}} := \max\{\beta_{k+1}^{\text{PR}}, 0\}.$$

We mention that nonlinear CG methods generally outperform the steepest descent method, and refer to [3] for a comparison of the various step length selection strategies. For an application of nonlinear conjugate gradient methods for the solution of optimal control problems, we refer to, e.g., [10, 11].

## 3 Higher-Order Methods

The methods presented previously in Section 2 typically exhibit only first order convergence because they solely rely on first order (gradient) information about the objective function. As a remedy, one resorts to methods which employ higher-order approximations.

### 3.1 Newton's Method

Newton's method falls into the category of black-box methods. In the absence of control constraints, a necessary optimality condition for (ROCP$_{\text{cc}}$) is that the gradient $\nabla\widehat{J}(\boldsymbol{u})$ vanishes. The application of Newton's method then results in the iteration

$$\nabla^2\widehat{J}(\boldsymbol{u}_k)\,\delta\boldsymbol{u} = -\nabla\widehat{J}(\boldsymbol{u}_k), \qquad \boldsymbol{u}_{k+1} := \boldsymbol{u}_k + \delta\boldsymbol{u}, \tag{3.1}$$

where $\nabla^2\widehat{J}$ denotes the Hessian of the reduced objective (the reduced Hessian).

The reduced Hessian matrix is usually not formed explicitly due to the tremendous computational effort to do so. By contrast, (3.1) is solved iteratively, using a Krylov method such as MINRES or CG which take advantage of the symmetry of $\nabla^2\widehat{J}(\boldsymbol{u}_k)$. Every iteration then requires the evaluation of one matrix-vector product $\nabla^2\widehat{J}(\boldsymbol{u})\,\delta\boldsymbol{u}$. Algorithm 3.1 describes how to achieve this.

Quasi-Newton methods, such as BFGS, offer an alternative to the exact evaluation of the Hessian matrix. By constrast, they store and accumulate gradient information from iteration to iteration as a substitute for second derivatives. Due to the high dimension of discretized optimal control problems, limited-memory versions such as LM-BFGS, should be employed.

**Algorithm 3.1** Evaluation of the reduced Hessian times a vector $\nabla^2\widehat{J}(\boldsymbol{u})\,\delta\boldsymbol{u}$

```
Require: $\boldsymbol{y}$ is the velocity obtained from (1.1), and $\boldsymbol{z}$ is the adjoint velocity obtained from (2.4)
1: Solve the linearized Navier-Stokes system (2.3) for $(\delta\boldsymbol{y}, \delta p)$
2: Calculate $\delta\boldsymbol{r} = \delta\boldsymbol{y} + (\nabla\delta\boldsymbol{y})^\top \boldsymbol{z} - (\delta\boldsymbol{y}\cdot\nabla)\,\boldsymbol{z}$
3: Solve the adjoint Navier-Stokes system (2.4) for $(\delta\boldsymbol{z}, \delta q)$ with right hand side $\delta\boldsymbol{r}$ instead of $\boldsymbol{y} - \boldsymbol{y}_d$
4: Return $\nabla^2\widehat{J}(\boldsymbol{u})\,\delta\boldsymbol{u} := \chi_\omega\delta\boldsymbol{z} + \alpha\,\delta\boldsymbol{u}$
```

In the presence of control constraints $\boldsymbol{u}_a \le \boldsymbol{u} \le \boldsymbol{u}_b$, the condition $\nabla\widehat{J}(\boldsymbol{u}) = 0$ must be replaced by the variational inequality $(\nabla\widehat{J}(\boldsymbol{u}), \boldsymbol{v} - \boldsymbol{u})_\omega \ge 0$ for all admissible $\boldsymbol{v}$ (satisfying the same inequalities as $\boldsymbol{u}$ does). An alternative representation of this condition uses Lagrange multipliers and takes the following form:

$$\nabla \widehat{J}(\boldsymbol{u}) + \boldsymbol{\mu}_b - \boldsymbol{\mu}_a = \mathbf{0} \tag{3.2a}$$

$$\left.\begin{aligned} &\boldsymbol{\mu}_a \ge \mathbf{0}, \quad \boldsymbol{u}_a - \boldsymbol{u} \le \mathbf{0}, \quad \boldsymbol{\mu}_a^\top(\boldsymbol{u}_a - \boldsymbol{u}) = 0 \\ &\boldsymbol{\mu}_b \ge \mathbf{0}, \quad \boldsymbol{u} - \boldsymbol{u}_b \le \mathbf{0}, \quad \boldsymbol{\mu}_b^\top(\boldsymbol{u} - \boldsymbol{u}_b) = 0. \end{aligned}\right\} \tag{3.2b}$$

All of the above have to be understood in a pointwise almost everywhere sense in $\omega$. For an algorithmic treatment, it is useful to combine the Lagrange multipliers into one by setting $\boldsymbol{\mu} = \boldsymbol{\mu}_b - \boldsymbol{\mu}_a$, and to convert the complementarity conditions (3.2b) into one equation, which results in

$$\nabla \widehat{J}(\boldsymbol{u}) + \boldsymbol{\mu} = \mathbf{0} \tag{3.3a}$$

$$\max\{\mathbf{0}, \boldsymbol{\mu} + c\,(\boldsymbol{u} - \boldsymbol{u}_b)\} + \min\{\mathbf{0}, \boldsymbol{\mu} + c\,(\boldsymbol{u} - \boldsymbol{u}_a)\} - \boldsymbol{\mu} = \mathbf{0}. \tag{3.3b}$$

Note that (3.3) is equivalent to (3.2) for any value of $c > 0$, and $\max$ and $\min$ have to be understood again in a component-wise and pointwise sense.

Although equation (3.3b) is not differentiable in the classical sense, it enjoys a property called Newton differentiability in the case $c = \alpha$. Thus, a so-called semi-smooth Newton method can be applied and it is well posed, not only in a discretized setting but also in function space, see [12–14]. Other choices $c \neq \alpha$ are justified as well [15]. Naturally, the Jacobian of (3.3) depends on the subsets of the control domain $\omega$ where the $\max$ and $\min$ is attained by either of the two expressions. These so-called active and inactive sets at an iterate $(\boldsymbol{u}_k, \boldsymbol{\mu}_k)$ are defined as

$$\begin{aligned} \mathcal{A}_k^{+,\ell} &:= \{x \in \omega : \mu_k^\ell + c\,(u_k^\ell - u_b^\ell) > 0\}, \quad \mathcal{A}_k^\ell := \mathcal{A}_k^{+,\ell} \cup \mathcal{A}_k^{-,\ell} \\ \mathcal{A}_k^{-,\ell} &:= \{x \in \omega : \mu_k^\ell + c\,(u_k^\ell - u_a^\ell) < 0\}, \quad \mathcal{I}_k^\ell := \omega \setminus \mathcal{A}_k^\ell. \end{aligned} \tag{3.4}$$

Here, $u^\ell$ denotes ones component of the velocity vector $\boldsymbol{u}$, with $\ell = 1, \ldots, d$. Due to the dependence of the active sets on both primal and dual variables $\boldsymbol{u}$ and $\boldsymbol{\mu}$, the resulting algorithm (Algorithm 3.2) is called a primal-dual active set method.

The Newton direction for (3.3) at an iterate $(\boldsymbol{u}_k, \boldsymbol{\mu}_k)$ is found to be

$$\begin{bmatrix} \nabla^2 \widehat{J}(\boldsymbol{u}_k) & I \\ c\,\chi_{\mathcal{A}_k} & -\chi_{\mathcal{I}_k} \end{bmatrix} \begin{pmatrix} \delta \boldsymbol{u} \\ \delta \boldsymbol{\mu} \end{pmatrix} = - \begin{pmatrix} \nabla \widehat{J}(\boldsymbol{u}_k) + \boldsymbol{\mu}_k \\ c\,\chi_{\mathcal{A}_k^+}(\boldsymbol{u}_k - \boldsymbol{u}_b) + c\,\chi_{\mathcal{A}_k^-}(\boldsymbol{u}_k - \boldsymbol{u}_a) - \chi_{\mathcal{I}_k}\boldsymbol{\mu}_k \end{pmatrix}, \tag{3.5}$$

with the appropriate component-wise interpretation of $\mathcal{A}_k^\pm$ etc. Equation (3.5) is solved iteratively, and matrix-vector products with the matrix in (3.5) can be easily formed using Algorithm 3.1. For algorithmic purposes, it is beneficial to convert (3.5) into its equivalent symmetric form,

$$\begin{bmatrix} \nabla^2 \widehat{J}(\boldsymbol{u}_k) & \chi_{\mathcal{A}_k} \\ \chi_{\mathcal{A}_k} & 0 \end{bmatrix} \begin{pmatrix} \delta \boldsymbol{u} \\ \delta \boldsymbol{\mu}_{|\mathcal{A}_k} \end{pmatrix} = - \begin{pmatrix} \nabla \widehat{J}(\boldsymbol{u}_k) + \chi_{\mathcal{A}_k}\boldsymbol{\mu}_k \\ \chi_{\mathcal{A}_k^+}(\boldsymbol{u}_k - \boldsymbol{u}_b) + \chi_{\mathcal{A}_k^-}(\boldsymbol{u}_k - \boldsymbol{u}_a) \end{pmatrix}, \tag{3.6}$$

where $\delta \boldsymbol{\mu}_{|\mathcal{A}_k}$ denotes the restriction of $\delta \boldsymbol{\mu}$ to the active set, and $\boldsymbol{\mu}_{k+1}$ is set to zero on the inactive set. We summarize the resulting algorithm as Algorithm 3.2.

**Algorithm 3.2** Newton's method with primal-dual active set strategy

```
Require: initial guess $\boldsymbol{u}_0$ and $\boldsymbol{\mu}_0$
1: Set $k := 0$
2: while stopping criteria are violated do
3:     Evaluate the reduced gradient $\nabla \widehat{J}(\boldsymbol{u}_k)$ (Algorithm 2.1)
4:     Compute the active sets $\mathcal{A}_k^{\pm,\ell}$ for $\ell = 1, \ldots, d$
5:     Solve the Newton system (3.6) iteratively
6:     Set $\boldsymbol{u}_{k+1} := \boldsymbol{u}_k + \delta\boldsymbol{u}$, $\boldsymbol{\mu}_{k+1|\mathcal{A}_k} := \boldsymbol{\mu}_{k|\mathcal{A}_k} + \delta\boldsymbol{\mu}_{|\mathcal{A}_k}$, $\boldsymbol{\mu}_{k+1|\mathcal{I}_k} := \boldsymbol{0}$ and increase $k$
7: end while
```

In a practical implementation, the Hessian $\nabla^2 \widehat{J}(\boldsymbol{u})$ in (3.6) is often replaced by a quasi-Newton approximation which was already mentioned above. Moreover, it may be necessary to globalize Algorithm 3.2 in order to achieve convergence from arbitrary starting points. This can be done, for instance, by carrying out a few gradient steps (Algorithm 2.2) beforehand.

Interior point methods offer an alternative way in dealing with the complementarity system (3.3b). We refer to [16, 17] for their application to optimal control problems with control constraints, and to [18] for a comparison with Algorithm 3.2.

### 3.2 Sequential Quadratic Programming Methods

In contrast to Newton's method, sequential quadratic programming (SQP) methods treat the design and state variables as independent optimization variables and thus they fall into the category of all-at-once methods. Their explanation requires additional notation. From an abstract point of view, our problem (OCP$_{\text{cc}}$) is an optimization problem with equality constraints (the Navier-Stokes system) and inequality constraints for the control variable, i.e.,

$$\text{Minimize} \quad J(\boldsymbol{y}, \boldsymbol{u}) \quad \text{subject to} \quad e(\boldsymbol{y}, \boldsymbol{u}) = 0 \quad \text{and} \quad \boldsymbol{u}_a \le \boldsymbol{u} \le \boldsymbol{u}_b. \tag{3.7}$$

Here $e(\boldsymbol{y}, \boldsymbol{u}) = 0$ denotes the equality constraint given by the Navier Stokes equations (1.1). To avoid notational burden, we take the point of view that the divergence-free and the boundary condition are incorporated into a solenoidal space $\boldsymbol{Y}$, so that

$$e(\boldsymbol{y}, \boldsymbol{u}) = -\nu \triangle \delta \boldsymbol{y} + (\boldsymbol{y} \cdot \nabla)\, \boldsymbol{y} - \chi_\omega \boldsymbol{u}.$$

We can take $\boldsymbol{Y} = \{\boldsymbol{y} \in (H_0^1(\Omega))^d : \nabla \cdot \boldsymbol{y} = 0\}$ and denote by $\boldsymbol{Y}^*$ its dual. With $\boldsymbol{z} \in \boldsymbol{Y}^*$ we introduce the Lagrangian associated with (3.7) as

$$\mathcal{L}(\boldsymbol{y}, \boldsymbol{u}, \boldsymbol{z}) = J(\boldsymbol{y}, \boldsymbol{u}) + \langle e(\boldsymbol{y}, \boldsymbol{u}), \boldsymbol{z} \rangle_{\boldsymbol{Y}^*, \boldsymbol{Y}}.$$

SQP methods solve in every iteration a quadratic programming problem, obtained by building a quadratic approximation of the Lagrangian and by linearizing the equality and inequality constraints:

$$\begin{aligned} \text{Minimize} \quad & \frac{1}{2} \begin{pmatrix} \boldsymbol{y} - \boldsymbol{y}_k \\ \boldsymbol{u} - \boldsymbol{u}_k \end{pmatrix}^\top \begin{bmatrix} \mathcal{L}_{\boldsymbol{yy}} & \mathcal{L}_{\boldsymbol{yu}} \\ \mathcal{L}_{\boldsymbol{uy}} & \mathcal{L}_{\boldsymbol{uu}} \end{bmatrix} \begin{pmatrix} \boldsymbol{y} - \boldsymbol{y}_k \\ \boldsymbol{u} - \boldsymbol{u}_k \end{pmatrix} \\ & + J_{\boldsymbol{y}}(\boldsymbol{y}_k, \boldsymbol{u}_k)(\boldsymbol{y} - \boldsymbol{y}_k) + J_{\boldsymbol{u}}(\boldsymbol{y}_k, \boldsymbol{u}_k)(\boldsymbol{u} - \boldsymbol{u}_k) \\ \text{subject to} \quad & e_{\boldsymbol{y}}(\boldsymbol{y}_k, \boldsymbol{u}_k)(\boldsymbol{y} - \boldsymbol{y}_k) + e_{\boldsymbol{u}}(\boldsymbol{y}_k, \boldsymbol{u}_k)(\boldsymbol{u} - \boldsymbol{u}_k) + e(\boldsymbol{y}_k, \boldsymbol{u}_k) = \boldsymbol{0} \\ \text{and} \quad & \boldsymbol{u}_a \le \boldsymbol{u} \le \boldsymbol{u}_b. \end{aligned} \tag{3.8}$$

The Hessian of the Lagrangian is evaluated at the current iterate $(\boldsymbol{y}_k, \boldsymbol{u}_k, \boldsymbol{z}_k)$, and the solution $(\boldsymbol{y}, \boldsymbol{u})$ and the adjoint state $\boldsymbol{z}$ for (3.8), i.e. the Lagrange multiplier associated with the equality constraint in (3.8), serve as subsequent iterates $(\boldsymbol{y}_{k+1}, \boldsymbol{u}_{k+1}, \boldsymbol{z}_{k+1})$.

Due to the presence of inequality constraints in (3.8), its optimality system becomes nonlinear, and it cannot be solved in one go. We describe here the solution of (3.8) by a semi-smooth Newton (primal-dual active set) method and we use $j$ to denote the iterations with respect to this inner loop. At a given iterate $(\boldsymbol{y}_{k+1,j}, \boldsymbol{u}_{k+1,j}, \boldsymbol{z}_{k+1,j})$, a Newton step amounts to finding the active and inactive sets

$$\begin{aligned} \mathcal{A}_j^{+,\ell} &:= \{x \in \omega : \mu_j^\ell + c\,(u_j^\ell - u_b^\ell) > 0\}, \quad & \mathcal{A}_j^\ell &:= \mathcal{A}_j^{+,\ell} \cup \mathcal{A}_j^{-,\ell} \\ \mathcal{A}_j^{-,\ell} &:= \{x \in \omega : \mu_j^\ell + c\,(u_j^\ell - u_a^\ell) < 0\}, \quad & \mathcal{I}_j^\ell &:= \omega \setminus \mathcal{A}_j^\ell, \end{aligned}$$

for $\ell = 1, \ldots, d$, compare (3.4), and then solving

$$\begin{bmatrix} \mathcal{L}_{\boldsymbol{yy}} & \mathcal{L}_{\boldsymbol{yu}} & e_{\boldsymbol{y}}^\star & 0 \\ \mathcal{L}_{\boldsymbol{uy}} & \mathcal{L}_{\boldsymbol{uu}} & e_{\boldsymbol{u}}^\star & \chi_{\mathcal{A}_j} \\ e_{\boldsymbol{y}} & e_{\boldsymbol{u}} & 0 & 0 \\ 0 & \chi_{\mathcal{A}_j} & 0 & 0 \end{bmatrix} \begin{pmatrix} \boldsymbol{y}_{k+1,j+1} - \boldsymbol{y}_k \\ \boldsymbol{u}_{k+1,j+1} - \boldsymbol{u}_k \\ \boldsymbol{z}_{k+1,j+1} \\ \boldsymbol{\mu}_{j+1|\mathcal{A}_j} \end{pmatrix} = - \begin{pmatrix} J_{\boldsymbol{y}}(\boldsymbol{y}_k, \boldsymbol{u}_k) \\ J_{\boldsymbol{u}}(\boldsymbol{y}_k, \boldsymbol{u}_k) \\ e(\boldsymbol{y}_k, \boldsymbol{u}_k) \\ \chi_{\mathcal{A}_j^+}(\boldsymbol{u}_k - \boldsymbol{u}_b) + \chi_{\mathcal{A}_j^-}(\boldsymbol{u}_k - \boldsymbol{u}_a) \end{pmatrix}. \tag{3.9}$$

All terms in the left hand side matrix are evaluated at $(\boldsymbol{y}_k, \boldsymbol{u}_k, \boldsymbol{z}_k)$. For convenience of the reader, we summarize the basic SQP algorithm as Algorithm 3.3.

**Algorithm 3.3** Basic SQP method with primal-dual active set strategy

```
Require: initial guess $\boldsymbol{u}_0$
1: Set $k := 0$
2: while stopping criteria are violated do
3:     Set $j := 0$
4:     while stopping criteria are violated do
5:         Compute the active sets $\mathcal{A}_j^{\pm,\ell}$ for $\ell = 1, \ldots, d$
6:         Solve the Newton system (3.9) iteratively
7:         Set $\boldsymbol{\mu}_{j+1|\mathcal{I}_j} := \mathbf{0}$ and increase $j$
8:     end while
9:     Set $\boldsymbol{y}_{k+1} := \boldsymbol{y}_{k+1,j}$,  $\boldsymbol{u}_{k+1} := \boldsymbol{u}_{k+1,j}$,  $\boldsymbol{z}_{k+1} := \boldsymbol{z}_{k+1,j}$ and increase $k$
10: end while
```

There are many similarities between the steps (3.6) and (3.9) in the Newton and SQP methods. In fact, the matrix in (3.9) can be symmetrically reduced to the one in (3.6) in view of

$$\begin{bmatrix} \nabla^2 \widehat{J} & \chi_{\mathcal{A}} \\ \chi_{\mathcal{A}} & 0 \end{bmatrix} = \begin{bmatrix} -e_{\boldsymbol{u}}^\star e_{\boldsymbol{y}}^{-\star} & I & 0 & 0 \\ 0 & 0 & 0 & I \end{bmatrix} \begin{bmatrix} \mathcal{L}_{\boldsymbol{yy}} & \mathcal{L}_{\boldsymbol{yu}} & e_{\boldsymbol{y}}^\star & 0 \\ \mathcal{L}_{\boldsymbol{uy}} & \mathcal{L}_{\boldsymbol{uu}} & e_{\boldsymbol{u}}^\star & \chi_{\mathcal{A}} \\ e_{\boldsymbol{y}} & e_{\boldsymbol{u}} & 0 & 0 \\ 0 & \chi_{\mathcal{A}} & 0 & 0 \end{bmatrix} \begin{bmatrix} -e_{\boldsymbol{y}}^{-1} e_{\boldsymbol{u}} & 0 \\ I & 0 \\ 0 & 0 \\ 0 & I \end{bmatrix}.$$

Note that $\begin{bmatrix} -e_{\boldsymbol{y}}^{-1} e_{\boldsymbol{u}} \\ I \end{bmatrix}$ is the projection onto the kernel of the linearization of $e(\boldsymbol{y}, \boldsymbol{u})$. The main difference between Algorithm 3.2 and 3.3 is that the iterates of the former satisfy the nonlinear

Navier-Stokes system (1.1), while the iterates of the latter satisfy the *linearized* system as can be seen from the third equation in (3.9). The nonlinear Navier-Stokes system is satisified only in the limit.

**Remark 3.1** (Newton vs. SQP for time-dependent problems) One may argue that the solution of the nonlinear Navier-Stokes system in every step of Newton's method (Algorithm 3.2) is a disadvantage compared to the SQP Algorithm 3.3, which requires only linearized equations to be solved. However, this gap can be closed for time-dependent problems. In this case, the nonlinearity of the Navier-Stokes equation may be treated as an explicit term in the time-stepping routine. As a consequence, the effort for solving the nonlinear and linearized Navier-Stokes system become essentially the same. For a discussion of the SQP method in the case of time-dependent problems, we refer, e.g., to [19].

**Remark 3.2** (Preconditioning) Both the Newton and SQP approaches presented in the previous sections require the repeated solution of linear systems (3.6) and (3.9), respectively. These systems feature a saddle point structure, and they are inherently large scale and ill-conditioned. Their efficient solution thus calls for preconditioned iterative solvers. Due to space restrictions, we refer to [20–27] for more on the issue of preconditioning.

## 4 State Constraints and Alternative Objective Functionals

In this section we comment on extensions of previously considered problems in two directions. We first consider a prototypical example for a problem involving state constraints:

$$\begin{aligned}&\text{Minimize} \quad J(\boldsymbol{y},\boldsymbol{u}) := \frac{1}{2}\int_\Omega |\boldsymbol{y}-\boldsymbol{y}_d|^2\,\mathrm{d}x + \frac{\alpha}{2}\int_\omega |\boldsymbol{u}|^2\,\mathrm{d}x \\ &\text{subject to} \quad (1.1) \qquad \text{and} \quad M\boldsymbol{y} \le \boldsymbol{\psi} \quad \text{in } \Omega,\end{aligned} \tag{OCP$_\text{sc}$}$$

where $\boldsymbol{\psi} \in \mathbb{R}^m$, $M \in \mathbb{R}^{m\times d}$ and the inequality is considered in the almost everywhere sense. We may be tempted to treat these inequalities in a similar manner done for the control constraints in Algorithm 3.2. Upon discretization, the resulting algorithm, however, would not have the same, mesh-independent super-linear convergence properties as in the control constrained case. This is due to lack of Newton differentiability of the underlying optimality system. As a consequence we introduce the family of regularized problems

$$\min J(\boldsymbol{y},\boldsymbol{u}) + \frac{\gamma}{2}\|(M\boldsymbol{y}-\boldsymbol{\psi})^+\|^2 \quad \text{such that (1.1) holds,} \tag{OCP$_{\text{sc}_\gamma}$}$$

where $\gamma > 0$, and $(M\boldsymbol{y}-\boldsymbol{\psi})^+$ is defined coordinate-wise: $(M\boldsymbol{y}-\boldsymbol{\psi})^+_i = \max\{M_i^\top \boldsymbol{y} - \boldsymbol{\psi}_i, 0\}$, for $i = 1,\dots,m$. The optimality system for (OCP$_{\text{sc}_\gamma}$) is given by

$$\left.\begin{aligned}&-\nu\triangle\boldsymbol{y} + (\boldsymbol{y}\cdot\nabla)\,\boldsymbol{y} + \nabla p = \chi_\omega \boldsymbol{u} \text{ in } \Omega\\ &\nabla\cdot\boldsymbol{y} = 0 \text{ in } \Omega, \qquad \boldsymbol{y} = \boldsymbol{0} \text{ on } \partial\Omega\\[1ex] &-\nu\triangle\boldsymbol{z} + (\nabla\boldsymbol{y})^\top\boldsymbol{z} - (\boldsymbol{y}\cdot\nabla)\,\boldsymbol{z} + \nabla q + M^\top\boldsymbol{\lambda} = \boldsymbol{y}-\boldsymbol{y}_d \text{ in } \Omega\\ &\nabla\cdot\boldsymbol{z} = 0 \text{ in } \Omega, \qquad \boldsymbol{z} = \boldsymbol{0} \text{ on } \partial\Omega\\[1ex] &\alpha\boldsymbol{u} = \boldsymbol{z}\chi_\omega, \qquad \boldsymbol{\lambda} = \gamma\,(M\boldsymbol{y}-\boldsymbol{\psi})^+.\end{aligned}\right\} \tag{4.1}$$

Considering $(\boldsymbol{y}, \boldsymbol{z})$ as a function of $(\boldsymbol{u}, \boldsymbol{\lambda})$ this optimality system can be expressed as $F(\boldsymbol{u}, \boldsymbol{\lambda}) = \big(\alpha \boldsymbol{u} - \boldsymbol{z} \chi_\omega,\ \boldsymbol{\lambda} - \gamma\,(M\boldsymbol{y} - \boldsymbol{\psi})^+\big) = 0$. Interpreting the equation $\boldsymbol{\lambda} = \gamma\,(M\boldsymbol{y} - \boldsymbol{\psi})^+$ in $L^2(\Omega)$ we note that the $\max$ operation acts on the variable $\boldsymbol{y}$ which is in $H^1(\Omega)$. This extra regularity is necessary to argue that the mapping $F$ is Newton differentiable so that the semi-sooth Newton algorithm applied to (4.1) is locally super-linearly convergent, provided that the viscosity $\nu$ is large enough. For elliptic systems this argument with $d = 2$ was carried out in detail in [28].

The semi-smooth Newton step amounts to a Newton step on the system (4.1) where the generalized derivative $D\widetilde{F}(\boldsymbol{y})$ of $\boldsymbol{y} \mapsto \widetilde{F}(\boldsymbol{y}) = (M\boldsymbol{y} - \boldsymbol{\psi})^+$ at $\boldsymbol{y}$ is determined according to

$$D\,\widetilde{F}(\boldsymbol{y})\,\delta\boldsymbol{y} = \begin{pmatrix} M_1\chi_{A_1} \\ \vdots \\ M_m\chi_{A_m} \end{pmatrix} \delta\boldsymbol{y} - \begin{pmatrix} \boldsymbol{\psi}_1\chi_{A_1} \\ \vdots \\ \boldsymbol{\psi}_m\chi_{A_m} \end{pmatrix} \text{ with } M = \begin{pmatrix} M_1 \\ \vdots \\ M_m \end{pmatrix}, \quad \boldsymbol{\psi} = \begin{pmatrix} \boldsymbol{\psi}_1 \\ \vdots \\ \boldsymbol{\psi}_m \end{pmatrix},$$

$A_i = \{x : M_i\,\boldsymbol{y}(x) > \boldsymbol{\psi}_i(x)\}$ and $\chi_{A_i}$ is the characteristic function of $A_i$. A chain-rule argument then allows to determine the Newton derivative of $F$.

Let us now comment on the choice of the cost functional. In (OCP$_{\text{cc}}$) we considered a quadratic tracking functional with quadratic cost for the control. Obvious advantages of quadratic functional include the simplicity of obtaining derivatives and the stochastic interpretation in case of linear dynamics in terms of the Gaussian regulator problem. Recently attention was given to $L^1(\Omega)$ type costs, i.e., functionals including the term $\int_\omega |\boldsymbol{u}|\,\mathrm{d}x$. This cost is attractive since it can be interpreted as being proportional to the actual control action. Moreover the optimal $L^1(\Omega)$ cost typically has a sparse support which suggests the use of $L^1(\Omega)$ formulations for the determination of optimal actuator placement strategies. Disadvantages include the lack of differentiability and the fact that without further modifications $L^1(\Omega)$ formulations do not admit solutions, in general. One has to either employ some type of regularization or use a problem formulation where the controls are measures, see [29–31].

Next we turn our attention to that part of the cost functional which involves the state $\boldsymbol{y}$ and consider the time-dependent version of the state equation as in (1.1) with some additional forcing either in the interior of the domain or on its boundary, which creates vortices in the fluid. The control objective consists in suppressing these vortices. In the context of the quadratic cost in (OCP$_{\text{cc}}$) one may consider to choose $\boldsymbol{y}_d$ as the solution of the Stokes equation with the same forcing. Then $\boldsymbol{y}_d$ will not contain vortices, but otherwise retain some of the global features of the forced solution $\boldsymbol{y}$. One of the objections against this functional is that is not Galilean invariant, i.e. it is not invariant under transformations of the form $\mathcal{Q}\,x + d\,t$ of the flow field $\boldsymbol{y}$, where $t$ denotes time, $\mathcal{Q}$ is a time-independent matrix and $d$ is a constant vector. Another frequently used functional is

$$\int_0^T \int_\Omega |\mathrm{curl}\,\boldsymbol{y}(t,x)|^2 \,\mathrm{d}x\,\mathrm{d}t.$$

Just as the tracking functional this functional is again not Galilean invariant. Another shortcoming of using curl $\boldsymbol{y}$ for measuring vorticity is given by the fact that it does identify vortex cores in shear flow, especially if the background shear is comparable to the vorticity within the vortex.

In dimension two a Galilean invariant quantity will determine a region to be a vortex if $\nabla \boldsymbol{y}(x)$ has complex eigenvalues, or equivalently if $\det \nabla y(x) > 0$. This suggests the use of

$$\int_0^T \int_\Omega \max\{0, \det \nabla \boldsymbol{y}(t,x)\} \, \mathrm{d}x \, \mathrm{d}t$$

for vortex suppression and raises interesting questions concerning efficient algorithmic treatment of this cost functional. For a discussion of these issues we refer to [32].

## References

[1] P. A. Davidson, An Introduction to Magnetohydrodynamics (Cambridge University Press, Cambridge, 2001).

[2] R. Griesse and K. Kunisch, SIAM Journal on Control and Optimization **45**(5), 1822–1845 (2006).

[3] J. Nocedal and S. Wright, Numerical Optimization (Springer, New York, 1999).

[4] R. Fletcher, Practical methods of optimization, second edition (Wiley-Interscience [John Wiley & Sons], New York, 2001).

[5] M. Hinze and K. Kunisch, SIAM Journal on Control and Optimization **40**(3), 925–946 (2001).

[6] M. Gunzburger, Perspectives in Flow Control and Optimization, Advances in Design and Control (SIAM, Philadelphia, 2003).

[7] M. Gunzburger and S. Manservisi, SIAM Journal on Control and Optimization **37**(6), 1913–1945 (1999).

[8] F. Abergel and R. Temam, Theoretical and Computational Fluid Mechanics **1**(6), 303–325 (1990).

[9] V. Girault and P. A. Raviart, Finite Element Methods for Navier-Stokes Equations (Springer, 1986).

[10] S. Volkwein, Optimization Methods and Software **18**, 179–199 (2004).

[11] S. Volkwein, Optimal control of laser surface hardening by utilizing a nonlinear primal-dual active set strategy, Report No. 277, Special Research Center F003, Project Area II: Continuous Optimization and Control, University of Graz & Technical University of Graz, Austria, 2003.

[12] M. Hintermüller, K. Ito, and K. Kunisch, SIAM Journal on Optimization **13**(3), 865–888 (2002).

[13] M. Ulbrich, SIAM Journal on Control and Optimization **13**(3), 805–842 (2003).

[14] M. Ulbrich, Systems and Control Letters **48**, 297–311 (2003).

[15] M. Bergounioux, K. Ito, and K. Kunisch, SIAM Journal on Control and Optimization **37**(4), 1176–1194 (1999).

[16] M. Weiser, SIAM Journal on Control and Optimization **44**(5), 1766–1786 (2005).

[17] M. Weiser, T. Gänzler, and A. Schiela, Computational Optimization and Applications **41**, 127–145 (2008).

[18] M. Bergounioux, M. Haddou, M. Hintermüller, and K. Kunisch, SIAM Journal on Optimization **11**(2), 495–521 (2000).

[19] M. Hintermüller and M. Hinze, SIAM Journal on Optimization **16**(4), 1177–1200 (2006).

[20] M. Hintermüller, I. Kopacka, and S. Volkwein, ESAIM: Control, Optimisation and Calculus of Variations **15**(3) (2009).

[21] A. Battermann and M. Heinkenschloss, Preconditioners for Karush-Kuhn-Tucker Matrices Arising in the Optimal Control of Distributed Systems, in: Optimal Control of Partial Differential Equations, edited by W. Desch, F. Kappel, and K. Kunisch, International Series of Numerical Mathematics Vol. 126 (Birkhäuser, Basel, 1998), pp. 15–32.

[22] A. Battermann and E. W. Sachs, Block preconditioners for KKT systems in PDE-governed optimal control problems, in: Fast solution of discretized optimization problems (Berlin, 2000), edited by K. H. Hoffmann, R. Hoppe, and V. Schulz, International Series of Numerical Mathematics Vol. 138 (Birkhäuser, Basel, 2001), pp. 1–18.

[23] G. Biros and O. Ghattas, SIAM Journal on Scientific Computing **27**(2), 687–713 (2005).
[24] K. Ito, K. Kunisch, I. Gherman, and V. Schulz, SIAM Journal on Matrix Analysis and Applications **31**(4), 1835–1847 (2010).
[25] T. Mathew, M. Sarkis, and C. Schaerer, Numerical Linear Algebra with Applications **14**, 257–279 (2007).
[26] T. Rees and M. Stoll, Numerical Linear Algebra with Applications (to appear), http://dx.doi.org/10.1002/nla.693.
[27] R. Herzog and E. Sachs, SIAM Journal on Matrix Analysis and Applications **31**(5) (2010).
[28] K. Kunisch, K. Liang, and X. Lu, SIAM Journal on Control and Optimization (to appear).
[29] G. Stadler, Computational Optimization and Applications **44**(2), 159–181 (2009).
[30] R. Herzog, G. Stadler, and G. Wachsmuth, submitted (2010), http://www.tu-chemnitz.de/mathematik/part_dgl/publications.php.
[31] C. Clason and K. Kunisch, ESAIM: Control, Optimisation, and Calculus of Variations (to appear).
[32] K. Kunisch and B. Vexler, SIAM Journal on Control and Optimization **46**(4), 1368–1397 (2007).