Chapter 6

Implicit Constraints

6.1 Introducton

Often one has to solve minimization problems

$$\min_u \widehat{f}(u), \tag{6.1}$$

where the objective function is of the form $\widehat{f}(u) = f(y(u), u)$ and $y(u)$ is the solution of an implicit equation

$$c(y, u) = 0, \tag{6.2}$$

where

$$c : \mathbb{R}^{n_y \times n_u} \to \mathbb{R}^{n_y}.$$

This is, for example, the case in the parameter identification problem studied in Section 3.5. There, $u = p \in \mathbb{R}^l$ and the evaluation of $\widehat{f}(p) = \frac{1}{2}\|R(p)\|_2$ involves the computation of the solution $y(\cdot; p)$ of an ordinary differential equation. Thus, in this example, $c(y, p) = 0$ symbolizes the system of ordinary differential equations. Since $c : \mathbb{R}^{n_y \times n_u} \to \mathbb{R}^{n_y}$ the parameter identification in ordinary differential equations as discussed in Section 3.5 is not included. In the example in Section 3.5, y is a function in $C(I, \mathbb{R}^n)$. However, if the ordinary differential equation is discretized using, e.g., a Runge Kutta method, we would obtain a system $c(y_h, u) = 0$ for the discretized solution y_h which fits into our framework. We see more examples later in this chapter.

One purpose of this chapter is to study the computation of the gradient and the Hessian on $\widehat{f}$, and to discuss the solution of the system $\nabla^2 \widehat{f}(u) s_u = -\nabla \widehat{f}(u)$, all of which are important ingredients in optimization methods for the solution of (6.1), (6.2). Another goal of this chapter is to introduce some specific problems of the type (6.1), (6.2) and to make the gradient and Hessian computation more concrete for these problems.

Note that (6.1), (6.2) can also be formulated as an equality constrained problem. In fact, since y is tied to u via the implicit equation, we could just include this equation into the problem formulation and reformulate (6.1), (6.2) as

$$\begin{array}{ll} \min & f(y,u), \\ \text{s.t.} & c(y,u) = 0. \end{array} \tag{6.3}$$

In (6.3), the optimization variables are $y \in \mathbb{R}^{n_y}$ and $u \in \mathbb{R}^{n_u}$. In (6.1), (6.2) the constraint $c(y,u) = 0$ is implicit. It is enforced by requiring that $y(u)$ is a solution of $c(y,u) = 0$. In (6.3) this constraint is exposed to the optimizer and the variables y, u are decoupled. The formulation (6.3) can have significant advantages over (6.1), (6.2) and we will discuss the solution of constrained optimization problems of the type (6.3) later. It turns out, however, that most of the tasks that arise in the computation of gradients $\nabla \widehat{f}(u)$ and Hessians $\nabla^2 \widehat{f}(u)$ for (6.1) also arise in the solution of (6.3). Therefore it is useful to study (6.1) even if the optimization formulation (6.3) is preferred.

6.2 Gradient and Hessian Computations

Let

$$\begin{array}{ll} c: & \mathbb{R}^{n_y \times n_u} \longrightarrow \mathbb{R}^{n_y}, \\ f: & \mathbb{R}^{n_y \times n_u} \longrightarrow \mathbb{R}. \end{array}$$

To make the formulation (6.1), (6.2) rigorous, we make the following assumptions.

Assumption 6.2.1

- *For all $u \in \mathbb{R}^{n_u}$ there exists a unique $y \in \mathbb{R}^{n_y}$ such that $c(y,u) = 0$.*
- *There exists an open set $D \subset \mathbb{R}^{n_y \times n_u}$ with $\{(y,u) \; : \; u \in \mathbb{R}^{n_u}, \;\; c(y,u) = 0\} \subset D$ such that f and c are twice differentiable on D.*
- *The inverse $c_y(y,u)^{-1}$ exists for all $(y,u) \in \{(y,u) \; : \; u \in \mathbb{R}^{n_u}, \;\; c(y,u) = 0\}$.*

Under these assumptions the implicit function theorem guarantees the existence of a differentiable function

$$y : U \rightarrow \mathbb{R}^{n_y}$$

defined by

$$c(y(u), u) = 0. \tag{6.4}$$

We consider the problem

$$\min_u \widehat{f}(u), \tag{6.5}$$

where

$$\widehat{f}(u) = f(y(u), u)$$

and $y(u)$ is the solution of $c(y(u), u) = 0$.

The implicit function theorem guarantees the differentiability of $y(\cdot)$. The derivative can be obtained by differentiating $c(y(u), u) = 0$. This yields[1]

$$y_u(u) = -c_y(y(u), u)^{-1} c_u(y(u), u). \tag{6.6}$$

The derivative $y_u(u)$ is also called the *sensitivity* (of y with respect to u).

Since $y(\cdot)$ is differentiable the function $\widehat{f}$ is differentiable and its gradient is given by

$$\begin{aligned} \nabla \widehat{f}(u) &= y_u(u)^T \nabla_y f(y(u), u) + \nabla_u f(y(u), u) \\ &= -c_u(y(u), u)^T c_y(y(u), u)^{-T} \nabla_y f(y(u), u) + \nabla_u f(y(u), u). \end{aligned} \tag{6.7}$$

If $\lambda(u) \in \mathbb{R}^{n_y}$ is the solution of

$$c_y(y(u), u)^T \lambda(u) = -\nabla_y f(y(u), u), \tag{6.8}$$

then the gradient can be written as

$$\nabla \widehat{f}(u) = \nabla_u f(y(u), u) + c_u(y(u), u)^T \lambda(u). \tag{6.9}$$

The system (6.8) are also called the adjoint equations.

The equations (6.7) and (6.9) suggest two methods for computing the gradient.

Algorithm 6.2.2 (Gradient Computation Using Sensitivities)

1. *Given* u*, solve* $c(y, u) = 0$ *for* y.
2. *Compute the sensitivities* $S = y_u(u)$ *by solving* n_u *linear systems* $c_y(y(u), u)\, S = -c_u(y(u), u)$.
3. *Compute* $\nabla \widehat{f}(u) = S^T \nabla_y f(y(u), u) + \nabla_u f(y(u), u)$.

Algorithm 6.2.3 (Gradient Computation Using Adjoints)

1. *Given* u*, solve* $c(y, u) = 0$ *for* y.
2. *Solve the adjoint equation* $c_y(y(u), u)^T \lambda = -\nabla_y f(y(u), u)$ *for* λ.
3. *Compute* $\nabla \widehat{f}(u) = \nabla_u f(y(u), u) + c_u(y(u), u)^T \lambda(u)$.

[1] We use the simplified expressions $c_y(y(u), u)$ and $c_u(y(u), u)$ instead of $c_y(y, u)|_{y=y(u)}$ and $c_u(y, u)|_{y=y(u)}$, respectively. Similar simplifications in notation are applied throughout this section.

It is useful to introduce the so-called Lagrange function or Lagrangian

$$L(y, u, \lambda) = f(y, u) + \lambda^T c(y, u). \tag{6.10}$$

Using the Lagrangian, the equation (6.8) can be written as

$$\nabla_y L(y(u), u, \lambda(u)) = 0. \tag{6.11}$$

Moreover, (6.9) can be written as

$$\nabla \widehat{f}(u) = \nabla_u L(y(u), u, \lambda(u)). \tag{6.12}$$

It will also be helpful to introduce the matrix

$$W(y, u) = \begin{pmatrix} -c_y(y, u)^{-1} c_u(y, u) \\ I \end{pmatrix}. \tag{6.13}$$

If $y = y(u)$ then

$$W(y(u), u) = \begin{pmatrix} y_u(u) \\ I \end{pmatrix} \tag{6.14}$$

and the gradient of $\widehat{f}$ can be written as

$$\nabla \widehat{f}(u) = W(y(u), u)^T \nabla_x f(y(u), u), \tag{6.15}$$

cf., (6.8), (6.9).

Under the general assumption of this section, the Hessian of $\widehat{f}$ exists. If we differentiate (6.12), we find that

$$\begin{aligned} \nabla^2 \widehat{f}(u) \;&=\; \nabla_{uy} L(y(u), u, \lambda(u))\, y_u(u) + \nabla_{uu} L(y(u), u, \lambda(u)) \\ &\quad + \nabla_{u\lambda} L(y(u), u, \lambda(u))\, \lambda_u(u), \end{aligned} \tag{6.16}$$

Under the general assumption of this section, the partial derivatives of L exist and the derivative $y_u(u)$ exists and is given by (6.6). The existence of the derivative $\lambda_u(u)$ is also guaranteed under the general assumptions of this section. In fact, the function $\lambda(u)$ is implicitly defined by (6.9) or, equivalently, (6.11). Under the general assumptions of this section, the differentiability of this function follows from the implicit function theorem and the derivative $\lambda_u(u)$ is obtained by differentiating (6.11):

$$\begin{aligned} \nabla_{yy} L(y(u), u, \lambda(u))\; y_u(u) + \nabla_{yu} L(y(u), u, \lambda(u)) \\ + \nabla_{y\lambda} L(y(u), u, \lambda(u))\, \lambda_u(u) \;&=\; 0. \end{aligned}$$

Using $\nabla_{y\lambda} L(y(u), u, \lambda(u)) = c_y(y(u), u)^T$ and (6.6) we find that

$$\begin{aligned} \lambda_u(u) \;&=\; c_y(y(u), u)^{-T} \big[\nabla_{yy} L(y(u), u, \lambda(u))\, c_y(y(u), u)^{-1} c_u(y(u), u) \\ &\qquad - \nabla_{yu} L(y(u), u, \lambda(u)) \big]. \end{aligned} \tag{6.17}$$

Now we insert (6.17) and (6.6) into (6.16) and observe that $\nabla_{u\lambda} L(y(u), u, \lambda(u)) = c_u(y(u), u)$ to write

$$\begin{aligned} \nabla^2 \widehat{f}(u) &= W(y,u)^T \begin{pmatrix} \nabla_{yy} L(y,u,\lambda) & \nabla_{yu} L(y,u,\lambda) \\ \nabla_{uy} L(y,u,\lambda) & \nabla_{uu} L(y,u,\lambda) \end{pmatrix} W(y,u) \\ &= c_u(y,u)^T c_y(y,u)^{-T} \nabla_{yy} L(y,u,\lambda)\, c_y(y,u)^{-1} c_u(y,u) \\ &\quad -c_u(y,u)^T c_y(y,u)^{-T} \nabla_{yu} L(y,u,\lambda) \\ &\quad -\nabla_{uy} L(y,u,\lambda)\, c_y(y,u)^{-1} c_u(y,u) + \nabla_{uu} L(y,u,\lambda). \end{aligned} \tag{6.18}$$

Many optimization algorithms require the computation of Hessian times vector products $\nabla^2 \widehat{f}(u) v$. Using the equality (6.18) this can be done as follows.

Algorithm 6.2.4 (Hessian–Times–Vector Computation Using Adjoints)

1. *Given u, solve $c(y,u) = 0$ for y (if not done already).*
2. *Solve the adjoint equation $c_y(y(u),u)^T \lambda = -\nabla_y f(y(u),u)$ for λ (if not done already).*
3. *Solve the equation $c_y(y(u),u)\, w = c_u(y(u),u) v$.*
4. *Solve the equation*
   $c_y(y(u),u)^T p = \nabla_{yy} L(y(u),u,\lambda(u)) w - \nabla_{yu} L(y(u),u,\lambda(u)) v$.
5. *Compute*
   $\nabla^2 \widehat{f}(u) v = c_u(y(u),u)^T p - \nabla_{uy} L(y(u),u,\lambda(u)) w + \nabla_{uu} L(y(u),u,\lambda(u)) v$.

Hence, if $y(u)$ and $\lambda(u)$ are already known, then the computation of $\nabla^2 \widehat{f}(u) v$ requires the solution of two linear equations. One similar to the linearized state equation, Step 3, and one similar to the adjoint equation, Step 4.

One can show the following result.

Theorem 6.2.5 *Let $c_y(y(u),u)$ be invertible. The Newton equation*

$$\nabla^2 \widehat{f}(u)\, s_u = -\nabla \widehat{f}(u). \tag{6.19}$$

has a (unique) solution if and only if the problem

$$\begin{aligned} \min \quad & \begin{pmatrix} \nabla_y f(y,u)^T \\ \nabla_u f(y,u) \end{pmatrix}^T \begin{pmatrix} s_y \\ s_u \end{pmatrix} + \frac{1}{2} \begin{pmatrix} s_y \\ s_u \end{pmatrix}^T \begin{pmatrix} \nabla_{yy} L(y,u,\lambda) & \nabla_{yu} L(y,u,\lambda) \\ \nabla_{uy} L(y,u,\lambda) & \nabla_{uu} L(y,u,\lambda) \end{pmatrix} \begin{pmatrix} s_y \\ s_u \end{pmatrix}, \\ \text{s.t.} \quad & c_y(y,u) s_y + c_u(y,u) s_u = 0, \end{aligned} \tag{6.20}$$

where $y = y(u)$ and $\lambda = \lambda(u)$, has a (unique) solution. If s_u solves the Newton equation (6.19), then $s_y = c_y(y(u),u)^{-1} c_u(y(u),u) s_u$, s_u solves (6.20).

Similarly, one can show the following result.

Theorem 6.2.6 *Let $c_y(y(u), u)$ be invertible. the Newton–like equation*

$$\widehat{H}\, s_u = -\nabla \widehat{f}(u), \tag{6.21}$$

where $\widehat{H} \in \mathbb{R}^{n_u \times n_u}$ is symmetric has a (unique) solution if and only if the problem

$$\begin{aligned} \min \quad & \begin{pmatrix} \nabla_y f(y,u)^T \\ \nabla_u f(y,u) \end{pmatrix}^T \begin{pmatrix} s_y \\ s_u \end{pmatrix} + \frac{1}{2} \begin{pmatrix} s_y \\ s_u \end{pmatrix}^T \begin{pmatrix} 0 & 0 \\ 0 & \widehat{H} \end{pmatrix} \begin{pmatrix} s_y \\ s_u \end{pmatrix}, \\ \textit{s.t.} \quad & c_y(y,u)s_y + c_u(y,u)s_u = 0, \end{aligned} \tag{6.22}$$

where $y = y(u)$ and $\lambda = \lambda(u)$, has a (unique) solution. Again, if s_u solves the Newton–like equation (6.21), then $s_y = c_y(y(u), u)^{-1} c_u(y(u), u) s_u$, s_u solves (6.22).

6.3 Solution of a Time–Dependent Linear Quadratic Optimal Control Problem

The example in this section builds on Section **??**. Again, let $\Omega \subset \mathbb{R}^2$ be an open convex set with boundary $\partial\Omega$.

We want to minimize

$$\min_u \frac{1}{2} \int_0^T \int_\Omega (y(x,t) - \hat{y}(x,t))^2 dx dt + \frac{\omega}{2} \int_0^T \int_{\partial\Omega} u^2(x,t) dx dt, \tag{6.23}$$

where y is the solution of

$$\begin{aligned} \frac{\partial}{\partial t} y(x,t) - \nabla \cdot (c(x,t)\nabla y(x,t)) + a(x,t)y(x,t) &= r(x,t) \quad (x,t) \in \Omega \times (0,T), \\ n(x) \cdot (c(x,t)\nabla y(x,t)) &= u(x,t) \quad (x,t) \in \partial\Omega \times (0,T), \\ y(x,0) &= y_0(x) \quad x \in \Omega, \end{aligned} \tag{6.24}$$

where $a, c, r : \Omega \times (0,T) \to \mathbb{R}$, and $y_0 : \Omega \to \mathbb{R}$ are given functions and $n(x)$ denotes the unit outward normal to $\partial\Omega$ at $x \in \partial\Omega$.

To discretize (6.23), (6.24), we first use a finite element discretization in space. We approximate y and u by functions of the form

$$y_h(x,t) = \sum_{j=1}^{n_y} y_j(t)\varphi_j(x) \tag{6.25}$$

and

$$u_h(x,t) = \sum_{j=1}^{n_u} u_j(t)\varphi_j(x). \tag{6.26}$$

If we set

$$\vec{y}(t) = (y_1(t), \ldots, y_{n_y}(t))^T \quad \text{and} \quad \vec{u}(t) = (u_1(t), \ldots, u_{n_u}(t))^T,$$

then a discretization of the linear quadratic optimal control problem (6.24), (6.23) is given by

$$\min_{\vec{u}} \int_0^T \frac{1}{2}\vec{y}(t)^T M_h \vec{y}(t) + (g_h(t))^T \vec{y}(t) + \frac{\omega}{2}\vec{u}(t)^T Q_h \vec{u}(t) dt, \tag{6.27}$$

where $\vec{y}(t)$ is the solution of

$$\begin{aligned} M_h \frac{d}{dt}\vec{y}(t) + A_h(t)\vec{y}(t) + B_h(t)\vec{u}(t) &= r_h(t), \quad t \in (0,T), \\ \vec{y}(0) &= \vec{y}_0, \end{aligned} \tag{6.28}$$

where $\vec{y}_0$ is the solution of

$$M_h \vec{y}_0 = \begin{pmatrix} \int_\Omega y_0(x)\varphi_1(x)dx \\ \vdots \\ \int_\Omega y_0(x)\varphi_{n_y}(x)dx \end{pmatrix}.$$

In (6.27), (6.28) the vectors and matrices g_h, A_h, B_h may depend on time because the functions $c, a, \hat{y}$ may depend on time.

To discretize the problem in time, let

$$0 = t_0 < t_1 < \ldots < t_{N+1} = T$$

be a partition of the time interval and define $\Delta t_i = (t_{i+1} - t_i)$ for $i = 0, \ldots, N$. Using (6.28) we obtain

$$\begin{aligned} M_h\vec{y}(t_{i+1}) - M_h\vec{y}(t_i) &= \int_{t_i}^{t_{i+1}} \frac{d}{dt} M_h\vec{y}(t)dt \\ &= \int_{t_i}^{t_{i+1}} r_h(t) - A_h(t)\vec{y}(t) - B_h(t)\vec{u}(t) \\ &\approx \frac{\Delta t_i}{2}\left(r_h(t_i) - A_h(t_i)\vec{y}(t_i) - B_h(t_i)\vec{u}(t_i)\right) \\ &\quad + \frac{\Delta t_i}{2}\left(r_h(t_{i+1}) - A_h(t_{i+1})\vec{y}(t_{i+1}) - B_h(t_{i+1})\vec{u}(t_{i+1})\right) \end{aligned}$$

$i = 0, \ldots, N$. This leads to the Crank-Nicolson method for the approximation of (6.28):

$$\begin{aligned} \Big(M_h + \frac{\Delta t_i}{2}A_h(t_{i+1})\Big)\vec{y}_{i+1} + \frac{\Delta t_i}{2}B_h(t_{i+1})\vec{u}_{i+1} + \Big(-M_h + \frac{\Delta t_i}{2}A_h(t_i)\Big)\vec{y}_i + \frac{\Delta t_i}{2}B_h(t_i)\vec{u}_i \\ = \frac{\Delta t_i}{2}\Big(r_h(t_i) + r_h(t_{i+1})\Big), \quad i = 0, \ldots, N, \end{aligned}$$

where $\vec{y}_0$ is given by the initial conditions in (6.28). The objective function (6.27) is discretized using the composite trapezoidal rule

$$\begin{aligned} &\int_0^T \frac{1}{2}\vec{y}(t)^T M_h\vec{y}(t) + (g_h(t))^T\vec{y}(t) + \frac{\omega}{2}\vec{u}(t)^T Q_h\vec{u}(t)dt \\ &= \sum_{i=0}^{N} \int_{t_i}^{t_{i+1}} \frac{1}{2}\vec{y}(t)^T M_h\vec{y}(t) + (g_h(t))^T\vec{y}(t) + \frac{\omega}{2}\vec{u}(t)^T Q_h\vec{u}(t)dt \\ &\approx \sum_{i=0}^{N} \frac{\Delta t_i}{2}\Big(\frac{1}{2}\vec{y}(t_i)^T M_h\vec{y}(t_i) + (g_h(t_i))^T\vec{y}(t_i) + \frac{\omega}{2}\vec{u}(t_i)^T Q_h\vec{u}(t_i) \\ &\qquad + \frac{1}{2}\vec{y}(t_{i+1})^T M_h\vec{y}(t_{i+1}) + (g_h(t_{i+1}))^T\vec{y}(t_{i+1}) + \frac{\omega}{2}\vec{u}(t_{i+1})^T Q_h\vec{u}(t_{i+1})\Big) \\ &= \sum_{i=0}^{N+1} \frac{\Delta t_{i-1} + \Delta t_i}{2}\left(\frac{1}{2}\vec{y}_i^T M_h\vec{y}_i + (g_h)_i^T\vec{y}_i + \frac{\omega}{2}\vec{u}_i^T Q_h\vec{u}_i\right), \end{aligned}$$

where we have set $\Delta t_{-1} = \Delta t_{N+1} = 0$.

Now the fully discretized problem is given by

$$\min_{\vec{u}_1,\ldots,\vec{u}_{N+1}} \sum_{i=0}^{N+1} \frac{\Delta t_{i-1} + \Delta t_i}{2} \left(\frac{1}{2} \vec{y}_i^T M_h \vec{y}_i + (g_h)_i^T \vec{y}_i + \frac{\omega}{2} \vec{u}_i^T Q_h \vec{u}_i \right), \tag{6.29}$$

where $\vec{y}_1, \ldots, \vec{y}_{N+1}$ is the solution of

$$\begin{aligned} \Big(M_h + \frac{\Delta t_i}{2} A_h(t_{i+1}) \Big) \vec{y}_{i+1} + \frac{\Delta t_i}{2} B_h(t_{i+1}) \vec{u}_{i+1} + \Big(- M_h + \frac{\Delta t_i}{2} A_h(t_i) \Big) \vec{y}_i + \frac{\Delta t_i}{2} B_h(t_i) \vec{u}_i \\ = \frac{\Delta t_i}{2} \Big(r_h(t_i) + r_h(t_{i+1}) \Big), \quad i = 0, \ldots, N, \end{aligned} \tag{6.30}$$

and $\vec{y}_0$ is given by the initial conditions in (6.28). This problem is a DTOC of the form (**??**), (**??**). We denote the objective function in (6.29) by $\widehat{f}$.

The Lagrangian corresponding to (6.29), (6.30) is given by

$$\begin{aligned} & L(\vec{y}_1, \ldots, \vec{y}_{N+1}, \vec{u}_0, \ldots, \vec{u}_{N+1}, \vec{\lambda}_1, \ldots, \vec{\lambda}_{N+1}) \\ = & \sum_{i=0}^{N+1} \frac{\Delta t_{i-1} + \Delta t_i}{2} \left(\frac{1}{2} \vec{y}_i^T M_h \vec{y}_i + (g_h)_i^T \vec{y}_i + \frac{\omega}{2} \vec{u}_i^T Q_h \vec{u}_i \right) \\ & \sum_{i=0}^{N} \vec{\lambda}_{i+1}^T \Big[\Big(M_h + \frac{\Delta t_i}{2} A_h(t_{i+1}) \Big) \vec{y}_{i+1} + \frac{\Delta t_i}{2} B_h(t_{i+1}) \vec{u}_{i+1} \\ & \qquad + \Big(- M_h + \frac{\Delta t_i}{2} A_h(t_i) \Big) \vec{y}_i + \frac{\Delta t_i}{2} B_h(t_i) \vec{u}_i \\ & \qquad - \frac{\Delta t_i}{2} \Big(r_h(t_i) + r_h(t_{i+1}) \Big) \Big]. \end{aligned} \tag{6.31}$$

The adjoint equations corresponding to (6.8) or (**??**) are obtained by setting the partial derivatives with respect to y_i of the Lagrangian (6.31) to zero and are given by

$$\begin{aligned} \Big(M_h + \tfrac{\Delta t_N}{2} A_h(t_{N+1}) \Big)^T \vec{\lambda}_{N+1} &= -\tfrac{\Delta t_N}{2} (M_h \vec{y}_{N+1} + (g_h)_{N+1}), \\ \Big(M_h + \tfrac{\Delta t_{i-1}}{2} A_h(t_i) \Big)^T \vec{\lambda}_i &= -\Big(- M_h + \tfrac{\Delta t_i}{2} A_h(t_i) \Big)^T \vec{\lambda}_{i+1} \\ & \quad - \tfrac{\Delta t_{i-1} + \Delta t_i}{2} (M_h \vec{y}_i + (g_h)_i), \quad i = N, \ldots, 1. \end{aligned} \tag{6.32}$$

(Recall that $\Delta t_{N+1} = 0$.) Given the solution of (6.32), the gradient of the objective function $\widehat{f}$ in (**??**) can be obtained by computing the partial derivatives with respect to u_i of the Lagrangian (6.31). The gradient is given by

$$= \begin{pmatrix} \omega \frac{\Delta t_0}{2} Q_h \vec{u}_0 + \frac{\Delta t_0}{2} B_h^T \vec{\lambda}_1 \\ \omega \frac{\Delta t_0 + \Delta t_1}{2} Q_h \vec{u}_1 + B_h^T (\frac{\Delta t_0}{2} \vec{\lambda}_1 + \frac{\Delta t_1}{2} \vec{\lambda}_2) \\ \vdots \\ \omega \frac{\Delta t_{N-1} + \Delta t_N}{2} Q_h \vec{u}_N + B_h^T (\frac{\Delta t_{N-1}}{2} \vec{\lambda}_N + \frac{\Delta t_N}{2} \vec{\lambda}_{N+1}) \\ \omega \frac{\Delta t_N}{2} Q_h \vec{u}_{N+1} + \frac{\Delta t_N}{2} B_h^T \vec{\lambda}_{N+1} \end{pmatrix}. \tag{6.33}$$

(Recall that $\Delta t_{-1} = \Delta t_{N+1} = 0$.)

We summarize the gradient computation using adjoints in the following algorithm.

Algorithm 6.3.1 (Gradient Computation Using Adjoints)

1. *Given* $\vec{u}_1, \ldots, \vec{u}_{N+1}$*, and* $\vec{y}_0$ *compute* $\vec{y}_1, \ldots, \vec{y}_{N+1}$ *by solving*
$$\begin{aligned}\Big(M_h + \frac{\Delta t_i}{2} A_h(t_{i+1})\Big)\vec{y}_{i+1} &= -\frac{\Delta t_i}{2} B_h(t_{i+1})\vec{u}_{i+1} - \Big(-M_h + \frac{\Delta t_i}{2} A_h(t_i)\Big)\vec{y}_i \\ &\quad -\frac{\Delta t_i}{2} B_h(t_i)\vec{u}_i + \frac{\Delta t_i}{2}\Big(r_h(t_i) + r_h(t_{i+1})\Big), \quad i = 0, \ldots, N.\end{aligned}$$

2. *Compute* $\vec{\lambda}_{N+1}, \ldots, \vec{\lambda}_1$ *by solving*
$$\begin{aligned}\Big(M_h + \frac{\Delta t_N}{2} A_h(t_{N+1})\Big)^T \vec{\lambda}_{N+1} &= -\frac{\Delta t_N}{2}(M_h\vec{y}_{N+1} + (g_h)_{N+1}), \\ \Big(M_h + \frac{\Delta t_{i-1}}{2} A_h(t_i)\Big)^T \vec{\lambda}_i &= -\Big(-M_h + \frac{\Delta t_i}{2} A_h(t_i)\Big)^T \vec{\lambda}_{i+1} \\ &\quad -\frac{\Delta t_{i-1} + \Delta t_i}{2}(M_h\vec{y}_i + (g_h)_i), \quad i = N, \ldots, 1.\end{aligned}$$

3. *Compute* $\nabla_u \widehat{f}(u)$ *from* (6.33).

We conclude by adapting Algorithms 6.2.4 and **??** to our problem. Since the the objective function (6.29) is quadratic and the implicit constraints (6.30) are linear, most of the second derivative terms are zero. The multiplication of the Hessian $\nabla_u^2 \widehat{f}(u)$ times vector v computation can be performed using the the following algorithm.

Algorithm 6.3.2 (Hessian–Times–Vector Computation Using Adjoints)

1. *Given* $\vec{u}_1, \ldots, \vec{u}_{N+1}$*, and* $\vec{y}_0$ *compute* $\vec{y}_1, \ldots, \vec{y}_{N+1}$ *by solving*
$$\begin{aligned}\Big(M_h + \frac{\Delta t_i}{2} A_h(t_{i+1})\Big)\vec{y}_{i+1} &= -\frac{\Delta t_i}{2} B_h(t_{i+1})\vec{u}_{i+1} - \Big(-M_h + \frac{\Delta t_i}{2} A_h(t_i)\Big)\vec{y}_i \\ &\quad -\frac{\Delta t_i}{2} B_h(t_i)\vec{u}_i + \frac{\Delta t_i}{2}\Big(r_h(t_i) + r_h(t_{i+1})\Big), \quad i = 0, \ldots, N\end{aligned}$$
(if not done already).

2. *Compute* $\vec{\lambda}_{N+1}, \ldots, \vec{\lambda}_1$ *by solving*
$$\begin{aligned}\Big(M_h + \frac{\Delta t_N}{2} A_h(t_{N+1})\Big)^T \vec{\lambda}_{N+1} &= -\frac{\Delta t_N}{2}(M_h\vec{y}_{N+1} + (g_h)_{N+1}), \\ \Big(M_h + \frac{\Delta t_{i-1}}{2} A_h(t_i)\Big)^T \vec{\lambda}_i &= -\Big(-M_h + \frac{\Delta t_i}{2} A_h(t_i)\Big)^T \vec{\lambda}_{i+1} \\ &\quad -\frac{\Delta t_{i-1} + \Delta t_i}{2}(M_h\vec{y}_i + (g_h)_i), \quad i = N, \ldots, 1\end{aligned}$$
(if not done already).

3. *Compute* $\vec{w}_1, \ldots, \vec{w}_{N+1}$ *from*

$$\Big(M_h + \frac{\Delta t_i}{2} A_h(t_{i+1})\Big)\vec{w}_{i+1} = -\frac{\Delta t_i}{2} B_h(t_{i+1})\vec{v}_{i+1} - \Big(-M_h + \frac{\Delta t_i}{2} A_h(t_i)\Big)\vec{w}_i - \frac{\Delta t_i}{2} B_h(t_i)\vec{v}_i,$$

$i = 0, \ldots, N$*, where* $\vec{w}_0 = 0$*.*

4. *Compute* $\vec{p}_{N+1}, \ldots, \vec{p}_1$ *by solving*

$$\begin{aligned} \Big(M_h + \frac{\Delta t_N}{2} A_h(t_{N+1})\Big)^T \vec{p}_{N+1} &= \frac{\Delta t_N}{2} M_h \vec{w}_{N+1}, \\ \Big(M_h + \frac{\Delta t_{i-1}}{2} A_h(t_i)\Big)^T \vec{p}_i &= -\Big(-M_h + \frac{\Delta t_i}{2} A_h(t_i)\Big)^T \vec{p}_{i+1} \\ &\quad + \frac{\Delta t_{i-1} + \Delta t_i}{2} M_h \vec{w}_i, \quad i = N, \ldots, 1. \end{aligned}$$

5. *Compute*

$$\nabla^2 \widehat{f}(u)v = \begin{pmatrix} \omega \frac{\Delta t_0}{2} Q_h \vec{v}_0 + \frac{\Delta t_0}{2} B_h^T \vec{p}_1 \\ \omega \frac{\Delta t_0 + \Delta t_1}{2} Q_h \vec{v}_1 + B_h^T (\frac{\Delta t_0}{2} \vec{p}_1 + \frac{\Delta t_1}{2} \vec{p}_2) \\ \vdots \\ \omega \frac{\Delta t_{N-1} + \Delta t_N}{2} Q_h \vec{v}_N + B_h^T (\frac{\Delta t_{N-1}}{2} \vec{p}_N + \frac{\Delta t_N}{2} \vec{p}_{N+1}) \\ \omega \frac{\Delta t_N}{2} Q_h \vec{v}_{N+1} + \frac{\Delta t_N}{2} B_h^T \vec{p}_{N+1} \end{pmatrix}.$$

6.4 Optimal Control of Burgers' Equation

We want to minimize

$$\min_u \tfrac{1}{2} \int_0^T \int_0^1 (y(x,t) - \hat{y}(x,t))^2 + \omega u^2(x,t) dx dt, \tag{6.34}$$

where y is the solution of

$$\begin{aligned} \tfrac{\partial}{\partial t} y(x,t) - \nu \tfrac{\partial^2}{\partial x^2} y(x,t) + \tfrac{\partial}{\partial x} y(x,t) y(x,t) &= r(x,t) + u(x,t) \qquad && (x,t) \in (0,1) \times (0,T), \\ y(0,t) = y(1,t) &= 0 && t \in (0,T), \\ y(x,0) &= y_0(x) && x \in (0,1), \end{aligned} \tag{6.35}$$

where $r : (0,1) \times (0,T) \to \mathbb{R}$, and $y_0 : (0,1) \to \mathbb{R}$ are given functions and $\omega, \nu > 0$ are given parameters. The parameter $\nu > 0$ is also called the viscosity and the differential equation (6.35) is known as the (viscous) Burgers' equation. Burgers' equation can be viewed as a simplified version of the Navier-Stokes equations. It was introduced by Burgers [Bur40, Bur48] as a simple model to investigate turbulence.

To discretize (6.34), (6.35), we first use a finite element discretization in space. For this purpose, we multiply the differential equation in (6.35) by a function $v \in C^1(\Omega)$ which satisfies $v(0) = v(1) = 0$; then we integrate both sides over $(0,1)$, and apply integration by parts. This leads to

$$\begin{aligned} &\frac{d}{dt} \int_0^1 y(x,t) v(x) dx + \nu \int_0^1 \frac{\partial}{\partial x} y(x,t) \frac{d}{dx} v(x) dx + \int_0^1 \frac{\partial}{\partial x} y(x,t) y(x,t) v(x) dx \\ = &\int_0^1 (r(x,t) + u(x,t)) v(x) dx \end{aligned} \tag{6.36}$$

for all $v \in C^1(\Omega)$ with $v(0) = v(1) = 0$ and $t \in (0,T)$. We approximate y and u by functions of the form

$$y_h(x,t) = \sum_{j=1}^{n_y} y_j(t) \varphi_j(x) \tag{6.37}$$

and

$$u_h(x,t) = \sum_{j=1}^{n_u} u_j(t) \psi_j(x), \tag{6.38}$$

where $\varphi_j(0) = \varphi_j(1) = 0$, $j = 1, \ldots, n_y$. We set

$$\vec{y}(t) = (y_1(t), \ldots, y_{n_y}(t))^T \quad \text{and} \quad \vec{u}(t) = (u_1(t), \ldots, u_{n_u}(t))^T,$$

If we insert these approximations into (6.36) and require (6.36) to hold for for $v = \varphi_j$, $j = 1, \ldots, n_y$, then we obtain the system of ordinary differential equations

$$M_h \frac{d}{dt} \vec{y}(t) + A_h \vec{y}(t) + N_h(\vec{y}(t)) + B_h \vec{u}(t) = r_h(t), \quad t \in (0,T), \tag{6.39}$$

where $M_h, A_h \in \mathbb{R}^{n_y \times n_y}$, $B_h \in \mathbb{R}^{n_y \times n_u}$, $r_h(t) \in \mathbb{R}^{n_y}$, and $N_h(\vec{y}(t)) \in \mathbb{R}^{n_y}$ are matrices or vectors with entries

$$
\begin{aligned}
(M_h)_{ij} &= \int_0^1 \varphi_j(x)\varphi_i(x)dx, \\
(A_h)_{ij} &= \nu \int_0^1 \frac{d}{dx}\varphi_j(x)\frac{d}{dx}\varphi_i(x)dx, \\
(B_h)_{ij} &= -\int_0^1 \psi_j(x)\varphi_i(x)dx, \\
(N_h(\vec{y}(t)))_i &= \sum_{j=1}^{n_y}\sum_{k=1}^{n_y}\int_0^1 \frac{d}{dx}\varphi_j(x)\varphi_k(x)\varphi_i(x)dx y_k(t)y_j(t), \\
(r_h(t))_i &= \int_0^1 r(x,t)\varphi_i(x)dx.
\end{aligned}
$$

If we insert (6.37), (6.38) into (6.34), we obtain

$$
\int_0^T \frac{1}{2}\vec{y}(t)^T M_h \vec{y}(t) + (g_h(t))^T\vec{y}(t) + \frac{\omega}{2}\vec{u}(t)^T Q_h \vec{u}(t)dt + \int_0^T\int_0^1 \frac{1}{2}\hat{y}^2(x,t)dxdt,
$$

where $M_h \in \mathbb{R}^{n_y \times n_y}$ is defined as before and $Q_h \in \mathbb{R}^{n_u \times n_u}$, $g_h(t) \in \mathbb{R}^{n_y}$ are a matrix and vector with entries

$$
\begin{aligned}
(Q_h)_{ij} &= \int_0^1 \psi_j(x)\psi_i(x)dx, \\
(g_h(t))_i &= -\int_0^1 \hat{y}(x,t)\varphi_i(x)dx.
\end{aligned}
$$

Since $\int_0^T\int_0^1 \frac{1}{2}\hat{y}^2(x,t)dxdt$ is a given constant we will omit it from the objective function. Thus a semi–discretization of the optimal control problem (6.34), (6.35) is given by

$$
\min_{\vec{u}} \int_0^T \frac{1}{2}\vec{y}(t)^T M_h \vec{y}(t) + (g_h(t))^T\vec{y}(t) + \frac{\omega}{2}\vec{u}(t)^T Q_h \vec{u}(t)dt, \tag{6.40}
$$

where $\vec{y}(t)$ is the solution of

$$
\begin{aligned}
M_h \tfrac{d}{dt}\vec{y}(t) + A_h\vec{y}(t) + N_h(\vec{y}(t)) + B_h\vec{u}(t) &= r_h(t), \quad t \in (0,T), \\
\vec{y}(0) &= \vec{y}_0,
\end{aligned} \tag{6.41}
$$

where $\vec{y}_0$ is the solution of

$$
M_h\vec{y}_0 = \begin{pmatrix} \int_\Omega y_0(x)\varphi_1(x)dx \\ \vdots \\ \int_\Omega y_0(x)\varphi_{n_y}(x)dx \end{pmatrix}.
$$

To discretize the problem in time, we proceed as in Section 6.3. We let

$$0 = t_0 < t_1 < \ldots < t_{N+1} = T$$

and we define

$$\Delta t_i = t_{i+1} - t_i, \quad i = 0, \ldots, N.$$

We also introduce

$$\Delta t_{-1} = \Delta t_{N+1} = 0.$$

For the time discretization of (6.41) we use the Crank-Nicolson method and for the the discretization of the integral in (6.40) we use the composite Trapezoidal rule. The fully discretized problem is given by

$$\min_{\vec{u}_0,\ldots,\vec{u}_{N+1}} \sum_{i=0}^{N+1} \frac{\Delta t_{i-1} + \Delta t_i}{2} \left(\frac{1}{2} \vec{y}_i^T M_h \vec{y}_i + (g_h)_i^T \vec{y}_i + \frac{\omega}{2} \vec{u}_i^T Q_h \vec{u}_i \right), \tag{6.42}$$

where $\vec{y}_1, \ldots, \vec{y}_{N+1}$ is the solution of

$$\begin{aligned} &\Big(M_h + \frac{\Delta t_i}{2} A_h\Big) \vec{y}_{i+1} + \frac{\Delta t_i}{2} N_h(\vec{y}_{i+1}) + \frac{\Delta t_i}{2} B_h \vec{u}_{i+1} \\ &+ \Big(- M_h + \frac{\Delta t_i}{2} A_h\Big) \vec{y}_i + \frac{\Delta t_i}{2} N_h(\vec{y}_i) + \frac{\Delta t_i}{2} B_h \vec{u}_i \\ &- \frac{\Delta t_i}{2} \Big(r_h(t_i) + r_h(t_{i+1}) \Big) = 0, \quad i = 0, \ldots, N, \end{aligned} \tag{6.43}$$

and $\vec{y}_0$ is given by the initial conditions in (6.41). This problem is a DTOC of the form (**??**), (**??**). We denote the objective function in (6.42) by $\widehat{f}$.

Note that since the Burgers' equation (6.43) is quadratic in $\vec{y}_{i+1}$, the computation of $\vec{y}_{i+1}$, $i = 0, \ldots, N$, requires the solution of system of nonlinear equations. This can be done using Newton's method, which we will discuss in Chapter 9. The invertibility of the Jacobian $M_h + \frac{\Delta t_i}{2} A_h + \frac{\Delta t_i}{2} N_h'(\vec{y}_{i+1})$ typically requires a restriction of the spatial or temporal mesh which depends on the solution $\vec{y}_{i+1}$ and the viscosity ν (see, e.g., [Vol97, pp. 103,104]). These conditions are satisfied for the discretization used in our numerical example, Example 6.4.3.

The Lagrangian corresponding to (6.42), (6.43) is given by

$$\begin{aligned} &L(\vec{y}_1, \ldots, \vec{y}_{N+1}, \vec{u}_0, \ldots, \vec{u}_{N+1}, \vec{\lambda}_1, \ldots, \vec{\lambda}_{N+1}) \\ &= \sum_{i=0}^{N+1} \frac{\Delta t_{i-1} + \Delta t_i}{2} \left(\frac{1}{2} \vec{y}_i^T M_h \vec{y}_i + (g_h)_i^T \vec{y}_i + \frac{\omega}{2} \vec{u}_i^T Q_h \vec{u}_i \right) \\ &\quad + \sum_{i=0}^{N} \vec{\lambda}_{i+1}^T \Big[\Big(M_h + \frac{\Delta t_i}{2} A_h\Big) \vec{y}_{i+1} + \frac{\Delta t_i}{2} N_h(\vec{y}_{i+1}) + \frac{\Delta t_i}{2} B_h \vec{u}_{i+1} \\ &\qquad + \Big(- M_h + \frac{\Delta t_i}{2} A_h\Big) \vec{y}_i + \frac{\Delta t_i}{2} N_h(\vec{y}_i) + \frac{\Delta t_i}{2} B_h \vec{u}_i \\ &\qquad - \frac{\Delta t_i}{2} \Big(r_h(t_i) + r_h(t_{i+1}) \Big) \Big]. \end{aligned} \tag{6.44}$$

The adjoint equations corresponding to (6.8) or (**??**) are obtained by setting the partial derivatives with respect to y_i of the Lagrangian (6.44) to zero and are given by

$$\begin{aligned}\Big(M_h + \tfrac{\Delta t_N}{2} A_h + \tfrac{\Delta t_N}{2} N_h'(\vec{y}_{N+1})\Big)^T \vec{\lambda}_{N+1} &= -\tfrac{\Delta t_N}{2}(M_h \vec{y}_{N+1} + (g_h)_{N+1}),\\ \Big(M_h + \tfrac{\Delta t_{i-1}}{2} A_h + \tfrac{\Delta t_{i-1}}{2} N_h'(\vec{y}_i)\Big)^T \vec{\lambda}_i &= -\Big(-M_h + \tfrac{\Delta t_i}{2} A_h + \tfrac{\Delta t_i}{2} N_h'(\vec{y}_i)\Big)^T \vec{\lambda}_{i+1}\\ &\quad -\tfrac{\Delta t_{i-1}+\Delta t_i}{2}(M_h \vec{y}_i + (g_h)_i), \quad i = N, \ldots, 1,\end{aligned} \tag{6.45}$$

where $N_h'(\vec{y}_i)$ denotes the Jacobian of $N_h(\vec{y}_i)$. (Recall that $\Delta t_{N+1} = 0$.) Given the solution of (6.45), the gradient of the objective function $\widehat{f}$ in (**??**) can be obtained by computing the partial derivatives with respect to u_i of the Lagrangian (6.44). The gradient is given by

$$\nabla_u \widehat{f}(u) = \begin{pmatrix} \omega \frac{\Delta t_0}{2} Q_h \vec{u}_0 + \frac{\Delta t_0}{2} B_h^T \vec{\lambda}_1 \\ \omega \frac{\Delta t_0 + \Delta t_1}{2} Q_h \vec{u}_1 + B_h^T (\frac{\Delta t_0}{2} \vec{\lambda}_1 + \frac{\Delta t_1}{2} \vec{\lambda}_2) \\ \vdots \\ \omega \frac{\Delta t_{N-1} + \Delta t_N}{2} Q_h \vec{u}_N + B_h^T (\frac{\Delta t_{N-1}}{2} \vec{\lambda}_N + \frac{\Delta t_N}{2} \vec{\lambda}_{N+1}) \\ \omega \frac{\Delta t_N}{2} Q_h \vec{u}_{N+1} + \frac{\Delta t_N}{2} B_h^T \vec{\lambda}_{N+1} \end{pmatrix}. \tag{6.46}$$

(Recall that $\Delta t_{-1} = \Delta t_{N+1} = 0$.)

We summarize the gradient computation using adjoints in the following algorithm.

Algorithm 6.4.1 (Gradient Computation Using Adjoints)

1. *Given* $\vec{u}_1, \ldots, \vec{u}_{N+1}$*, and* $\vec{y}_0$ *compute* $\vec{y}_1, \ldots, \vec{y}_{N+1}$ *by solving*

$$\begin{aligned}&\Big(M_h + \frac{\Delta t_i}{2} A_h\Big)\vec{y}_{i+1} + \frac{\Delta t_i}{2} N_h(\vec{y}_{i+1})\\ &= -\Big(-M_h + \frac{\Delta t_i}{2} A_h\Big)\vec{y}_i - \frac{\Delta t_i}{2} N_h(\vec{y}_i) - \frac{\Delta t_i}{2} B_h(\vec{u}_{i+1} + \vec{u}_i) + \frac{\Delta t_i}{2}\Big(r_h(t_i) + r_h(t_{i+1})\Big),\end{aligned}$$

for $i = 0, \ldots, N$.

2. *Compute* $\vec{\lambda}_{N+1}, \ldots, \vec{\lambda}_1$ *by solving*

$$\begin{aligned}\Big(M_h + \frac{\Delta t_N}{2} A_h + \frac{\Delta t_N}{2} N_h'(\vec{y}_{N+1})\Big)^T \vec{\lambda}_{N+1} &= -\frac{\Delta t_N}{2}(M_h \vec{y}_{N+1} + (g_h)_{N+1}),\\ \Big(M_h + \frac{\Delta t_{i-1}}{2} A_h + \frac{\Delta t_{i-1}}{2} N_h'(\vec{y}_i)\Big)^T \vec{\lambda}_i &= -\Big(-M_h + \frac{\Delta t_i}{2} A_h + \frac{\Delta t_i}{2} N_h'(\vec{y}_i)\Big)^T \vec{\lambda}_{i+1}\\ &\quad -\frac{\Delta t_{i-1} + \Delta t_i}{2}(M_h \vec{y}_i + (g_h)_i),\end{aligned}$$

for $i = N, \ldots, 1$.

3. *Compute* $\nabla_u \widehat{f}(u)$ *from* (6.46).

We conclude by adapting Algorithms 6.2.4 and **??** to our problem. Since the the objective function (6.42) is quadratic and the implicit constraints (6.43) are quadratic in y and linear in u, most of the second derivative terms are zero. The multiplication of the Hessian $\nabla_u^2 \widehat{f}(u)$ times vector v computation can be performed using the following algorithm. In step 4 of the following algorithm we use that $N_h(y)$ is quadratic. Hence $\frac{d}{dy}(N_h'(\vec{y})^T \vec{\lambda})\vec{w} = N_h'(\vec{w})^T \vec{\lambda}$.

Algorithm 6.4.2 (Hessian–Times–Vector Computation Using Adjoints)

1. *Given* $\vec{u}_1, \ldots, \vec{u}_{N+1}$, *and* $\vec{y}_0$ *compute* $\vec{y}_1, \ldots, \vec{y}_{N+1}$ *by solving*

$$\begin{aligned} &\Big(M_h + \frac{\Delta t_i}{2} A_h\Big)\vec{y}_{i+1} + \frac{\Delta t_i}{2} N_h(\vec{y}_{i+1}) \\ &\quad = \; -\Big(- M_h + \frac{\Delta t_i}{2} A_h\Big)\vec{y}_i - \frac{\Delta t_i}{2} N_h(\vec{y}_i) - \frac{\Delta t_i}{2} B_h(\vec{u}_{i+1} + \vec{u}_i) + \frac{\Delta t_i}{2}\Big(r_h(t_i) + r_h(t_{i+1})\Big), \end{aligned}$$

for $i = 0, \ldots, N$ *(if not done already).*

2. *Compute* $\vec{\lambda}_{N+1}, \ldots, \vec{\lambda}_1$ *by solving*

$$\begin{aligned} \Big(M_h + \frac{\Delta t_N}{2} A_h + \frac{\Delta t_N}{2} N_h'(\vec{y}_{N+1})\Big)^T \vec{\lambda}_{N+1} &= -\frac{\Delta t_N}{2}(M_h \vec{y}_{N+1} + (g_h)_{N+1}), \\ \Big(M_h + \frac{\Delta t_{i-1}}{2} A_h + \frac{\Delta t_{i-1}}{2} N_h'(\vec{y}_i)\Big)^T \vec{\lambda}_i &= -\Big(- M_h + \frac{\Delta t_i}{2} A_h + \frac{\Delta t_i}{2} N_h'(\vec{y}_i)\Big)^T \vec{\lambda}_{i+1} \\ &\quad - \frac{\Delta t_{i-1} + \Delta t_i}{2}(M_h \vec{y}_i + (g_h)_i), \end{aligned}$$

for $i = N, \ldots, 1$ *(if not done already).*

3. *Compute* $\vec{w}_1, \ldots, \vec{w}_{N+1}$ *from*

$$\Big(M_h + \frac{\Delta t_i}{2} A_h + \frac{\Delta t_i}{2} N_h'(\vec{y}_{i+1})\Big)\vec{w}_{i+1} = -\Big(-M_h + \frac{\Delta t_i}{2} A_h + \frac{\Delta t_i}{2} N_h'(\vec{y}_i)\Big)\vec{w}_i + \frac{\Delta t_i}{2} B_h(\vec{v}_i + \vec{v}_{i+1}),$$

$i = 0, \ldots, N$, *where* $\vec{w}_0 = 0$.

4. *Compute* $\vec{p}_{N+1}, \ldots, \vec{p}_1$ *by solving*

$$\begin{aligned} \Big(M_h + \frac{\Delta t_N}{2} A_h + \frac{\Delta t_N}{2} N_h'(\vec{y}_{N+1})\Big)^T \vec{p}_{N+1} &= \frac{\Delta t_N}{2} M_h \vec{w}_{N+1} + \frac{\Delta t_N}{2} N_h'(\vec{w}_{N+1})^T \vec{\lambda}_{N+1}, \\ \Big(M_h + \frac{\Delta t_{i-1}}{2} A_h + \frac{\Delta t_{i-1}}{2} N_h'(\vec{y}_i)\Big)^T \vec{p}_i &= -\Big(- M_h + \frac{\Delta t_i}{2} A_h + \frac{\Delta t_i}{2} N_h'(\vec{y}_i)\Big)^T \vec{p}_{i+1} \\ &\quad + \frac{\Delta t_{i-1} + \Delta t_i}{2} M_h \vec{w}_i + N_h'(\vec{w}_i)^T \Big(\frac{\Delta t_{i-1}}{2}\vec{\lambda}_i + \frac{\Delta t_i}{2}\vec{\lambda}_{i+1}\Big), \end{aligned}$$

for $i = N, \ldots, 1$.

5. *Compute*

$$\nabla^2 \widehat{f}(u)v = \begin{pmatrix} \omega\frac{\Delta t_0}{2}Q_h\vec{v}_0 + \frac{\Delta t_0}{2}B_h^T\vec{p}_1 \\ \omega\frac{\Delta t_0+\Delta t_1}{2}Q_h\vec{v}_1 + B_h^T(\frac{\Delta t_0}{2}\vec{p}_1 + \frac{\Delta t_1}{2}\vec{p}_2) \\ \vdots \\ \omega\frac{\Delta t_{N-1}+\Delta t_N}{2}Q_h\vec{v}_N + B_h^T(\frac{\Delta t_{N-1}}{2}\vec{p}_N + \frac{\Delta t_N}{2}\vec{p}_{N+1}) \\ \omega\frac{\Delta t_N}{2}Q_h\vec{v}_{N+1} + \frac{\Delta t_N}{2}B_h^T\vec{p}_{N+1} \end{pmatrix}.$$

Example 6.4.3 For the spatial discretization we subdivide $[0,1]$ into n_x subintevals of length $\Delta x = 1/n_x$ and we define

$$\varphi_i(x) = \begin{cases} (\Delta x)^{-1}(x-(i-1)\Delta x) & x \in [(i-1)\Delta x, i\Delta x], \\ (\Delta x)^{-1}(-x+(i+1)\Delta x) & x \in [i\Delta x, (i+1)\Delta x], \\ 0 & \text{else} \end{cases} \qquad i = 1,\dots,n_y \stackrel{\text{def}}{=} n_x - 1,$$

$$\psi_i(x) = \begin{cases} (\Delta x)^{-1}(x-(i-2)\Delta x) & x \in [(i-2)\Delta x, (i-1)\Delta x], \\ (\Delta x)^{-1}(-x+i\Delta x) & x \in [(i-1)\Delta x, i\Delta x], \\ 0 & \text{else} \end{cases} \qquad i = 1,\dots,n_u \stackrel{\text{def}}{=} n_x + 1.$$

This choice yields

$$M_h = \frac{\Delta x}{6}\begin{pmatrix} 4 & 1 & & & \\ 1 & 4 & 1 & & \\ \ddots & \ddots & \ddots & & \\ & 1 & 4 & 1 \\ & & 1 & 4 \end{pmatrix} \in \mathbb{R}^{n_y \times n_y}, \quad A_h = \frac{\nu}{\Delta x}\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ \ddots & \ddots & \ddots & & \\ & -1 & 2 & -1 \\ & & -1 & 2 \end{pmatrix} \in \mathbb{R}^{n_y \times n_y},$$

$$N_h(\vec{y}(t)) = \frac{1}{6}\begin{pmatrix} y_1(t)y_2(t) + y_2^2(t) \\ -y_1^2(t) - y_1(t)y_2(t) + y_2(t)y_3(t) + y_3^2(t) \\ \vdots \\ -y_{i-1}^2(t) - y_{i-1}(t)y_i(t) + y_i(t)y_{i+1}(t) + y_{i+1}^2(t) \\ \vdots \\ -y_{n_y-2}^2(t) - y_{n_y-2}(t)y_{n_y-1}(t) + y_{n_y-1}(t)y_{n_y}(t) + y_{n_y}^2(t) \\ -y_{n_y-1}^2(t) - y_{n_y-1}(t)y_{n_y}(t) \end{pmatrix} \in \mathbb{R}^{n_y}$$

and

$$B_h = -\frac{\Delta x}{6}\begin{pmatrix} 1 & 4 & 1 & & & \\ & 1 & 4 & 1 & & \\ & \ddots & \ddots & \ddots & & \\ & & 1 & 4 & 1 & \\ & & & 1 & 4 & 1 \end{pmatrix} \in \mathbb{R}^{n_y \times n_u}, \quad Q_h = \frac{\Delta x}{6}\begin{pmatrix} 2 & 1 & & & \\ 1 & 4 & 1 & & \\ \ddots & \ddots & \ddots & & \\ & 1 & 4 & 1 \\ & & 1 & 2 \end{pmatrix} \in \mathbb{R}^{n_u \times n_u}.$$

The matrix $N'_h(\vec{y}(t)) \in \mathbb{R}^{n_y \times n_y}$ is shown in Figure 6.1. To approximate the integrals arising in the definition of $r_h(t)$ and $g_h(t)$ we apply the composite trapezoidal rule. This yields

$$\begin{aligned} (r_h(t))_i &= \Delta x \, r(i\Delta x, t), \\ (g_h(t))_i &= \Delta x \, \hat{y}(i\Delta x, t). \end{aligned}$$

We consider the optimal control problem (6.34), (6.35) with data $T = 1$, $\omega = 0.05$, $\nu = 0.01$, $r = 0$,

$$y_0(x) = \begin{cases} 1 & x \in (0, \frac{1}{2}], \\ 0 & \text{else}, \end{cases}$$

and $\hat{y}(x,t) = y_0(x)$, $t \in (0,T)$ (cf. [KV99]). For the discretization we use $n_x = 40$ spatial subintervals and 40 time steps, i.e., $\Delta t = 1/40$.

The solution y of the discretized(6.43) of Burgers' equation (6.35) with $u(x,t) = 0$ as well as the desired state are shown in Figure 6.2.

The solution u of the optimal control problem (6.34), (6.35), the corresponding solution $y(u_*)$ of the discretized Burgers' equation (6.35) and the solution $\lambda(u_*)$ of (6.45) are plotted in Figure 6.3.

The convergence history of the Newton–CG method with Armijo-line search applied to $\min \widehat{f}(u)$ is shown in Table 6.1. Here

$$\widehat{f}(u) = \sum_{i=0}^{N+1} \frac{\Delta t_{i-1} + \Delta t_i}{2} \left(\frac{1}{2} \vec{y}_i^T M_h \vec{y}_i + (g_h)_i^T \vec{y}_i + \frac{\omega}{2} \vec{u}_i^T Q_h \vec{u}_i \right)$$

(see (6.42)), where $\vec{y}_1, \ldots, \vec{y}_{N+1}$ is the solution of (6.43). In this version of the Newton–CG method the Newton system is solved using the conjugate gradient method such that the inexact Newton step s_k satisfies

$$\|\nabla^2 \widehat{f}(u_k) s_k + \nabla \widehat{f}(u_k)\|_2 \le \eta_k \|\nabla \widehat{f}(u_k)\|_2,$$

where $\eta_k = \min\{0.01, \|\nabla \widehat{f}(u_k)\|_2\}$.

Table 6.1: Performance of a Newton-CG method applied to the solution of (6.34), (6.35)

k	$\widehat{f}(u_k)$	$\|\nabla \widehat{f}(u_k)\|$	$\|s_k\|$	α_k	#CG iters
0	$-8.500121e-02$	$6.080307e-03$	$6.969352e+01$	0.5	7
1	$-1.785521e-01$	$1.434887e-03$	$1.757575e+01$	1.0	10
2	$-1.891822e-01$	$1.366862e-04$	$1.798857e+00$	1.0	17
3	$-1.892868e-01$	$1.853288e-06$	$2.274169e-02$	1.0	24

◇

Figure 6.1: The matrix $N_h'(\vec{y}(t))$

$$
N_h'(\vec{y}(t)) = \frac{1}{6}\begin{pmatrix}
y_2(t) & y_1(t)+2y_2(t) & & & \\
-2y_1(t)-y_2(t) & y_3(t)-y_1(t) & y_2(t)+2y_3(t) & & \\
\ddots & \ddots & \ddots & & \\
 & -2y_{i-1}(t)-y_i(t) & y_{i+1}(t)-y_{i-1}(t) & y_i(t)+2y_{i+1}(t) & \\
 & \ddots & \ddots & \ddots & \\
 & & -2y_{n_y-2}(t)-y_{n_y-1}(t) & y_{n_y}(t)-y_{n_y-2}(t) & y_{n_y-1}(t)+2y_{n_y}(t) \\
 & & & -2y_{n_y-1}(t)-y_{n_y}(t) & -y_{n_y-1}(t)
\end{pmatrix}
$$

Figure 6.2: Solution of Burgers' equation with $u = 0$ (no control) (left) and desired state $\hat{y}$ (right)

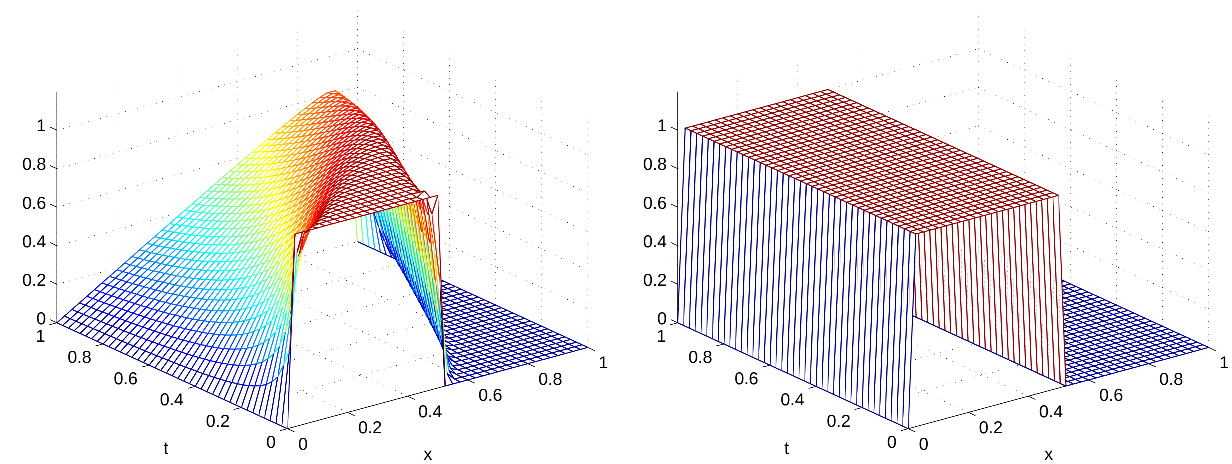

Figure 6.3: Optimal control u_* (upper left), corresponding solution $y(u_*)$ of Burgers' equation (upper right) and corresponding Lagrange multipliers $\lambda(u_*)$ (bottom)