DIPLOMA THESIS

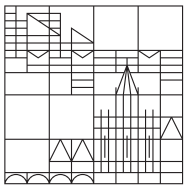
Trust Region POD for Optimal Boundary Control of a Semilinear Heat Equation

submitted by

Sabrina Rogg

at the

Universität Konstanz



Faculty of Sciences

Department of Mathematics and Statistics

Konstanz, 2014

Supervisor and Reviewer: Prof. Dr. Stefan Volkwein, University of Konstanz

2nd Reviewer: Prof. Dr. Ekkehard Sachs, University of Trier

Contents

1 Introduction and Outline

Optimal control problems governed by partial differential equations (PDEs) occur in various important scientific and technical fields. The PDE mathematically describes a certain process and couples a control variable u with a state variable y. The control influences the state in such a way that every control determines a unique associated state by the solution of the given PDE. The aim is to find a control with an associated state that minimizes a cost functional which mostly depends on both the state and the control.

In this thesis we consider an optimal control problem of a thermal process. The control describes a heating source that acts on the boundary of a given spatial domain. The control is applied for a fixed time period $[0, T]$, $T > 0$. At the beginning, the domain has a certain temperature distribution. We are searching for an optimal control such that the resulting final temperature distribution in the domain is the best possible approximation to a desired state. At the same time we keep control costs to a minimum.

In this work the results in [42, 43] are extended with respect to the following issues: The governing linear heat equation is replaced by a semilinear heat equation. The part of the equation which includes the nonlinearity is of type

$$c_p y_t - \Delta y + N(y(\cdot, \cdot)) = f \quad \text{in } Q,$$

with a real-valued function $N : \mathbb{R} \to \mathbb{R}$ and Q being the time-space cylinder of interest. In numerics we consider the cubic nonlinearities $N(y) = y^3$ and $N(y) = -0.5y^3$. In addition to this extension, the problem is formulated as a reduced problem by including the state equation in the formulation of the cost functional. Moreover, the boundary of the spatial domain is divided into a number of segments.

For numerical optimization we pursue the 'first optimize, then discretize' approach; compare [21, Chapter 3]. This means that the considered algorithms are formulated for a general Hilbert space setting. Afterwards, the discrete schemes are derived. The use of a classical discretization technique, we investigate a finite element (FE) Galerkin method, leads to several high-dimensional nonlinear and linear systems of equations. These have to be solved repeatedly. Hence, the numerical solution of the optimal control problem is generally time consuming.

In recent years, efficient model reduction techniques have been developed to obtain low-dimensional approximations of high quality. Besides the method of Proper Orthogonal Decomposition (POD), reduced basis methods have as well emerged as promising tools; see [15, 18, 38]. The method of balanced truncation shall also be mentioned for the sake of completeness; see [51]. The basic idea of POD is to replace the local FE basis functions for the applied Galerkin method by global and problem-dependent POD basis functions. If these POD basis functions properly represent system dynamics, only a few of them suffice to obtain satisfactory approximations.

We give an outline of the thesis with a detailed description of the chapters' contents. Chapters 2 to 5 establish the theoretical foundations for the numerical solution of the

given semilinear heat equation and the considered optimal control problem. All numerical results are presented in Chapter 6. The focus of this work lies on the numerical examples and their analysis.

Chapter 2
In the beginning we state the problem setting and name the required definitions. Unique weak solvability of the semilinear heat equation is shown and existence of optimal controls is proven. By viewing the state as a function of the control variable through the solution of the state equation, we derive the reduced cost functional. Thus, we can write the given optimal control problem as an equivalent reduced minimization problem for the control variable. For later purposes the computation of first- and second-order derivatives of the reduced cost functional is given in detail. To conclude the chapter, first- and second-order optimality conditions are investigated.

Chapter 3
For the numerical solution of the original optimal control problem we consider the equivalent reduced problem. We start by presenting a local inexact Newton method. The approximative Newton steps are computed with a conjugate gradient (CG) algorithm. Inexact Newton methods are well-known for their fast local convergence. But they are not globally convergent. First, we add a line search strategy for globalization of our method. An Armijo backtracking algorithm is chosen. This leads to a so-called line search Newton-CG (LSNCG) method. Note that the LSNCG procedure can just be proved to converge to stationary points. In our numerical experiments we assume that the obtained controls are optimal. The second section of this chapter deals with the required spatial and temporal discretization. We review the FE Galerkin method for discretizing the spatial variable. The time integration is carried out by the implicit Euler method.

Chapter 4
This chapter provides an introduction to reduced-order modeling using POD. First, the continuous version of the POD method is explained. Second, the numerically feasible discrete version of the POD approach is discussed. The computation of a POD basis and the derivation of the reduced-order models (ROMs) using a POD Galerkin method is explained. To fully discretize the POD based semi-discrete schemes we also use the implicit Euler method. The chapter ends with the investigation of two different versions of empirical interpolation.

The LSNCG strategy can now be applied by using the FE Galerkin discretization or the POD Galerkin method. Taking the latter technique one has to choose a control that is utilized to set up the POD basis. A good choice is essential for accurate approximations. Only then, few POD basis functions suffice to obtain good, so-called suboptimal controls. The dependence of accuracy of the ROMs on a somehow random choice certainly constitutes the main weakness of the POD method applied in optimal control. A POD basis which is computed from an optimal FE control is interpreted to be itself optimal because it yields the best results; compare also [16, 42, 43].

Chapter 5
To overcome the risk of receiving poor ROMs we present the approach of successively

improving the POD basis in the course of optimization by utilizing the updated control values. This is realized by embedding the reduced-order Newton-CG strategy in a trust region framework. By doing so, we ensure that the POD approximations are sufficiently accurate.

Chapter 6
In this chapter numerical results are presented and analyzed. Section 6.1 contains two examples for the solution of a semilinear heat equation with $N(y) = y^3$. In the first example the exact analytical solution is known. In the second example the initial condition (initial temperature distribution) is given by a step function. Numerical experiments for the solution of the optimal control problem are content of Section 6.2. This is doubtless the most important part of this thesis. Three different problems are considered. We investigate two examples with nonlinearity $N(y) = y^3$. In the first one the local Newton-CG procedure (using the FE Galerkin discretization) could be applied without a globalization strategy. But in the second one negative curvature in the Hessian of the reduced cost functional occurs. In a last example we replace the nonlinearity by $N(y) = -0.5y^3$.

Chapter 7
Finally, we discuss our results and draw a conclusion.

2 Semilinear Optimal Control Problem

This chapter consists of five sections. First of all, we specify the problem settings and name the required definitions. We expect readers to be familiar with basic concepts of functional analysis and of optimal control of partial differential equations (PDEs). In Section 2.2 we consider the solvability of the given state equation and in Section 2.3 we prove the existence of optimal controls. These two sections are based on [45, Chapter 5]. The detailed computation of first- and second-order derivatives is given in Section 2.4 and carried out analogous to [21, Section 1.6]. The derivatives are needed to name first- and second-order optimality conditions in Section 2.5 and they are required for Newton methods.

2.1 Problem Formulation

We directly start with the boundary control problem under consideration and name the specifications that are fixed to hold throughout this thesis.

$$
\begin{aligned}
&\min J(y,u) := \frac{1}{2}\int_\Omega \left|y(T)-y_d\right|^2 \mathrm{d}\boldsymbol{x} + \frac{1}{2}\sum_{k=1}^{\mathsf{k}} \gamma_k \int_0^T \left|u_k(t)\right|^2 \mathrm{d}t \\
&\text{subject to (s.t.)} \\
&\text{(SE)} \quad \begin{cases} c_p y_t(t,\boldsymbol{x}) - \Delta y(t,\boldsymbol{x}) + N(y(t,\boldsymbol{x})) = f(t,\boldsymbol{x}) & \text{for } (t,\boldsymbol{x}) \in Q, \\ \dfrac{\partial y}{\partial \nu}(t,\boldsymbol{x}) + q y(t,\boldsymbol{x}) = \displaystyle\sum_{k=1}^{\mathsf{k}} u_k(t)\chi_k(\boldsymbol{x}) & \text{for } (t,\boldsymbol{x}) \in \Sigma, \\ y(0,\boldsymbol{x}) = y_0(\boldsymbol{x}) & \text{for } x \in \Omega, \end{cases} \\
&\text{and} \\
&\qquad u_a(t) \le u(t) \le u_b(t) \text{ almost everywhere (a.e.) in } [0,T].
\end{aligned}
\tag{\mathbf{P}}
$$

The set $\Omega \subset \mathbb{R}^{m_\Omega}$, $m_\Omega \in \mathbb{N}_{>0}$, denotes an open and bounded spatial domain with Lipschitz boundary $\Gamma := \partial\Omega$. The boundary is divided into $\mathsf{k} \in \mathbb{N}$ disjunct segments Γ_k, $k = 1, \dots, \mathsf{k}$. The associated characteristic functions $\chi_k := \chi_{\Gamma_k}$, $k = 1, \dots, \mathsf{k}$, are the considered control shape functions. The vector $\nu \in \mathbb{R}^{m_\Omega}$ is the outward unit normal to Γ.

For a given final time $T > 0$ we consider the time-space cylinder $Q := (0,T) \times \Omega$ and set $\Sigma := (0,T) \times \Gamma$.

The function $y_d \in C(\bar{\Omega})$ denotes a desired state and $\gamma_k > 0$, $k = 1, \dots, \mathsf{k}$, are regularization parameters.

The function $y_0 \in C(\bar{\Omega})$ is a given initial condition, f belongs to $L^r(Q)$ with $r > m_\Omega/2+1$ and $q \ge 0$, $c_p > 0$ are given constants. The nonlinear function $N : \mathbb{R} \to \mathbb{R}$ must have certain properties that get specified when needed. In this thesis we mainly focus on $N(y) = y^3$. In numerics we additionally investigate the linear function $N(y) \equiv 0$ and the

nonlinearity $N(y) = -0.5y^3$. In case of $N(y) \equiv 0$ less restrictive assumptions are required with a larger state space; see point (2) of Remark 2.5.

The separable Hilbert space $U := L^2(0,T;\mathbb{R}^{\mathsf{k}})$ is the control space. Let us recall the inner product in U

$$\langle u, w\rangle_U := \int_0^T \langle u(t), w(t)\rangle_{\mathbb{R}^{\mathsf{k}}}\,\mathrm{d}t = \int_0^T \sum_{k=1}^{\mathsf{k}} u_k(t)w_k(t)\,\mathrm{d}t \quad \text{for all } u, w \in U. \tag{2.1}$$

We define the set of admissible controls by

$$U_{\mathsf{ad}} := \{u \in U \mid u_a(t) \le u(t) \le u_b(t) \text{ a.e. in } [0,T]\},$$

with $u_a, u_b \in L^\infty(0,T;\mathbb{R}^{\mathsf{k}})$, $u_a(t) \le u_b(t)$ a.e. in $[0,T]$. All inequalities between vectors are understood to hold componentwise. The space U_{ad} is a closed, convex and bounded subset of $L^\infty(0,T;\mathbb{R}^{\mathsf{k}})$. To get an idea, a component $u_k(t)$, $k \in \{1,\dots,\mathsf{k}\}$, of any $u \in U_{\mathsf{ad}}$ describes the control intensity on the corresponding boundary part Γ_k at time t.

Due to Robin boundary conditions in the state equation (SE) we consider the space $V := H^1(\Omega)$ which is densely and continuously (even compactly) embedded in $H := L^2(\Omega)$. It is well known that V and H are separable real Hilbert spaces. Using the Riesz representation theorem we identify H with its dual space H'; see, e.g., Theorem 1.4 in [21]. This yields the chain of continuous and dense embeddings $V \hookrightarrow H = H' \hookrightarrow V'$, a so-called Gelfand triple.

The space $L^2(0,T;V)$ denotes the space of (equivalence classes of) measurable Banach space valued functions $y : [0,T] \to V$ which are square integrable in the sense of Bochner. The quadratic cost functional J maps from $W(0,T) \times U$ to $\mathbb{R}$ with $W(0,T)$ defined as follows.

Definition 2.1. By $W(0,T)$ we denote the function space

$$W(0,T) := \{y \in L^2(0,T;V) \mid y_t \in L^2(0,T;V')\},$$

with y_t being the weak derivative of y with respect to time. It is equipped with the norm

$$\|y\|_{W(0,T)} := \sqrt{\|y\|^2_{L^2(0,T;V)} + \|y_t\|^2_{L^2(0,T;V')}}.$$

Later on we will use the following properties of $W(0,T)$:

- $W(0,T)$ is a Hilbert space with inner product related to the above norm.
- $W(0,T)$ is continuously embedded into $C([0,T];H)$, the space of continuous functions from $[0,T]$ to H. Hence, there exists a constant $c_W > 0$ such that

  $$\|y(t)\|_H \le \|y\|_{C([0,T];H)} \le c_W\|y\|_{W(0,T)} \tag{2.2}$$

  holds for any $t \in [0,T]$ and any function $y \in W(0,T)$.
- $\langle y_t(t), \varphi\rangle_{V',V} = \frac{\mathrm{d}}{\mathrm{d}t}\langle y(t), \varphi\rangle_H$ holds for $y \in W(0,T)$, $\varphi \in V$ and a.e. in $[0,T]$.

- The formula of integration by parts

$$\int_0^T \langle y_t(t), p(t)\rangle_{V',V}\,\mathrm{d}t = \langle y(T), p(T)\rangle_H - \langle y(0), v(0)\rangle_H - \int_0^T \langle p_t(t), y(t)\rangle_{V',V}\,\mathrm{d}t$$

  holds for $y, p \in W(0,T)$.

The corresponding proofs can, for instance, be found in [11, pp. 473-477].

In the following sections we will see that $Y := W(0,T) \cap C(\bar{Q})$ is an appropriate state space for (**P**).

2.2 Solvability of the State Equation

To get unique weak solvability of (SE) we introduce the assumptions

(**A1**) $N : \mathbb{R} \to \mathbb{R}$ is locally Lipschitz continuous and

(**A2**) monotone increasing.

Remark 2.2. (1) If the nonlinearity N additionally depends on the variables $(t, \boldsymbol{x})$, then $|N(t, \boldsymbol{x}, y)|$ must be uniformly bounded at $y = 0$ for almost all (f.a.a.) $(t, \boldsymbol{x}) \in Q$; see condition (5.2) in [45].

(2) The nonlinearity $N(y) = y^3$ satisfies assumptions (**A1**) and (**A2**). The equality $N(y_1) - N(y_2) = y_1^3 - y_2^3 = (y_1^2 + y_1y_2 + y_2^2)(y_1 - y_2)$ yields (**A1**). Assumption (**A2**) follows from $N'(y) = 3y^2 \geq 0$.

By a weak solution to (SE) we mean an element $y \in W(0,T) \cap L^\infty(Q)$ which satisfies the initial condition $y(0) = y_0$ in H as well as the variational equation or weak formulation

$$\begin{aligned}\int_0^T \int_\Omega c_p y_t \varphi \,\mathrm{d}\boldsymbol{x}\mathrm{d}t + \int_0^T \int_\Omega \nabla y \cdot \nabla \varphi + N(y)\varphi \,\mathrm{d}\boldsymbol{x}\mathrm{d}t + q\int_0^T \int_\Gamma y\varphi \,\mathrm{d}\boldsymbol{x}\mathrm{d}t \\ = \int_0^T \int_\Omega f\varphi \,\mathrm{d}\boldsymbol{x}\mathrm{d}t + \int_0^T \sum_{k=1}^{\mathsf{k}} u_k \int_\Gamma \chi_k \varphi \,\mathrm{d}\boldsymbol{x}\mathrm{d}t \quad \text{for all } \varphi \in L^2(0,T;V);\end{aligned} \tag{2.3}$$

compare [21, 45]. To improve readability we have omitted the arguments of the functions.

Remark 2.3. (1) The initial condition is meaningful for any $y \in W(0,T)$ due to the embedding $W(0,T) \hookrightarrow C([0,T];H)$.

(2) For an arbitrary element $y \in W(0,T)$ the integral $\int_0^T \int_\Omega N(y)\varphi \,\mathrm{d}\boldsymbol{x}\mathrm{d}t$ might not be well-defined. But $y \in W(0,T) \cap L^\infty(Q)$ and assumption (**A1**) provide boundedness.

The following theorem is based on Theorem 5.5 in [45] where Tröltzsch considers a more general semilinear parabolic PDE than the given heat equation.

Theorem 2.4. *Suppose that* (**A1**)-(**A2**) *hold and let* $u \in L^s(0,T;\mathbb{R}^{\mathsf{k}})$ *with* $s > m_\Omega + 1$. *For every* $y_0 \in L^\infty(\Omega)$ *there exists a unique weak solution* $y \in W(0,T) \cap L^\infty(Q)$ *to* (SE). *If* $y_0 \in C(\bar{\Omega})$, *this weak solution is continuous on* $\bar{Q}$, *i.e.* $y \in Y$ *is satisfied. Then the estimate*

$$\|y\|_{W(0,T)} + \|y\|_{C(\bar{Q})} \leq c_\infty \left(\|f - N(0)\|_{L^r(Q)} + \|u\|_{L^s(0,T;\mathbb{R}^{\mathsf{k}})} + \|y_0\|_{C(\bar{\Omega})} \right) \tag{2.4}$$

holds for a constant $c_\infty > 0$, *which is independent of* f, N, u, y_0, q.

Proof. Let $s > m_\Omega + 1$ and define the continuous linear operator $B_s : L^s(0,T;\mathbb{R}^{\mathsf{k}}) \to L^s(\Sigma)$ by $(B_s u)(t,\boldsymbol{x}) := \sum_{k=1}^{\mathsf{k}} u_k(t)\chi_k(\boldsymbol{x})$ f.a.a. $(t,\boldsymbol{x}) \in \Sigma$. The characteristic functions χ_k, $k = 1,\dots,\mathsf{k}$, belong to $L^\infty(\Gamma)$ so that $B_s u$ lies in $L^s(0,T;L^\infty(\Gamma)) \hookrightarrow L^s(\Sigma)$ for any $u \in L^s(0,T;\mathbb{R}^{\mathsf{k}})$. Inequality $\|B_s u\|_{L^s(\Sigma)} \le c_s \|u\|_{L^s(0,T;\mathbb{R}^{\mathsf{k}})}$ with a constant c_s independent of u and Theorem 5.5 in [45] yield the claim.

□

Remark 2.5. (1) Let $s > m_\Omega + 1$ and $u \in L^s(0,T;\mathbb{R}^{\mathsf{k}}) \supset U_{\mathsf{ad}}$. Remember that $y_0 \in C(\bar{\Omega})$ had been specified. The uniquely determined weak solution $y = y(u)$ to (SE) is said to be the state associated with u. We introduce the control-to-state operator $u \in L^s(0,T;\mathbb{R}^{\mathsf{k}}) \mapsto y(u) \in Y$.

(2) The state equation is linear in case of a linear function N. Then $W(0,T)$ is the standard state space and it suffices that the right-hand sides in (SE) are L^2-functions including the control; see [42, 45].

(3) In addition to the basic assumptions, let Ω be convex and let $y_0 \in H^2(\Omega) \hookrightarrow C(\bar{\Omega})$. Suppose $u \in L^\infty(0,T;\mathbb{R}^{\mathsf{k}})$, for instance $u \in U_{\mathsf{ad}}$. Using a so-called 'bootstrap' argument we can improve the regularity of the state $y = y(u) \in Y$: We write (SE) as a linear equation of type $c_p y_t - \Delta y = \tilde{f}$ by defining $\tilde{f} := f - N(y(\cdot,\cdot)) \in L^2(0,T;H)$. Theorem 5 in [8, p. 382] gives $y \in L^2(0,T;H^2(\Omega) \cap V) \cap H^1(0,T;V)$. Hence, $y \in H^1(0,T;V) \hookrightarrow C([0,T];V)$ holds.

2.3 Solvability of the Optimal Control Problem

In this thesis it is of central importance to formulate (**P**) as a reduced problem by including the state equation in the formulation of the cost functional: We denote by

$$\hat{J}(u) := J(y(u),u)$$

the reduced cost functional. Therewith the reduced problem is given by

$$\min_{u \in U_{\mathsf{ad}}} \hat{J}(u). \tag{$\hat{\mathbf{P}}$}$$

Definition 2.6. We call $\bar{u} \in U_{\mathsf{ad}}$ an optimal control for problem (**P**) and $\bar{y} = y(\bar{u})$ the associated optimal state if

$$\hat{J}(\bar{u}) = J(y(\bar{u}),\bar{u}) \le J(y(u),u) = \hat{J}(u) \quad \text{for all } u \in U_{\mathsf{ad}}. \tag{2.5}$$

We say that $\bar{u} \in U_{\mathsf{ad}}$ is a locally optimal control for (**P**) in the sense of $L^s(0,T;\mathbb{R}^{\mathsf{k}})$ if there exists an $\varepsilon > 0$ such that (2.5) holds for all $u \in U_{\mathsf{ad}}$ with $\|u - \bar{u}\|_{L^s(0,T;\mathbb{R}^{\mathsf{k}})} \le \varepsilon$.

Theorem 2.7. *Under the assumptions* (**A1**)-(**A2**) *above,* (**P**) *possesses at least one optimal control $\bar{u}$ with associated optimal state $\bar{y} = y(\bar{u})$.*

Proof. We follow the lines of proof of Theorem 5.7 in [45].

1. *Uniform boundedness of the states associated with controls in U_{ad}:*
   Let $s > m_\Omega + 1$. Since U_{ad} is a bounded subset of $L^\infty(0,T;\mathbb{R}^{\mathsf{k}})$ it is bounded in any

space $L^s(0,T;\mathbb{R}^{\mathsf{k}})$. Estimate (2.4) yields the existence of some $M > 0$ such that

$$\|y(u)\|_{C(\bar{Q})} \leq M \quad \text{for all } u \in U_{\mathsf{ad}}. \tag{2.6}$$

2. *Find a candidate $\bar{u} \in U_{\mathsf{ad}}$ for an optimal control*:
The infimum
$$J^* := \inf_{u \in U_{\mathsf{ad}}} J(y(u), u)$$
exists due to $J \geq 0$ and $U_{\mathsf{ad}} \neq \emptyset$. Let $\{u_n\}_{n\in\mathbb{N}} \subset U_{\mathsf{ad}}$ be a minimizing sequence, i.e.
$$\lim_{n\to\infty} J(y(u_n), u_n) = J^*,$$
and define $y_n := y(u_n)$, $n \in \mathbb{N}$. Recall that the set U_{ad} is convex, closed and bounded. By viewing U_{ad} as a subset of the reflexive Banach space $L^s(0,T;\mathbb{R}^{\mathsf{k}})$ we can deduce that U_{ad} is weakly sequentially compact; see [45, Theorem 2.11]. Consequently, there exists a subsequence, without loss of generality we can choose $\{u_n\}_{n\in\mathbb{N}}$ itself, that converges weakly in $L^s(0,T;\mathbb{R}^{\mathsf{k}})$ to some limit $\bar{u}$ that belongs to U_{ad}:
$$u_n \rightharpoonup \bar{u} \text{ as } n \to \infty.$$

3. *The state sequence $\{y_n\}_{n\in\mathbb{N}}$ converges strongly to some $\bar{y} \in C(\bar{Q})$*:
Let
$$z_n(t,\boldsymbol{x}) := -N(y_n(t,\boldsymbol{x})),\ (t,\boldsymbol{x}) \in Q \text{ a.e.}, n \in \mathbb{N}.$$
Estimate (2.6) and assumption (**A1**) ensure that $\{z_n\}_{n\in\mathbb{N}}$ is uniformly bounded in $L^r(Q)$. Hence, we find a subsequence, once more we choose the sequence itself, that converges weakly in $L^r(Q)$:
$$z_n \rightharpoonup z \in L^r(Q) \text{ as } n \to \infty.$$
Now, we use the operator B_s from the proof of Theorem 2.4. Let $n \in \mathbb{N}$. The semilinear heat equation can be written as a linear problem:
$$\begin{aligned} c_p y_{n,t} - \Delta y_n &= z_n + f && \text{in } Q, \\ \partial_\nu y_n + q y_n &= B_s u_n && \text{on } \Sigma, \\ y_n(0) &= y_0 && \text{in } \Omega. \end{aligned} \tag{2.7}$$
A continuous linear operator is weakly continuous; compare [45, p. 45]. This gives $B_s u_n \rightharpoonup B_s \bar{u}$ as $n \to \infty$. System (2.7) possesses a weakly continuous solution operator from $L^2(Q) \times L^2(\Sigma)$ to $W(0,T)$; see [45, Theorem 3.12]. Hence, we can first conclude weak convergence in $W(0,T)$ and afterwards strong convergence in $C(\bar{Q})$ to some $\bar{y} \in C(\bar{Q})$; for more details see the proof of Theorem 5.7 in [45].

4. *$\bar{y}$ is the weak solution associated with $\bar{u}$*:
Strong convergence of the state sequence $\{y_n\}_{n\in\mathbb{N}}$ in $C(\bar{Q})$ provides that $\bar{y}$ satisfies the initial condition. Assumption (**A1**) yields
$$N(y_n) \to N(\bar{y}) \quad \text{strongly in } L^\infty(Q) \text{ and in } L^2(Q).$$
We insert y_n, u_n into the weak formulation (2.3) and pass to the limit as $n \to \infty$ to

see that $\bar{y} = y(\bar{u})$ holds.

5. *Optimality of $\bar{u}$*:
   We can decompose J as
   $$J(y,u) = J_1(y) + J_2(u)$$
   with
   $$J_1 : W(0,T) \to \mathbb{R},\ J_1(y) := \frac{1}{2}\int_\Omega |y(T) - y_d|^2 \,\mathrm{d}\boldsymbol{x},$$
   $$J_2 : U \to \mathbb{R},\ J_2(u) := \frac{1}{2}\sum_{k=1}^{\mathsf{k}} \gamma_k \int_0^T |u_k(t)|^2 \,\mathrm{d}t.$$
   Since $y_n \to \bar{y}$ but $u_n \rightharpoonup \bar{u}$ as $n \to \infty$ we treat the functions J_1 and J_2 separately. Note that nonlinear continuous functions are not neccessarily weakly continuous. The function J_2 is convex and consequently weakly lower semicontinuous, see [45, Theorem 2.12]. I.e. we have
   $$\liminf_{n\to\infty} J_2(u_n) \geq J_2(\bar{u}) \text{ as } u_n \rightharpoonup \bar{u}.$$
   The following estimate finishes the proof:
   $$\begin{aligned} J^* &= \lim_{n\to\infty} J(y_n,u_n) = \lim_{n\to\infty} J_1(y_n) + \liminf_{n\to\infty} J_2(u_n) \\ &= J_1(\bar{y}) + \liminf_{n\to\infty} J_2(u_n) \geq J_1(\bar{y}) + J_2(\bar{u}) = J(\bar{y},\bar{u}). \end{aligned}$$
   Obviously, to show optimality of $(\bar{y},\bar{u})$ we did not need the specific structure of the cost functional. Lower semicontinuity of J would have been sufficient.

□

Remark 2.8. The cost functional is convex. If (SE) is linear, problem (**P**) is strictly convex with respect to u. But in case of a semilinear equation it might be non-convex. Hence, there might exist several (local) optimal controls and further assumptions would be necessary to prove uniqueness; compare, e.g., [45].

2.4 First- and Second-Order Derivatives

For the numerical solution of the original problem (**P**) we will work with the equivalent nonlinear reduced problem ($\hat{\mathbf{P}}$). We turn to the first- and second-order derivatives of the reduced cost functional. For an introduction to the generalization of the notion of differentiability to Banach spaces we refer to [21, 45].

First, we name the derivatives of the cost functional $J : W(0,T) \times U \to \mathbb{R}$. Further below, the derivatives of the reduced cost functional $\hat{J}$ are computed following a Lagrangian function based approach. The chain rule, see [45, Theorem 2.20], is applied. This requires the derivatives of the control-to-state operator. We restrict the dimension of the spatial domain to $m_\Omega := 2$ but we will point out when this restriction is actually needed.

Recall the Hilbert space $U = L^2(0,T;\mathbb{R}^{\mathsf{k}})$. We identify U' with U via $\langle\cdot,\cdot\rangle_{U',U} = \langle\cdot,\cdot\rangle_U$. Let $Y_1 := W(0,T)$ and $U_1 := L^\infty(0,T;\mathbb{R}^{\mathsf{k}})$.

Cost functional:
Let $y, v, v_1, v_2 \in W(0,T)$ and $u, w, w_1, w_2 \in U$. The Fréchet derivatives of J are given by

$$\begin{aligned}
\langle J_y(y,u), v\rangle_{Y_1',Y_1} &= \langle y(T) - y_d, v(T)\rangle_H,\\
\langle J_{yy}(y,u)v_2, v_1\rangle_{Y_1',Y_1} &= \langle v_1(T), v_2(T)\rangle_H,\\
\langle J_u(y,u), w\rangle_U &= \int_0^T \sum_{k=1}^{\mathsf{k}} \big(\gamma_k u_k(t)\big) w_k(t)\,\mathrm{d}t,\\
\langle J_{uu}(y,u)w_2, w_1\rangle_U &= \int_0^T \sum_{k=1}^{\mathsf{k}} \big(\gamma_k w_{2k}(t)\big) w_{1k}(t)\,\mathrm{d}t,
\end{aligned}$$

while the second-order mixed derivatives vanish. The linear mapping $y \in W(0,T) \mapsto y(T) \in H$ is continuous due to the embedding $W(0,T) \hookrightarrow C([0,T];H)$.

The Riesz representation for $J_u(y,u)$ is directly visible within the third of the above equations: For $t \in [0,T]$ a.e. it holds

$$\big(J_u(y,u)\big)(t) = D_\gamma u(t) \quad \text{with } D_\gamma := \operatorname{diag}(\gamma_1, \ldots, \gamma_{\mathsf{k}}), \tag{2.8}$$

and thus

$$\big(J_{uu}(y,u)w\big)(t) = D_\gamma w(t). \tag{2.9}$$

Remark 2.9. J is twice continuously Fréchet differentiable with Lipschitz continuous second-order derivative because $J_{uu}(y,u)$ and $J_{yy}(y,u)$ are both independent of (y,u).

We continue by considering the control-to-state operator. To obtain the desired differentiability we require

(**A3**) N is twice differentiable with locally Lipschitz continuous second-order derivative.

Assumption (**A3**) implies local Lipschitz continuity of N' and N by using the mean value theorem. Therefore assumption (**A1**) holds.

Remark 2.10. The second-order derivative $N''(y) = 6y$ of the function $N(y) = y^3$ is obviously globally Lipschitz continuous.

The given control-to-state operator is differentiable as a mapping from $L^s(0,T;\mathbb{R}^{\mathsf{k}})$ to Y with $s > m_\Omega + 1$; compare [45, Chapter 5]. In advance of the derivative computation we give the following theorem, where we restrict ourselves to $L^\infty(0,T;\mathbb{R}^{\mathsf{k}}) \supset U_{\mathsf{ad}}$.

Theorem 2.11. *Suppose that* (**A2**)-(**A3**) *hold. Then, the control-to-state operator is twice continuously Fréchet differentiable as a function from* $L^\infty(0,T;\mathbb{R}^{\mathsf{k}})$ *to* Y.

Proof. We have the continuous linear operator B_∞ from the proof of Theorem 2.4 so that Theorem 5.15 in [45] yields the claim. □

By the chain rule it follows:

Corollary 2.12. *With* (**A2**)-(**A3**) *holding the reduced cost functional* $\hat{J}$ *is twice continuously Fréchet differentiable on* $L^\infty(0,T;\mathbb{R}^{\mathsf{k}})$.

Unfortunately, the control-to-state operator and hence the reduced cost functional are not differentiable from the Hilbert space U to Y and to $\mathbb{R}$ respectively. Here, we encounter

the two-norm discrepancy being well-known for occuring in optimal control problems governed by semilinear parabolic PDEs; see [24, 45].

In [24] the problem is overcome by using continuous extensions. Let $u \in L^\infty(0,T;\mathbb{R}^{\mathsf{k}})$ arbitrary but fixed. Motivated by the argumentation given in [24] the following will be shown:

- We can view $y'(u) \in \mathcal{L}(L^\infty(0,T;\mathbb{R}^{\mathsf{k}}),Y)$ as continuous linear operator from U to $W(0,T)$ so that its dual operator maps continuously $W(0,T)'$ to $U' \sim U$.
- $\hat{J}'(u) \in L^\infty(0,T;\mathbb{R}^{\mathsf{k}})'$ belongs to $U' \sim U$.
- $\hat{J}''(u)$ maps continuously U to $U' \sim U$.

For the derivative computation we write (SE) elegantly as a nonlinear operator equation '$e(y,u) = 0$'. We use the two abbreviations $L^2(V) := L^2(0,T;V)$ and $L^2(V') := L^2(0,T;V')$ so that $L^2(V')' = L^2(V)$ holds. We introduce the required mappings:

- Define $F \in L^2(V')$ by

$$\langle F, \varphi\rangle_{L^2(V'),L^2(V)} = \int_0^T \langle F(t), \varphi(t)\rangle_{V',V}\,\mathrm{d}t := \int_0^T \int_\Omega f(t,\boldsymbol{x})\varphi(t,\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}\mathrm{d}t$$

  for $\varphi \in L^2(V)$.

- The continuous linear operator $\mathcal{A} : L^2(V) \to L^2(V')$,

$$\langle \mathcal{A}y, \varphi\rangle_{L^2(V'),L^2(V)} = \int_0^T \langle (\mathcal{A}y)(t), \varphi(t)\rangle_{V',V}\,\mathrm{d}t := \int_0^T a(y(t),\varphi(t))\,\mathrm{d}t$$

  for $y, \varphi \in L^2(V)$, with the symmetric and bounded bilinear form $a : V \times V \to \mathbb{R}$,

$$a(\varphi_1,\varphi_2) := \int_\Omega \nabla\varphi_1(\boldsymbol{x}) \cdot \nabla\varphi_2(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} + q\int_\Gamma \varphi_1(\boldsymbol{x})\varphi_2(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \quad \text{for } \varphi_1,\varphi_2 \in V.$$

  The boundedness of a is transferred to $\mathcal{A}$. Therefore, the operator $\mathcal{A}$ is indeed continuous; compare [21, p. 90] or see also [45].

- The continuous linear operator $\mathcal{B} : U \to L^2(0,T;V')$,

$$\begin{aligned}\langle \mathcal{B}u, \varphi\rangle_{L^2(V'),L^2(V)} &= \int_0^T \langle (\mathcal{B}u)(t), \varphi(t)\rangle_{V',V}\,\mathrm{d}t \\ &:= \int_0^T \sum_{k=1}^{\mathsf{k}} u_k(t)\int_\Gamma \chi_k(\boldsymbol{x})\varphi(t,\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}\mathrm{d}t \quad \text{for } u \in U, \varphi \in L^2(V).\end{aligned}$$

- The nonlinear operator $\mathcal{N} : L^\infty(Q) \to L^2(V')$,

$$\begin{aligned}\langle \mathcal{N}(y), \varphi\rangle_{L^2(V'),L^2(V)} &= \int_0^T \langle (\mathcal{N}(y))(t), \varphi(t)\rangle_{V',V}\,\mathrm{d}t \\ &:= \int_0^T \int_\Omega N(y(t,\boldsymbol{x}))\varphi(t,\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}\mathrm{d}t \quad \text{for } y \in L^\infty(Q), \varphi \in L^2(V).\end{aligned}$$

In addition, $y \in W(0,T)$ implies $y_t \in L^2(V')$. Hence, the weak formulation (2.3) defines the nonlinear operator equation

$$c_p y_t + \mathcal{A}y + \mathcal{N}(y) - F - \mathcal{B}u = 0 \quad \text{in } L^2(V'). \tag{2.10}$$

Let $Z := L^2(V') \times H$. We define the operator $e : Y \times U \to Z$ by

$$e(y,u) := \begin{pmatrix} e^1(y,u) \\ e^2(y,u) \end{pmatrix} := \begin{pmatrix} c_p y_t + \mathcal{A}y + \mathcal{N}(y) - F - \mathcal{B}u \\ y(0) - y_0 \end{pmatrix}.$$

Recall that a linear and bounded operator is Fréchet differentiable and that the derivative is given by the operator itself; see [45]. Thus, concerning differentiability the only delicate term in the above definition is the nonlinear operator $\mathcal{N}$.

Lemma 2.13. *With* (**A3**) *holding the operator* $\mathcal{N} : L^\infty(Q) \to L^2(V')$ *is twice continuously Fréchet differentiable. The action of the derivatives reads*

$$\langle \mathcal{N}'(y)v, \varphi \rangle_{L^2(V'),L^2(V)} = \int_0^T \int_\Omega N'(y(t,\boldsymbol{x}))v(t,\boldsymbol{x})\varphi(t,\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x}\mathrm{d}t, \tag{2.11}$$

$$\langle \mathcal{N}''(y)(v_1,v_2), \varphi \rangle_{L^2(V'),L^2(V)} = \int_0^T \int_\Omega N''(y(t,\boldsymbol{x}))v_1(t,\boldsymbol{x})v_2(t,\boldsymbol{x})\varphi(t,\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x}\mathrm{d}t, \tag{2.12}$$

for all $y, v, v_1, v_2 \in L^\infty(Q), \varphi \in L^2(V)$. *Moreover, for any* $y \in L^\infty(Q)$ *the derivates* $\mathcal{N}'(y)$ *and* $\mathcal{N}''(y)$ *can be applied to elements* $v, v_1, v_2 \in W(0,T)$.

Proof. First, one has to verify that the expressions above represent the Fréchet derivatives. This can be shown by using the estimation techniques from [45, Sections 4.3, 4.9] where Tröltzsch considers Nemytskii operators and their first- and second-order derivatives as mappings from $L^\infty(Q)$ to $L^\infty(Q)$.

We briefly show that the derivatives can be continuously extended. Let $y \in L^\infty(Q)$. Local Lipschitz continuity of N' and N'' implies $N'(y(\cdot,\cdot)), N''(y(\cdot,\cdot)) \in L^\infty(Q)$. This is why

$$\Big| \int_0^T \int_\Omega N'(y(t,\boldsymbol{x}))v(t,\boldsymbol{x})\varphi(t,\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x}\mathrm{d}t \Big| \le \big\| N'(y(\cdot,\cdot)) \big\|_{L^\infty(Q)} \int_0^T \int_\Omega \big| v(t,\boldsymbol{x})\varphi(t,\boldsymbol{x}) \big| \,\mathrm{d}\boldsymbol{x}\mathrm{d}t$$

is bounded for $v \in W(0,T) \subset L^2(Q)$, $\varphi \in L^2(V)$. For the second-order derivative we obtain

$$\begin{aligned} &\Big| \int_0^T \int_\Omega N''(y(t,\boldsymbol{x}))v_1(t,\boldsymbol{x})v_2(t,\boldsymbol{x})\varphi(t,\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x}\mathrm{d}t \Big| \\ &\qquad \le \big\| N''(y(\cdot,\cdot)) \big\|_{L^\infty(Q)} \int_0^T \|v_1(t)\|_H \|v_2(t)\|_{L^4(\Omega)} \|\varphi(t)\|_{L^4(\Omega)} \,\mathrm{d}t \\ &\qquad \le \big\| N''(y(\cdot,\cdot)) \big\|_{L^\infty(Q)} \big\| v_1 \big\|_{C([0,T];H)} \big\| v_2 \big\|_{L^2(0,T;L^4(\Omega))} \big\| \varphi \big\|_{L^2(0,T;L^4(\Omega))} < \infty \end{aligned}$$

for $v_1, v_2 \in W(0,T)$ and $\varphi \in L^2(V)$. The first inequality follows from an extension of Hölders inequality; see [21, Lemma 1.13]. Boundedness is given due to $W(0,T) \hookrightarrow C([0,T];H)$ and $L^2(V) \hookrightarrow L^2(0,T;L^q(\Omega))$ for $2 \le q \le 6$. The latter embedding is true for $m_\Omega \le 3$ by the Sobolev embedding theorem; see [21, Theorem 1.14]. □

Now, we can name the derivatives of the **operator e:**

$$e'(y,u)(v,w) = \underbrace{\begin{pmatrix} c_p v_t + \mathcal{A}v + \mathcal{N}'(y)v \\ v(0) \end{pmatrix}}_{=e_y(y,u)v} + \underbrace{\begin{pmatrix} -\mathcal{B}w \\ 0 \end{pmatrix}}_{=e_u(y,u)w},$$

$$e''(y,u)((v_1,w_1),(v_2,w_2)) = \begin{pmatrix} \mathcal{N}''(y)(v_1,v_2) \\ 0 \end{pmatrix},$$

for $y, v, v_1, v_2 \in Y$ and $u, w, w_1, w_2 \in U$.

Remark 2.14. Let $(y,u) \in Y \times U$. The above formula for $e_y(y,u)v$ and Lemma 2.13 show that $e_y(y,u)$ can be viewed as a continuous linear operator from $W(0,T)$ to $Z = L^2(V') \times H$. So, its dual operator maps continuously $Z' = L^2(V) \times H$ to $W(0,T)'$.

Control-to-state operator:
From the chain rule and Theorem 2.11, it follows that the equation

$$e(y(u),u) = 0$$

can be differentiated in a direction $w \in L^\infty(0,T;\mathbb{R}^{\mathsf{k}})$. This yields

$$e_y(y(u),u)y'(u)w + e_u(y(u),u)w = 0. \tag{2.13}$$

Thus, the sensitivity $v := y'(u)w$ is given by the solution to the

linearized state equation

$$e_y(y(u),u)v = -e_u(y(u),u)w.$$

Let $y_u := y(u)$. Written in expanded form the linearized state equation reads

$$\begin{pmatrix} c_p v_t + \mathcal{A}v + \mathcal{N}'(y_u)v \\ v(0) \end{pmatrix} = \begin{pmatrix} \mathcal{B}w \\ 0 \end{pmatrix}.$$

This is the weak formulation of

$$\text{(LSE)} \quad \begin{cases} c_p v_t - \Delta v + N'(y_u(\cdot,\cdot))v = 0 & \text{in } Q, \\ \dfrac{\partial v}{\partial \nu} + qv = \displaystyle\sum_{k=1}^{\mathsf{k}} w_k \chi_k & \text{on } \Sigma, \\ v(0) = 0 & \text{in } \Omega. \end{cases}$$

We investigate the solvability of (LSE):

1. $w \in U \Rightarrow v \in W(0,T)$:
   The operator $\mathcal{B}$ is linear and continuous. From (**A2**)-(**A3**) we can deduce that the function $(t,\boldsymbol{x}) \mapsto N'(y_u(t,\boldsymbol{x})) \geq 0$ belongs to $L^\infty(Q)$. Hence, the linearized state equation has a continuous linear solution operator $w \mapsto v$ from U to $W(0,T)$; see [11, Chapter XVIII].

2. $w \in L^\infty(0,T;\mathbb{R}^{\mathsf{k}}) \Rightarrow v \in Y$:
   If the control w belongs to $L^\infty(0,T;\mathbb{R}^{\mathsf{k}})$, we even obtain $v \in C(\bar{Q})$ and hence $y \in Y$; see [45, Chapter 5].

Remark 2.15. (1) The above point 2 would allow to apply the implicit function theorem, see [21, Theorem 1.41], to prove Theorem 2.11, i.e. differentiability of the control-to-state operator.

(2) Let $u \in L^\infty(0,T;\mathbb{R}^{\mathsf{k}})$. Point 1 above justifies to write

$$y'(u) = -e_y(y(u),u)^{-1}\, e_u(y(u),u) \in \mathcal{L}(U, W(0,T)). \tag{2.14}$$

Consequently, the dual operator $y'(u)^*$ maps continuously $W(0,T)'$ to U. Point 2 above yields $y'(u)w \in Y$, if $w \in L^\infty(0,T;\mathbb{R}^{\mathsf{k}})$.

(3) The operator $e'(y,u) = (e_y(y,u), e_u(y,u))$ is surjective for all $(y,u) \in Y \times U$ because the operator $e_y(y,u)$ is bijective. In order to see this, we consider the linearized state equation with an arbitrary right-hand side. Surjectivity of $e_y(y,u)$ follows if and only if for all $(g,v_0) \in Z$ there exists $v \in Y$ such that $e_y(y,u)v = (g,v_0)$. The reference from point 1 above yields the existence of a weak solution $v \in W(0,T)$ which is even unique. By a bootstrap argument the regularity of v can be improved such that $v \in C(\bar{Q})$ is satisfied.
Hence, a so-called regular point condition is fulfilled and provides the existence of a Lagrange muliplier $p = (p_1, p_2) \in Z'$ associated with (SE) in the context of Karush-Kuhn-Tucker theory; see [32, Theorem 4.1]. By variational arguments it follows that $p_{1,t}$ belongs to $L^2(V')$. Thus, we even have $p_1 \in W(0,T)$.

(4) Differentiating equation (2.13) with $w_1 := w$ once again in another direction $w_2 \in L^\infty(0,T;\mathbb{R}^{\mathsf{k}})$ yields an equation for the second-order derivative $y''(u)(w_1,w_2)$. We will not have to compute this derivative. But we need that $y''(u)$ can also be applied to elements $w_1, w_2 \in U$. Therefore, let us name a representation formula which is also stated in [45, Theorem 5.16]. The application $v := y''(u)(w_1,w_2)$ is the solution to

$$\begin{cases} c_p v_t - \Delta v + N'(y_u(\cdot,\cdot))v = -N''(y_u(\cdot,\cdot))v_1 v_2 & \text{in } Q, \\ \dfrac{\partial v}{\partial \nu} + qv = 0 & \text{on } \Sigma, \\ v(0) = 0 & \text{in } \Omega, \end{cases} \tag{2.15}$$

with $v_i = y'(u)w_i$, $i = 1,2$. For $v_1, v_2 \in W(0,T)$ we obtain $N''(y_u(\cdot,\cdot))v_1v_2 \in L^2(Q)$: The embedding $W(0,T) \hookrightarrow L^4(0,T;L^4(\Omega)) \sim L^4(Q)$ for $m_\Omega = 2$ gives

$$\begin{aligned} \int_0^T \int_\Omega \left|v_1(t,\boldsymbol{x})v_2(t,\boldsymbol{x})\right|^2 \mathrm{d}\boldsymbol{x}\mathrm{d}t &\le \int_0^T \|v_1(t)\|^2_{L^4(\Omega)} \|v_2(t)\|^2_{L^4(\Omega)}\, \mathrm{d}t \\ &\le \|v_1\|^2_{L^4(Q)} \|v_2\|^2_{L^4(Q)}. \end{aligned}$$

Thus, the reference from point 1 above ensures that equation (2.15) has a unique weak solution $v \in W(0,T)$. Let us mention that the use of bootstrapping even yields $v \in L^\infty(0,T;H^2(\Omega)) \cap H^1(0,T;H)$.

Reduced cost functional:
Now, we can compute $\hat{J}'(u)$ for any $u \in L^\infty(0,T;\mathbb{R}^{\mathsf{k}})$ using a Lagrangian function based approach.

We introduce the Lagrange function $L : Y \times L^\infty(0,T;\mathbb{R}^{\mathsf{k}}) \times Z' \to \mathbb{R}$ by

$$\begin{aligned} L(y,u,p) &:= J(y,u) + \langle p, e(y,u)\rangle_{Z',Z} \\ &= J(y,u) + \langle e^1(y,u), p_1\rangle_{L^2(V'),L^2(V)} + \langle p_2, e^2(y,u)\rangle_H, \end{aligned}$$

where $p = (p_1,p_2) \in Z' = L^2(V) \times H$.

For any $u \in L^\infty(0,T;\mathbb{R}^{\mathsf{k}})$ and $p \in Z'$ we have

$$\hat{J}(u) = J(y(u),u) = J(y(u),u) + \langle p, e(y(u),u)\rangle_{Z',Z} = L(y(u),u,p),$$

because $e(y(u),u) = 0$ holds. The first-order derivative of $\hat{J}$ thus reads

$$\langle \hat{J}'(u), w_1\rangle_{U_1',U_1} = \langle L_u(y(u),u,p), w_1\rangle_{U_1',U_1} + \langle L_y(y(u),u,p), y'(u)w_1\rangle_{Y',Y} \tag{2.16}$$

for $w_1 \in L^\infty(0,T;\mathbb{R}^{\mathsf{k}})$.

The left term $L_u(y(u),u,p)$ is given by

$$\begin{aligned} \langle L_u(y(u),u,p), w_1\rangle_{U_1',U_1} &= \langle J_u(y(u),u), w_1\rangle_U + \langle e_u^1(y(u),u)w_1, p_1\rangle_{L^2(V'),L^2(V)} \\ &= \langle J_u(y(u),u), w_1\rangle_U + \langle -\mathcal{B}w_1, p_1\rangle_{L^2(V'),L^2(V)} \\ &= \langle J_u(y(u),u) - \mathcal{B}^* p_1, w_1\rangle_U, \end{aligned}$$

with the dual operator $\mathcal{B}^*$ of $\mathcal{B}$. Thus, $L_u(y(u),u,p)$ can be identified with the element

$$L_u(y(u),u,p) = J_u(y(u),u) - \mathcal{B}^* p_1 \in U. \tag{2.17}$$

We determine the dual operator $\mathcal{B}^* : L^2(V) \to U$ of $\mathcal{B}$, satisfying

$$\langle \mathcal{B}u, \varphi\rangle_{L^2(V'),L^2(V)} = \langle u, \mathcal{B}^*\varphi\rangle_U \quad \text{for all } (u,\varphi) \in U \times L^2(V).$$

Actually, it can be directly read off from the definition of $\mathcal{B}$. We obtain

$$(\mathcal{B}^*\varphi)(t) = \begin{pmatrix} \int_\Gamma \chi_1(\boldsymbol{x})\varphi(t,\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \\ \vdots \\ \int_\Gamma \chi_{\mathsf{k}}(\boldsymbol{x})\varphi(t,\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \end{pmatrix} \quad \text{for all } \varphi \in L^2(V), \text{ a.e. in } [0,T]. \tag{2.18}$$

For the second term in (2.16) we introduce the adjoint state $p(u) \in Z'$ associated with the control u: It is given by the solution to

$$L_y(y(u),u,p(u)) = 0.$$

Let $v \in Y$. We have

$$\begin{aligned} \langle L_y(y(u),u,p(u)), v\rangle_{Y',Y} &= \langle J_y(y(u),u), v\rangle_{Y_1',Y_1} + \langle p(u), e_y(y(u),u)v\rangle_{Z',Z} \\ &= \langle J_y(y(u),u) + e_y(y(u),u)^* p(u), v\rangle_{Y_1',Y_1}. \end{aligned} \tag{2.19}$$

The second equality holds since $e_y(y(u),u)$ maps from $W(0,T)$ to Z, see Remark 2.14.

This shows that $L_y(y(u), u, p(u))$ belongs to $W(0,T)'$. We can deduce $\hat{J}'(u) \in U' \sim U$ via equation (2.16) and equation (2.17) with $y'(u) \in \mathcal{L}(U, W(0,T))$.

Let us determine the adjoint state $p(u) = (p(u)_1, p(u)_2)$. For clarity we write $p_1 := p(u)_1$ and $p_2 := p(u)_2$. The adjoint state can be viewed as a Lagrange multiplier associated with (SE). Hence, its existence with $p_1 \in W(0,T)$ follows from Remark 2.15(3). Equation (2.19) says that the adjoint state is given by the solution to the

adjoint equation

$$e_y(y(u), u)^* p(u) = -J_y(y(u), u).$$

Let $y_u := y(u)$. Written in variational form, the adjoint equation reads

$$\langle e_y^1(y_u, u)v, p_1\rangle_{L^2(V'),L^2(V)} + \langle e_y^2(y_u, u)v, p_2\rangle_H = -\langle J_y(y_u, u), v\rangle_{Y_1',Y_1} \quad \text{for all } v \in Y_1.$$

Expanding $e_y^1(y_u, u)$, $e_y^2(y_u, u)$ and inserting $J_y(y_u, u)$ gives

$$\langle c_p v_t + \mathcal{A}v + \mathcal{N}'(y_u)v, p_1\rangle_{L^2(V'),L^2(V)} + \langle p_2, v(0)\rangle_H = -\langle y_u(T) - y_d, v(T)\rangle_H.$$

The formula of integration by parts applied on the term $\langle c_p v_t, p_1\rangle_{L^2(V'),L^2(V)}$ yields

$$\begin{aligned}\langle c_p p_1(T), v(T)\rangle_H - \langle c_p p_1(0), v(0)\rangle_H \\ + \langle -c_p p_{1,t} + \mathcal{A}p_1 + \mathcal{N}'(y_u)p_1, v\rangle_{L^2(V'),L^2(V)} \\ + \langle p_2, v(0)\rangle_H = -\langle y_u(T) - y_d, v(T)\rangle_H.\end{aligned}$$

Notice that the equality $\mathcal{A} = \mathcal{A}^*$ was used, which follows from symmetry of the bilinear form a.

Recall the space $C_c^\infty((0,T); V)$ which consists of all functions in $C^\infty((0,T); V)$ with compact support in $(0,T)$. The space $C_c^\infty((0,T); V) \subset W(0,T)$ is dense in $L^2(V)$; see [21, p. 80]. Therefore, the above equation is equivalent to

$$\begin{aligned}&\langle -c_p p_{1,t} + \mathcal{A}p_1 + \mathcal{N}'(y_u)p_1, v\rangle_{L^2(V'),L^2(V)} = 0 \quad \text{for all } v \in L^2(0,T;V),\\ &c_p p_1(T) = -(y_u(T) - y_d) \ \in H, \quad p_2 = c_p p_1(0).\end{aligned}$$

This is the weak formulation of

$$(\mathrm{AE}_1) \quad \begin{cases} -c_p p_{1,t} - \Delta p_1 + N'(y_u(\cdot,\cdot))p_1 = 0 & \text{in } Q,\\ \dfrac{\partial p_1}{\partial \nu} + q p_1 = 0 & \text{on } \Sigma,\\ c_p p_1(T) = -(y_u(T) - y_d) & \text{in } \Omega. \end{cases}$$

We already know that $p_1 \in W(0,T)$ exists and Lemma 3.17 in [45] provides unique existence of a weak solution to (AE_1). From $y_u(T) - y_d \in C(\bar{\Omega})$, it even follows $p_1 \in C(\bar{Q})$. This implies $p_1 \in Y$; see [45, p. 279]. Hence, the unique adjoint state associated with u is given by $p(u) = (p_1, c_p p_1(0))$.

We insert the adjoint state $p(u)$ into equation (2.16). Thus, the derivative $\hat{J}'(u)$ is given by formula (2.17) and only the first component of $p(u)$ is needed. This is why we set $p_u := p(u)_1$. In the following we will refer to only this first component as the adjoint state

associated with u. Since we identified $\hat{J}'(u) \in U'$ with an element in U we refer to it as the gradient of the reduced cost functional $\hat{J}$ at $u \in L^\infty(0,T;\mathbb{R}^{\mathsf{k}})$.

We derived the following computation scheme:

Computation of the representation of $\hat{\mathbf{J}}'(\mathbf{u})$ in U:
Require: $u \in L^\infty(0,T;\mathbb{R}^{\mathsf{k}})$;

1. Solve (SE) to get $y_u = y(u)$;
2. Solve (AE_1) to get $p_u = p(u)_1$;
3. Insert u and p_u into

$$\begin{aligned}(\hat{J}'(u))(t) = \quad & D_\gamma u(t) \quad - \quad (\mathcal{B}^* p_u)(t) \\ = & \begin{pmatrix}\gamma_1 u_1(t)\\ \vdots \\ \gamma_{\mathsf{k}} u_{\mathsf{k}}(t)\end{pmatrix} - \begin{pmatrix}\int_\Gamma \chi_1(\boldsymbol{x}) p_u(t,\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}\\ \vdots \\ \int_\Gamma \chi_{\mathsf{k}}(\boldsymbol{x}) p_u(t,\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}\end{pmatrix};\end{aligned} \tag{2.20}$$

Let us turn to the second-order derivative. We differentiate equation (2.16) in the direction $w_2 \in L^\infty(0,T;\mathbb{R}^{\mathsf{k}})$, omit the mixed derivatives which vanish for the given problem and use point (4) from Remark 2.15:

$$\begin{aligned}\langle \hat{J}''(u)w_2, w_1\rangle_{U_1',U_1} = {} & \langle L_{uu}(y(u),u,p)w_2, w_1\rangle_{U_1',U_1} \\ & + \langle L_{yy}(y(u),u,p)y'(u)w_2, y'(u)w_1\rangle_{Y',Y} \\ & + \langle L_y(y(u),u,p), y''(u)(w_1,w_2)\rangle_{Y_1',Y_1}.\end{aligned}$$

By inserting the adjoint state $p = p(u)$, satisfying $L_y(y(u),u,p(u)) = 0$, we obtain

$$\begin{aligned}\langle \hat{J}''(u)w_2, w_1\rangle_{U_1',U_1} = {} & \langle L_{uu}(y(u),u,p(u))w_2, w_1\rangle_{U_1',U_1} \\ & + \langle L_{yy}(y(u),u,p(u))y'(u)w_2, y'(u)w_1\rangle_{Y',Y}.\end{aligned} \tag{2.21}$$

We verify that the controls w_1, w_2 in equation (2.21) can also belong to U:

The first term $L_{uu}(y(u),u,p(u))$ equals $J_{uu}(y(u),u) \in U' \sim U$ since the state equation is linear in u; compare equation (2.17). This gives

$$(L_{uu}(y(u),u,p(u))w)(t) = D_\gamma w(t) \quad \text{for } w \in U, \text{a.e. in } [0,T]. \tag{2.22}$$

The second term in (2.21) requires some effort. For any $v, \varphi \in Y$ we have

$$\begin{aligned}\langle L_{yy}(y(u),u,p(u))v, \varphi\rangle_{Y',Y} &= \langle J_{yy}(y(u),u)v, \varphi\rangle_{Y_1',Y_1} + \langle e^1_{yy}(y(u),u)(v,\varphi), p(u)_1\rangle_{L^2(V'),L^2(V)} \\ &= \langle v(T), \varphi(T)\rangle_H + \langle \mathcal{N}''(y(u))(v,\varphi), p(u)_1\rangle_{L^2(V'),L^2(V)}.\end{aligned}$$

Hence, v and φ can also belong to $W(0,T)$ and we can view $L_{yy}(y(u),u,p(u))$ as mapping from $W(0,T)$ to $W(0,T)'$. Consequently, the controls w_1, w_2 in (2.21) can belong to U and we have

$$\langle L_{yy}(y(u),u,p(u))y'(u)w_2, y'(u)w_1\rangle_{Y_1',Y_1} = \langle y'(u)^* L_{yy}(y(u),u,p(u))y'(u)w_2, w_1\rangle_U.$$

Let $w \in U$. The computation of $y'(u)^* L_{yy}(y(u), u, p(u)) y'(u) w$ needs to be done in several steps. We define

$$\begin{aligned} v &:= y'(u)w \overset{(2.14)}{=} -e_y(y(u), u)^{-1}\, e_u(y(u), u)w, \\ h &:= L_{yy}(y(u), u, p)v. \end{aligned}$$

Thus, we obtain

$$y'(u)^* L_{yy}(y(u), u, p(u)) y'(u) w = y'(u)^* h.$$

With $y_u := y(u)$ and $p_u := p(u)_1$ this means:

1. The sensitivity $v = y'(u)w$ is the solution to (LSE).
2. The application of h is given by

$$\langle h, \varphi \rangle_{Y_1', Y_1} = \langle v(T), \varphi(T) \rangle_H + \langle \mathcal{N}''(y_u)(v, \varphi), p_u \rangle_{L^2(V'), L^2(V)}.$$

3. The formula for $y'(u)^* h$ reads

$$y'(u)^* h = -e_u(y(u), u)^* e_y(y(u), u)^{-*} h.$$

   Hence, this requires one adjoint equation solve. We define $p := -e_y(y(u), u)^{-*} h$. This gives the adjoint equation with $J_y(y(u), u)$ replaced by h. Let $p_1 \in W(0, T)$ be the unique weak solution to

$$(\mathrm{AE}_2) \quad \begin{cases} -c_p p_{1,t} - \Delta p_1 + N'(y_u(\cdot,\cdot))p_1 = -N''(y_u(\cdot,\cdot))v p_u & \text{in } Q, \\ \dfrac{\partial p_1}{\partial \nu} + q p_1 = 0 & \text{on } \Sigma, \\ c_p p_1(T) = -v(T) & \text{in } \Omega. \end{cases}$$

   Thus, p is given by $p = (p_1, c_p p_1(0))$. Note that $w \in L^\infty(0, T; \mathbb{R}^{\mathsf{k}})$ first implies $v \in Y$ and hence $p_1 \in Y$. Consequently,

$$y'(u)^* h = e_u(y(u), u)^* h = -\mathcal{B}^* p_1.$$

We summarize the computation of the application of the Hessian $\hat{J}''(u)$, $u \in L^\infty(0, T; \mathbb{R}^{\mathsf{k}})$, on a vector $w \in U$:

Computation of Hessian-vector products $\hat{\mathbf{J}}''(\mathbf{u})\mathbf{w}$:
Require: $w \in U$, $y_u = y(u)$ and $p_u = p(u)_1$ (with $u \in L^\infty(0, T; \mathbb{R}^{\mathsf{k}})$);

1. Solve (LSE) to get v;
2. Solve (AE_2) to get p_1;
3. Insert w and p_1 into

$$\begin{aligned} (\hat{J}''(u)w)(t) &= D_\gamma w(t) - (\mathcal{B}^* p_1)(t) \\ &= \begin{pmatrix} \gamma_1 w_1(t) \\ \vdots \\ \gamma_{\mathsf{k}} w_{\mathsf{k}}(t) \end{pmatrix} - \begin{pmatrix} \int_\Gamma \chi_1(\boldsymbol{x}) p_1(t, \boldsymbol{x})\, \mathrm{d}\boldsymbol{x} \\ \vdots \\ \int_\Gamma \chi_{\mathsf{k}}(\boldsymbol{x}) p_1(t, \boldsymbol{x})\, \mathrm{d}\boldsymbol{x} \end{pmatrix}; \end{aligned} \tag{2.23}$$

Remark 2.16. Consider the linear case, i.e. let N be a linear function so that N' is constant and N'' equals zero. Then, via (SE) and (AE_1), $\hat{J}'(\cdot)$ is linear and, via (LSE) and (AE_2), $\hat{J}''(\cdot)$ is constant as expected.

2.5 Optimality Conditions

For a general consideration of optimality conditions for PDE constrained optimization problems we refer the reader to [21] and [45] where detailed explanations can be found.

In the following, we denote by $\bar{u} \in U_{\mathsf{ad}}$ a locally optimal control for problem (**P**) in the sense of $L^\infty(0,T;\mathbb{R}^{\mathsf{k}})$. We name a first-order necessary optimality condition that has to be obeyed by the control $\bar{u}$ and, via the associated state $y_{\bar{u}} := y(\bar{u})$, by the associated adjoint state $p_{\bar{u}} := p(\bar{u})_1$. It is derived with the aid of the reduced cost functional $\hat{J}$.

Theorem 2.17. *Suppose that* (**A2**)-(**A3**) *hold. Every locally optimal control $\bar{u} \in U_{\mathsf{ad}}$ for* (**P**) *satisfies, together with the associated adjoint state $p_{\bar{u}}$, the variational inequality*

$$\int_0^T \sum_{k=1}^{\mathsf{k}} \Big(\gamma_k \bar{u}_k(t) - \int_\Gamma \chi_k(\boldsymbol{x}) p_{\bar{u}}(t,\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}\Big)\big(u_k(t) - \bar{u}_k(t)\big)\,\mathrm{d}t \;\geq 0 \quad \textit{for all } u \in U_{\mathsf{ad}}. \tag{2.24}$$

Proof. Inequality (2.24) is the variational inequality

$$\hat{J}'(\bar{u})(u - \bar{u}) \geq 0 \quad \text{for all } u \in U_{\mathsf{ad}}, \tag{2.25}$$

which is, for instance, given in [45, Theorem 5.13]. Here, we inserted the representation (2.20) of $\hat{J}'(\bar{u})$. □

Remark 2.18. The first-order necessary optimality condition (2.24) can also be derived by using the Lagrange function L from Section 2.4. Then, the Karush-Kuhn-Tucker optimality conditions are given by

$$e(\bar{y},\bar{u}) = 0, \tag{2.26}$$
$$L_y(\bar{y},\bar{u},\bar{p}) = 0, \tag{2.27}$$
$$L_u(\bar{y},\bar{u},\bar{p})(u - \bar{u}) \geq 0 \quad \text{for all } u \in U_{\mathsf{ad}}, \tag{2.28}$$

with the unique Lagrange multiplier $\bar{p} \in Y \times H \subset Z'$; see [21, Corollary 1.3]. Equation (2.26) says that $\bar{y} = y(\bar{u})$ must be satisfied and equation (2.27) means that $\bar{p} = p(\bar{u})$ must hold. Hence, inequality (2.28) is equivalent to inequality (2.24) since $\hat{J}(u) = L(y(u),u,p)$ holds for all $u \in L^\infty(0,T;\mathbb{R}^{\mathsf{k}})$ and $p \in Z'$.

If $\hat{J}$ is convex, condition (2.24) is sufficient for global optimality; see [45, Lemma 2.21]. But as already mentioned in Remark 2.8, we might obtain a non-convex problem if N is nonlinear. In this case the first-order optimality condition (2.24) is just necessary. Thus, we turn to a second-order optimality condition that ensures local optimality.

Theorem 2.19. *Let the control $\bar{u}$ together with (the associated state $y_{\bar{u}}$ and) the adjoint state $p_{\bar{u}}$, satisfy the first-order optimality condition stated in Theorem 2.17. If there exists some $\mu > 0$ such that the coercivitiy condition*

$$\langle \hat{J}''(\bar{u})w, w\rangle_U \geq \mu \|w\|_U^2 \tag{2.29}$$

is satisfied for all $w \in L^\infty(0,T;\mathbb{R}^{\mathsf{k}})$*, then* $\bar{u}$ *is a locally optimal control for problem* (**P**).

The second-oder optimality condition (2.29) is named in Theorem 5.18 in [45], formulated for the Lagrange function. In this theorem, inequality (2.29) has to be satisfied for controls w belonging to a so-called τ-critical cone $C_\tau(\bar{u}) \subset L^\infty(0,T;\mathbb{R}^{\mathsf{k}})$. The reason why we chose the seemingly more restrictive condition and why we decided not to explicitly introduce the τ-critical cone is the following: In our numerical experiments we assume to have an (inactive) optimal control $\bar{u}$ for (**P**) which satisfies $u_a < \bar{u} < u_b$. Hence, the control $\bar{u}$ lies in the interior of the admissible set U_{ad} and the constraints are not active at $\bar{u}$. For such a solution it holds:

1. Inequality (2.25) turns into the well-known first-order optimality condition

$$\hat{J}'(\bar{u}) = 0. \tag{2.30}$$

   As usually done, we refer to points that satisfy equality (2.30) as stationary points of $\hat{J}$.

2. Any τ-critical cone corresponding to $\bar{u}$ is the entire space $L^\infty(0,T;\mathbb{R}^{\mathsf{k}})$.

Remark 2.20. (1) In our numerical test examples we do not specify the bounds u_a and u_b. These are assumed to be given such that we have an optimal control $\bar{u}$ which satisfies $u_a < \bar{u} < u_b$. By applying the algorithms under consideration we assume to find this optimal control although the methods can just be proved to converge to stationary points. There is no theoretical guarantee that the iterates (controls) are bounded. Hence, as a safeguard we stop the optimization, if they are unbounded. However, boundedness was given in all our numerical computations.
Note that the optimization will be carried out in the Hilbert space U while the iterates belong to $L^\infty(0,T;\mathbb{R}^{\mathsf{k}})$; see Remark 3.4(1). This is why the numerical methods are derived for a general Hilbert space setting.

(2) In Chapter 3 we consider a local inexact Newton method. An assumption in the convergence analysis even requires that inequality (2.29) is satisfied for all $w \in U$.

3 Line Search Newton-CG Method

The basic idea of the pursued approach for finding a solution to (**P**) is the following: By viewing the state y as a function of the control u and by introducing the reduced problem ($\hat{\mathbf{P}}$) we can apply the Newton method directly to the first-order optimality condition

$$\hat{J}'(\bar{u}) = 0. \tag{3.1}$$

In [20] this strategy has been compared to the approach of considering (y, u) as independent variables resulting in a sequential quadratic programming (SQP) method and has been applied to the Navier-Stokes equation.

The first section of this chapter is dedicated to the derivation of the so-called line search Newton-CG or truncated Newton method. It is abbreviated as LSNCG and gets formulated for an arbitrary real Hilbert space. We start by investigating a local inexact Newton method. The approximative Newton steps are computed iteratively with a conjugate gradient (CG) algorithm. Inexact Newton methods are well-known for their fast local convergence when the inexactness is suitably controlled. By adding a line search strategy we obtain a globally convergent algorithm.

For the numerical application of the LSNCG strategy a spatial and temporal discretization is required. The finite element (FE) Galerkin technique in combination with the implicit Euler method is presented in Section 3.2.

3.1 Derivation of the Algorithm

The derivation is mainly based on [25] and [49] which deal with the Euclidean space. To transfer the algorithm to an arbitrary real Hilbert space we take [20] and [21] into account.

The setting for this section is as follows: Let U be a real Hilbert space and $\hat{J} : U \to \mathbb{R}$. We consider the unconstrained optimization problem

$$\min_{u \in U} \hat{J}(u).$$

Assume that $\hat{J}$ is sufficiently smooth and that $\bar{u} \in U$ denotes a local minimum that satisfies the second-order sufficient optimality condition:

(**A4**) $\hat{J}$ is twice continuously Fréchet differentiable and bounded from below,
(**A5**) $\hat{J}'(\bar{u}) = 0$,
(**A6**) there exists some $\mu > 0$ such that $\langle \hat{J}''(\bar{u})w, w\rangle_U \geq \mu \|w\|_U^2$ holds for all $w \in U$.

An inexact Newton method allows to solve the Newton equation in iteration k,

$$\hat{J}''(u^{(k)})d^{(k)} = -\hat{J}'(u^{(k)}), \tag{3.2}$$

approximately.

Remark 3.1. Let $\hat{J}''(u^{(k)})$ be positive definite. Then the quadratic programming problem

$$\min_{d \in U} m^{(k)}(d) = \hat{J}^{(k)} + \langle \hat{J}'(u^{(k)}), d \rangle_U + \frac{1}{2} \langle \hat{J}''(u^{(k)})d, d \rangle_U \tag{3.3}$$

is equivalent to a Newton step. The function $m^{(k)}$ is strictly convex so that the Newton equation (3.2) is the necessary and sufficient first-order optimality condition for problem (3.3).

A basic local inexact Newton algorithm is defined in Algorithm 1; compare Algorithm 10.1 in [14] or Framework 11.2 in [49]. As done in [49], we refer to the sequence $\{\eta_k\}_{k \in \mathbb{N}}$

Algorithm 1 Local inexact Newton algorithm

Require: An initial point $u^{(0)} \in U$ in a neighbourhood of $\bar{u}$, $\eta \in [0, 1)$;

1: For $k = 0, 1, 2, \ldots$
choose $\eta_k \in [0, \eta]$ and solve

$$\left\| \hat{J}''(u^{(k)})d^{(k)} + \hat{J}'(u^{(k)}) \right\|_U \leq \eta_k \left\| \hat{J}'(u^{(k)}) \right\|_U; \tag{3.4}$$

2: Set $u^{(k+1)} = u^{(k)} + d^{(k)}$;

as the forcing sequence. The algorithm leaves the following two choices:

- How to choose the forcing sequence $\{\eta_k\}_{k \in \mathbb{N}}$?
- How to compute the approximative or inexact Newton steps/directions $d^{(k)}$, $k \in \mathbb{N}$?

In addition, we will investigate the question:

- How to obtain global convergence (to stationary points)?

Forcing sequence:
The method that is used to find the approximative Newton steps does not affect the rate of convergence of Algorithm 1. It is the forcing sequence which does so. Obviously, in case of $\eta_k = 0$ equation (3.4) turns into the exact Newton equation (3.2). The following theorem shows that the fast local convergence of the classical Newton method can also be achieved by its inexact version just depending on the choice of the forcing sequence. It is basically Theorem 10.2 in [14] which deals with the situation where $\hat{J}$ maps from an Euclidean space. The corresponding proof can be transferred to a general Hilbert space.

Theorem 3.2. *Under the assumptions* (**A4**)-(**A6**)*, there exists a neighbourhood* $\mathcal{U} \subset U$ *of* $\bar{u}$ *such that for every* $u^{(0)} \in \mathcal{U}$ *the following holds:*

- *If* $\eta \in [0, 1)$ *is sufficiently small, Algorithm 1 is well-defined and the convergence of* $\{u^{(k)}\}_{k \in \mathbb{N}}$ *to* $\bar{u}$ *is linear.*
- *If* $\eta_k \to 0$*, the convergence is superlinear.*
- *The convergence is quadratic if the Hessian* $\hat{J}''$ *is Lipschitz continuous in* $\mathcal{U}$ *and if* $\eta_k \leq C_\eta \left\| \hat{J}'(u^{(k)}) \right\|_U$ *for some* $C_\eta > 0$*.*

Remark 3.3. Theorem 3.2 gives concrete suggestions for the choice of the forcing sequence. In our numerical examples we will use

$$\eta_k := \min\left\{\left(\frac{\|\hat{J}'(u^{(k)})\|_U}{\|\hat{J}'(u^{(0)})\|_U}\right)^{\eta_a}, \eta_b \frac{\|\hat{J}'(u^{(k)})\|_U}{\|\hat{J}'(u^{(0)})\|_U}\right\}. \tag{3.5}$$

The exponent η_a determines the order of convergence of Algorithm 1. One chooses $\eta_a = 1, 1.5$ or 2 in order to get linear, superlinear or quadratic convergence. The value of $\eta_b < 1$ is important at the beginning. Notice that $\eta_0 = \eta_b$ holds; compare [21, p. 249].

Approximative Newton steps:
To compute the approximative or inexact Newton steps we use a CG algorithm which is iterative method. It belongs to the class of Krylov space methods and is known as an efficient algorithm for solving (large) linear systems of equations

$$Ax = b \ \in \mathbb{R}^m \tag{3.6}$$

with a symmetric positive definite matrix $A \in \mathbb{R}^{m\times m}$; compare [25]. First, we name two relevant properties of the CG method:

(1) It can be implemented in a so-called matrix-free manner since the matrix A needs not to be calculated or stored. The algorithm only requires a routine which performs the action of A on a vector. In our context A is the Hessian so that the algorithm turns out to be Hessian-free.

(2) Excluding rounding errors it can be viewed as a direct method: Given an m-dimensional system, the CG algorithm terminates in at most m iterations with the exact solution; see [25, Theorem 2.2.1]. Furthermore, the CG method is known to find a very good approximation to the exact solution in many fewer than m iterations depending on the distribution of the eigenvalues of A.

At this point, we refer the reader to [25] or [49] for more details about the CG method in $\mathbb{R}^m$. Here, we formulate the CG algorithm directly for the Hilbert space U. Let $A \in \mathcal{L}(U,U)$, $b \in U$ and consider the linear equation

$$Au = b \ \in U. \tag{3.7}$$

The principal ideas of the CG method can be summarized as follows: Equation (3.7) has the same unique solution as the quadratic minimization problem

$$\min_{u\in U} \phi(u) := \frac{1}{2}\langle Au, u\rangle_U - \langle b, u\rangle_U, \tag{3.8}$$

The solution is computed by minimizing ϕ successively by one-dimensional minimizations along so-called A-conjugate directions. In iteration j with given actual iterate u^j the next iterate is set to $u^{j+1} := u^j + \alpha_j p^j$ with the following specifications:

- The directions p^j are generated iteratively such that they are conjugated with respect to A, i.e.
$$\langle p^i, Ap^j\rangle_U = 0 \quad \text{for all } i \neq j.$$

Here, only the previous iterate p^{j-1} (instead of all previous elements $p^0, \dots, p^{j-1}$) is needed to compute p^j. This is interesting in terms of storage requirement. We have the j-th residual $r^j := b - Au^j$ and the direction $p^j := r^j + \beta_j p^{j-1}$. The scalar β_j is defined by $\beta_j := \|r^{j-1}\|_U^2 / \|r^{j-2}\|_U^2$. This guarantees A-conjugacy.

- α_j denotes the one-dimensional minimizer

$$\alpha_j := \arg\min_{\alpha>0} \phi(u^j + \alpha p^j).$$

It is given by the formula $\alpha_j = \|r^{j-1}\|_U^2 / \langle p^j, Ap^j \rangle_U$.

A typical termination criterion for the CG iteration is

$$\| \underbrace{b - Au^j}_{=r^j} \|_U \leq \eta_{CG} \|b\|_U \quad \text{for some } \eta_{CG} > 0. \tag{3.9}$$

This means that the iteration gets stopped as soon as the relative residual is small enough. For $A = \hat{J}''(u^{(k)})$ and $b = -\hat{J}'(u^{(k)})$ condition (3.9) just coincides with condition (3.4) in Algorithm 1. The CG method is defined in Algorithm 2. We modified Algorithm 2.4.1 from [25] analogously to [49, Chapter 7]. The number of iterations is limited to j_{max}. The initial iterate u is set to be zero and gets overwritten with the solution.

Algorithm 2 CG algorithm

Require: $A \in \mathcal{L}(U,U)$, $b \in U$, a maximal number of iterations j_{max}, constants $\eta_{CG} > 0$, $\varepsilon_{cv} > 0$;

Set $u = 0$, $r = b$, $p = r$, $\rho_0 = \|r\|_U^2$ and $j = 1$;
while $j < j_{max}$ **do**
  Evaluate $w = Ap$;
  if $\langle p, w \rangle_U / \|p\|_U^2 < \varepsilon_{cv}$ **then**
    if $j = 1$ **then**
      return $u = p$;
    else
      return u;
    end if
  end if
  Set $\alpha = \rho_{j-1} / \langle p, w \rangle_U$;
  Set $u = u + \alpha p$;
  Set $r = r - \alpha w$;
  Set $\rho_j = \|r\|_U^2$.
  if $\sqrt{\rho_j} \leq \eta_{CG} \|b\|_U$ **then**
    return u;
  end if
  Set $\beta = \rho_{j-1} / \rho_{j-2}$;
  Set $p = r + \beta p$;
  Set $j = j + 1$;
end while

We added a negative curvature test to the primary CG algorithm (line 4) to handle the case in which the Hessian $\hat{J}''(u^{(k)})$ as input value A is not positive definite. This can occur when the control iterate $u^{(k)}$ is not close enough to $\bar{u}$. In theory, we terminate the CG iteration as soon as a direction of negative curvature occurs, i.e. $\langle p^j, Ap^j\rangle_U \leq 0$. In practice, due to numerical rounding erros we test if $\langle p^j, Ap^j\rangle_U / \|p^j\|_U^2 \leq \varepsilon_{cv}$ holds for some $\varepsilon_{cv} > 0$. If this is the case, we stop the CG iteration. And by choosing the specific starting point $u = 0$ we can ensure that the CG output value is a descent direction for $\hat{J}$ at $u^{(k)}$ (if the input values are $A = \hat{J}''(u^{(k)})$ and $b = -\hat{J}'(u^{(k)})$). If negative curvature is encountered in the first CG iteration, the output value is the steepest descent direction $-\hat{J}'(u^{(k)})$. In the second iteration we would obtain the scaled negative gradient $-(1+\beta)\hat{J}'(u^{(k)})$. The realized modifications give an answer to the third above question: We can add a step length algorithm to Algorithm 1 in order to guarantee global convergence.

Before doing this, we have a brief look at the computational costs: Each CG iteration requires one application of A on a vector (line 3), two inner products (line 4/11 and 14) and three linear combinations (line 12, 13 and 19).

Global convergence:
It remains to add the mentioned line search strategy to Algorithm 1 (line 2). Assume that we have applied Algorithm 2 on $A = \hat{J}''(u^{(k)})$ and $b = -\hat{J}'(u^{(k)})$ with output $d^{(k)}$. Hence, $d^{(k)}$ is a descent direction for $\hat{J}$ at $u^{(k)}$. Thus, we can choose an Armijo backtracking strategy to provide the following sufficient decrease in $\hat{J}$ for some $\alpha \in (0,1)$:

$$\hat{J}\big(u^{(k)} + t_k d^{(k)}\big) \leq \hat{J}\big(u^{(k)}\big) + \alpha\, t_k\, \langle \hat{J}'\big(u^{(k)}\big), d^{(k)}\rangle_U. \tag{3.10}$$

The initial step length for the Armijo algorithm is chosen to be 1. Successively, as long as condition (3.10) is not satisfied it gets reduced by a factor $\beta \in (0,1)$. The above Armijo rule together with a general descent method in a Banach space setting is discussed in [21, Chapter 2]. Note that the Armijo line search technique just ensures that our method converges from an arbitrary initial point to a stationary point. But if the iterates are once close enough to a suitable minimum, step length 1 preserves the fast local convergence rate of the inexact Newton method.

Finally, the LSNCG algorithm is formulated in Algorithm 3. The number of iterations is limited to k_{max}. A stopping criterion based on an absolute tolerance bound τ_a, $0 < \tau_a \ll 1$, and on a desired relative reduction $\tau_r \in (0,1)$ in the norm of the initial gradient $\|\hat{J}'(u^{(0)})\|_U$ is added.

Algorithm 3 Line search Newton-CG algorithm

Require: An initial point $u^{(0)} \in U$, maximal numbers of iterations k_{max}, j_{max}, constants $\tau_r \in (0,1)$, $0 < \tau_a \ll 1$, $\varepsilon_{cv} > 0$;
1: Set $k = 0$ and $\tau_0 = \|\hat{J}'(u^{(0)})\|_U$;
2: **while** $\|\hat{J}'(u^{(k)})\|_U > \tau_r\tau_0 + \tau_a$ and $k < k_{max}$ **do**
3: Set $b = -\hat{J}'(u^{(k)})$ and $A = \hat{J}''(u^{(k)})$;
4: Choose $\eta_k \in [0, \eta]$;
5: Apply Algorithm 2 on A and b with $\eta_{CG} = \eta_k$, j_{max} and ε_{cv} to get $d^{(k)}$;
6: Set $u^{(k+1)} = u^{(k)} + t_k d^{(k)}$, where t_k satisfies the Armijo condition (3.10);
7: Set $k = k + 1$;
8: **end while**

Remark 3.4. (1) The reduced optimal control problem ($\hat{\mathbf{P}}$) does not exactly fit into the setting of this section. But the presented LSNCG procedure (Algorithm 3 together with Algorithm 2) can be applied as given and serves the purpose of finding stationary points. The argumentation is as follows: Let $u^{(k)} \in L^\infty(0,T;\mathbb{R}^{\mathsf{k}})$. We know that the gradient $\hat{J}'(u^{(k)})$ belongs to the Hilbert space $U = L^2(0,T;\mathbb{R}^{\mathsf{k}})$ and that the Hessian $\hat{J}''(u^{(k)})$ is an element of $\mathcal{L}(U,U)$. Thus, the following chain of implications provides what is needed:

$$\begin{aligned} u^{(k)} \in L^\infty(0,T;\mathbb{R}^{\mathsf{k}}) &\overset{(\mathrm{SE})}{\Rightarrow} y(u^{(k)}) \in Y \overset{(\mathrm{AE}_1)}{\Rightarrow} p(u^{(k)}) \in Y \\ &\overset{(2.20)}{\Rightarrow} \hat{J}'(u^{(k)}) \in L^\infty(0,T;\mathbb{R}^{\mathsf{k}}) \\ &\overset{\mathrm{CG}}{\Rightarrow} \text{ inexact Newton step } d^{(k)} \in L^\infty(0,T;\mathbb{R}^{\mathsf{k}}) \\ &\Rightarrow u^{(k+1)} = u^{(k)} + t_k d^{(k)} \in L^\infty(0,T;\mathbb{R}^{\mathsf{k}}). \end{aligned}$$

(2) We do not know an exact (locally) optimal control for the problems we will solve numerically. Nevertheless, we want to generate a reference solution. Already being in the context of line search methods, it just seems obvious to use the gradient algorithm. In fact, we only have to replace the lines 4 to 6 in Algorithm 3 by the formula $d^{(k)} = -\hat{J}'(u^{(k)})$ for the steepest descent direction. In this case, the Hessian application $A = \hat{J}''(u^{(k)})$ in line 3 is not needed.

3.2 FE Galerkin Discretization

We turn to the numerical realization of the LSNCG method. This includes a spatial and temporal discretization.

First, we define an equidistant time grid in $[0,T]$ by

$$0 = t_0 < t_1 < \ldots < t_{N_t} = T,$$

with step size $\Delta t = 1/N_t$.

Whenever we have to discretize the integral $\int_0^T \ldots \,\mathrm{d}t$ in this thesis we apply the composed trapezoidal rule. That way, we obtain the following discrete inner product in U:

$$\langle u, w\rangle_{U_h} = \sum_{k=1}^{\mathsf{k}} \Delta t \left(\frac{u_k^{(0)} w_k^{(0)}}{2} + \sum_{j=1}^{N_t-1} u_k^{(j)} w_k^{(j)} + \frac{u_k^{(N_t)} w_k^{(N_t)}}{2} \right) \quad \text{for all } u, w \in U, \tag{3.11}$$

where

$$\begin{aligned} u_k^{(0)} &:= \frac{2}{\Delta t} \int_0^{\Delta t/2} u_k(t)\,\mathrm{d}t, \quad u_k^{(N_t)} := \frac{2}{\Delta t} \int_{1-\Delta t/2}^{1} u_k(t)\,\mathrm{d}t, \\ u_k^{(j)} &:= \frac{1}{\Delta t} \int_{t_j-\Delta t/2}^{t_j+\Delta t/2} u_k(t)\,\mathrm{d}t \quad \text{for } j = 1,\ldots,N_t-1, \end{aligned} \tag{3.12}$$

for any $k \in \{1,\ldots,\mathsf{k}\}$. Thus, the discrete norm is given by

$$\|u\|_{U_h}^2 = \sum_{k=1}^{\mathsf{k}} \Delta t \left(\frac{(u_k^{(0)})^2}{2} + \sum_{j=1}^{N_t-1} (u_k^{(j)})^2 + \frac{(u_k^{(N_t)})^2}{2} \right) \quad \text{for all } u \in U.$$

In iteration k in the LSNCG procedure the current gradient $\hat{J}'(u^{(k)})$ has to be evaluated by solving (SE) and (AE_1). Moreover, in each associated CG iteration the action of the Hessian $\hat{J}''(u^{(k)})$ on a control has to be performed. This requires the solution of (LSE) and (AE_2). To solve the PDEs numerically we apply the (vertical) method of lines; compare [27]. Hence, we first discretize in space. We use a FE Galerkin method to transform the PDEs to ordinary differential equations (ODEs). We refer the reader to the numerous references such as [5, 12, 27, 40] for more details about the FE method. The remaining time integration is carried out by the implicit Euler method.

Let $N_x \in \mathbb{N}_{>0}$. We consider a regular triangulation $\mathcal{T}_h$ of Ω with nodes P_i, $i = 1, \dots, N_x$. By $\varphi_1, \dots, \varphi_{N_x} \in V$ we denote the corresponding linearly independent nodal basis functions. These define the N_x-dimensional subspace $V^h := \operatorname{span}\{\varphi_1, \dots, \varphi_{N_x}\} \subset V$. Let us already mention that we will use linear ansatz functions in our numerical experiments. We begin by discretizing the state equation.

State equation:
The semi-discretization of (SE) is based on the variational formulation

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\langle c_p y(t), \varphi\rangle_H + a(y(t), \varphi) + \langle \mathcal{N}(y(t)), \varphi\rangle_{V',V} &= \langle (F + \mathcal{B}u)(t), \varphi\rangle_{V',V}, \\ \langle y(0), \varphi\rangle_H &= \langle y_0, \varphi\rangle_H, \end{aligned} \tag{3.13}$$

for all $\varphi \in V$, a.e. in $(0, T]$. This formulation is equivalent to the variational equation (2.10) (or (2.3) respectively) with added initial condition; compare [21, Section 1.3]. Now, the FE Galerkin method consists in replacing V by V^h and in searching for a solution $y^h \in L^2(0, T; V^h)$ with $y_t^h \in L^2(0, T; (V^h)')$ to

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\langle c_p y^h(t), \varphi\rangle_H + a(y^h(t), \varphi) + \langle \mathcal{N}(y^h(t)), \varphi\rangle_{V',V} &= \langle (F + \mathcal{B}u)(t), \varphi\rangle_{V',V}, \\ \langle y^h(0), \varphi\rangle_H &= \langle y_0, \varphi\rangle_H, \end{aligned} \tag{3.14}$$

for all $\varphi \in V^h$, a.e. in $(0, T]$. We insert the Galerkin ansatz

$$y^h(t) := \sum_{i=1}^{N_x} \mathrm{y}_i(t)\varphi_i \quad \text{for } t \in [0, T],$$

in equation (3.14). Since it suffices to test equation (3.14) with the basis functions φ_j, $j = 1, \dots, N_x$, we obtain an ODE for the nodal coefficient vector $\mathrm{y}(t) := (\mathrm{y}_1(t), \dots, \mathrm{y}_{N_x}(t))^T \in \mathbb{R}^{N_x}$. This yields the following matrices and vectors for $1 \le i, j \le N_x$, $1 \le k \le \mathsf{k}$ and f.a.a. $t \in (0, T]$:

$$\begin{aligned} &\mathrm{M}_{ij} := \langle \varphi_i, \varphi_j\rangle_H, \quad (\mathrm{y}_0)_j := \langle y_0, \varphi_j\rangle_H, \\ &\mathrm{S}_{ij} := a(\varphi_i, \varphi_j) = \langle \nabla\varphi_i, \nabla\varphi_j\rangle_H + q\langle \varphi_i, \varphi_j\rangle_{L^2(\Gamma)} \\ &\mathrm{B}_{jk} := \langle \chi_k, \varphi_j\rangle_{L^2(\Gamma)} = \int_{\Gamma_k} \varphi_j(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}, \\ &(\mathrm{F}(t))_j := \langle F(t), \varphi_j\rangle_{V',V} = \int_\Omega f(t, \boldsymbol{x})\varphi_j(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}, \\ &\big(\mathrm{N}(\mathrm{y})\big)_j := \int_\Omega N\Big(\sum_{i=1}^{N_x} \mathrm{y}_i\varphi_i(\boldsymbol{x})\Big)\varphi_j(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \quad \text{for } \mathrm{y} \in \mathbb{R}^{N_x}. \end{aligned}$$

The symmetric positive definite matrix M is called mass matrix and the symmetric positive definite matrix S stiffness matrix. Thus, the semi-discretization of (SE) reads

$$\begin{aligned} c_p \mathrm{M}\dot{\mathrm{y}}(t) + \mathrm{S}\mathrm{y}(t) + \mathrm{N}(\mathrm{y}(t)) &= \mathrm{F}(t) + \mathrm{B}u(t) \quad \text{a.e. in } (0,T], \\ \mathrm{M}\mathrm{y}(0) &= \mathrm{y}_0. \end{aligned} \tag{3.15}$$

The vector $\mathrm{y}(t)$ contains the values of the FE state $y^h(t)$ at all FE nodes at time t. We refer to system (3.15) as the truth approximation or fine model for (SE).

Remark 3.5. (1) The functions $u \in L^\infty(0,T;\mathbb{R}^{\mathsf{k}})$ and $\mathrm{F} \in L^2(0,T;\mathbb{R}^{N_x})$ are not necessarily continuous. Therefore, the Picard-Lindelöf theorem can not be directly applied to conclude that system (3.15) has a local (classical) unique solution. We assume that a unique solution $\mathrm{y} \in H^1(0,T;\mathbb{R}^{N_x})$ is given so that $\mathrm{y} \in C([0,T];\mathbb{R}^{N_x})$ holds. In case of $N(y) = y^3$ the assumption can be proven for some maximal time. In [12, Lemma 5.27] the general proceeding for an appropriate proof is shown by means of a linear equation and can get adapted. The relevant estimate can be taken over since the resulting nonlinear term is non-negative.

(2) It is known that the method of lines reduces parabolic PDEs to stiff systems of ODEs; see [27]. Therefore, we choose the implicit or backward Euler method to solve the semi-discretizations. It is not only A- but also L-stable. For a definition and a proof we refer to [41]. Additionally, the implicit Euler method is consistent of order one provided that the 'right-hand side in the ODE' is continuously differentiable. Regarding common stability, the implicit Euler method demands no restriction on $\Delta t / h_{max}^2$ in case of a linear parabolic PDE with h_{max} denoting the maximal edge length in the FE triangulation.

We apply the implicit Euler method to system (3.15). Let $\mathrm{y}^{(j)}$ be the approximation to $\mathrm{y}(t_j)$, $j = 0, \dots, N_t$. Hence, the full-discretization of (SE) is given by

$$\Big(\frac{c_p}{\Delta t}\mathrm{M} + \mathrm{S}\Big)\mathrm{y}^{(j+1)} + \mathrm{N}(\mathrm{y}^{(j+1)}) = \frac{c_p}{\Delta t}\mathrm{M}\mathrm{y}^{(j)} + \mathrm{F}^{(j+1)} + \mathrm{B}u^{(j+1)},\ j = 0, \dots, N_t - 1, \tag{3.16}$$

$$\mathrm{M}\mathrm{y}^{(0)} = \mathrm{y}_0, \tag{3.17}$$

with

$$\mathrm{F}^{(0)} := \frac{2}{\Delta t}\int_0^{\Delta t/2} \mathrm{F}(t)\,\mathrm{d}t, \quad \mathrm{F}^{(N_t)} := \frac{2}{\Delta t}\int_{1-\Delta t/2}^{1} \mathrm{F}(t)\,\mathrm{d}t,$$

$$\mathrm{F}^{(j)} := \frac{1}{\Delta t}\int_{t_j-\Delta t/2}^{t_j+\Delta t/2} \mathrm{F}(t)\,\mathrm{d}t,\ j = 1, \dots, N_t - 1.$$

The vectors $u^{(j)} \in \mathbb{R}^{\mathsf{k}}$, $j = 0, \dots, N_t$, are defined by $u^{(j)} := \big(u_1^{(j)}, \dots, u_{\mathsf{k}}^{(j)}\big)^T$ with $u_k^{(j)}$, $k = 1, \dots, \mathsf{k}$, given in (3.12).

Remark 3.6. (1) Equation (3.16) is divided by the time step size Δt since this leads to a better conditioned problem.

(2) In each time step the nonlinear equation (3.16) is solved with the Newton method in $\mathbb{R}^{N_x}$. The iteration is stopped as soon as a required accuracy in the discrete H-norm is given; see Section 4.2.1. As done in [34] we will use a semi-implicit time step as initial guess for the Newton method. This means that the nonlinear term in equation

(3.16) is once evaluated explicitly to obtain the first iterate by solving the resulting linear system of equations.

Nonlinearity:
Let us consider the evaluation of the nonlinearity $\mathrm{N}(\mathrm{y})$ with

$$(\mathrm{N}(\mathrm{y}))_j = \int_\Omega N\Big(\sum_{i=1}^{N_x} \mathrm{y}_i \varphi_i(\boldsymbol{x})\Big)\varphi_j(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}, \quad \mathrm{y} \in \mathbb{R}^{N_x}, j = 1, \dots, N_x, \tag{3.18}$$

and of its Jacobian $\mathrm{N}'(\mathrm{y})$ with

$$\frac{\partial(\mathrm{N}(\mathrm{y}))_j}{\partial \mathrm{y}_k} = \int_\Omega N'\Big(\sum_{i=1}^{N_x} \mathrm{y}_i \varphi_i(\boldsymbol{x})\Big)\varphi_k(\boldsymbol{x})\varphi_j(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}, \quad \mathrm{y} \in \mathbb{R}^{N_x}, 1 \le j, k \le N_x. \tag{3.19}$$

The idea behind the used approximation is to apply the nonlinearity just on the FE coefficients. For the following we understand N as a function that maps from $\mathbb{R}$ to $\mathbb{R}$ or from $\mathbb{R}^{N_x}$ to $\mathbb{R}^{N_x}$ evaluating componentwise at its input vector. We introduce the approximation

$$(\mathrm{N}(\mathrm{y}))_j \approx \int_\Omega \Big(\sum_{i=1}^{N_x} N(\mathrm{y}_i)\varphi_i(\boldsymbol{x})\Big)\varphi_j(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = \sum_{i=1}^{N_x} N(\mathrm{y}_i) \int_\Omega \varphi_i(\boldsymbol{x})\varphi_j(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = (\mathrm{M}N(\mathrm{y}))_j.$$

Denote by $I : C(\bar{\Omega}) \to V^h$ the FE interpolation operator and recall the FE nodes P_k, $k = 1, \dots, N_x$. For any $g \in C(\bar{\Omega})$ we have $I(g) := \sum_{k=1}^{N_x} g(P_k)\varphi_k$. The above approximation can also be derived by replacing the term $g(\boldsymbol{x}) := N\big(\sum_{i=1}^{N_x} \mathrm{y}_i \varphi_i(\boldsymbol{x})\big)$ in (3.18) by $I(g(\boldsymbol{x}))$.

The corresponding Jacobian approximation is given by

$$\mathrm{N}'(\mathrm{y}) \approx \mathrm{M} \cdot \mathrm{diag}\left(N'(\mathrm{y}_1), \dots, N'(\mathrm{y}_{N_x})\right) \in \mathbb{R}^{N_x \times N_x}.$$

The Jacobian $N'(\mathrm{y})$ is not only needed for the Newton method to solve (SE) but it additionally occurs in the discrete schemes of (AE_1), (AE_2) and (LSE). In brief, we also name the semi- and full-discretizations of these PDEs without writing down their variational formulations analogously to system (3.13).

Linearized state equation:
For the linearized state we use the FE Galerkin ansatz

$$v^h(t) := \sum_{i=1}^{N_x} \mathrm{v}_i(t)\varphi_i, \ \mathrm{v}(t) := (\mathrm{v}_1(t), \dots, \mathrm{v}_{N_x}(t))^T \in \mathbb{R}^{N_x} \ \text{ for } t \in [0, T].$$

Denote by $y^h \approx y_u$ the FE state associated with u, with coefficient vector $\mathrm{y}(t)$. This gives

$$\begin{aligned} c_p \mathrm{M}\dot{\mathrm{v}}(t) + \left(\mathrm{S} + \mathrm{N}'(\mathrm{y}(t))\right)\mathrm{v}(t) &= \mathrm{B}s(t) \quad \text{a.e. in } (0, T], \\ \mathrm{v}(0) &= 0. \end{aligned} \tag{3.20}$$

Applying the implicit Euler method with $\mathrm{v}^{(j)} \approx \mathrm{v}(t_j)$, $j = 0, \ldots, N_t$, yields

$$\begin{aligned} \Big(\frac{c_p}{\Delta t}\mathrm{M} + \mathrm{S} + \mathrm{N}'(\mathrm{y}^{(j+1)})\Big)\mathrm{v}^{(j+1)} &= \frac{c_p}{\Delta t}\mathrm{M}\mathrm{v}^{(j)} + \mathrm{B}s^{(j+1)}, \quad j = 0, ..., N_t - 1, \\ \mathrm{v}^{(0)} &= 0. \end{aligned} \tag{3.21}$$

Adjoint equations:
We treat the adjoint equations (AE_1) and (AE_2) simultaneously by considering the following PDE:

$$\begin{aligned} -c_p p_t - \Delta p + N'(y_u(\cdot,\cdot))p &= r_Q \quad \text{in } Q, \\ \frac{\partial p}{\partial \nu} + qp &= 0 \quad \text{in } \Sigma, \\ c_p p(T) &= r_T \quad \text{in } \Omega, \end{aligned} \tag{3.22}$$

where $r_Q \in L^2(Q)$ and $r_T \in H$.

The FE Galerkin ansatz

$$p^h(t) := \sum_{i=1}^{N_x} \mathrm{p}_i(t)\varphi_i, \ \mathrm{p}(t) := (\mathrm{p}_1(t), \ldots, \mathrm{p}_{N_x}(t))^T \in \mathbb{R}^{N_x}, \ \text{ for } t \in [0,T],$$

yields the backward ODE

$$\begin{aligned} -c_p\mathrm{M}\dot{\mathrm{p}}(t) + (\mathrm{S} + \mathrm{N}'(\mathrm{y}(t)))\mathrm{p}(t) &= \mathrm{r}_\mathrm{Q}(t) \quad \text{a.e. in } [0,T), \\ c_p\mathrm{M}\mathrm{p}(T) &= \mathrm{r}_\mathrm{T}, \end{aligned} \tag{3.23}$$

with

$$(\mathrm{r}_\mathrm{Q}(t))_j := \int_\Omega r_Q(t,\boldsymbol{x})\varphi_j(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}, \quad (\mathrm{r}_\mathrm{T})_j := \langle r_T, \varphi_j\rangle_H \quad j = 1, \ldots, N_x.$$

By applying the implicit Euler method with $\mathrm{p}^{(j)} \approx \mathrm{p}(t_j)$, $j = 0, \ldots, N_t$, we obtain the full-discretization

$$\begin{aligned} \Big(\frac{c_p}{\Delta t}\mathrm{M} + \mathrm{S} + \mathrm{N}'(\mathrm{y}^{(j-1)})\Big)\mathrm{p}^{(j-1)} &= \frac{c_p}{\Delta t}\mathrm{M}\mathrm{p}^{(j)} + \mathrm{r}_\mathrm{Q}^{(j-1)} \quad j = N_t, \ldots, 1, \\ c_p\mathrm{M}\mathrm{p}^{(N_t)} &= \mathrm{r}_\mathrm{T}, \end{aligned} \tag{3.24}$$

where $\mathrm{r}_\mathrm{Q}^{(j)}$, $j = 0, \ldots, N_t$, is defined analogously to $\mathrm{F}^{(j)}$.

For the adjoint equations (AE_1) and (AE_2) we have

	(AE_1)	(AE_2)
r_Q	0	$-N''(y_u(\cdot,\cdot))\, v\, p_u$
r_T	$-(y_u(T) - y_d)$	$-v(T)$

To derive the corresponding terms for the full-discretization (3.24) we define

$$\mathrm{y}_d \in \mathbb{R}^{N_x} \ \text{ by } \ (\mathrm{y}_d)_j := \langle y_d, \varphi_j\rangle_H, \ j = 1, \ldots, N_x,$$

and insert any FE approximations. To evaluate $\mathrm{r}_\mathrm{Q}^{(j)}$, $j = 0, \ldots, N_t$, in case of (AE_2) we use the same approximation as for the nonlinearity. We denote by p_u the FE coefficient

vector belonging to the adjoint state $p_u = p(u)_1$. This gives

	(AE_1)	(AE_2)
$\mathrm{r}_{\mathrm{Q}}^{(j)}$	0	$-\mathrm{M} \cdot \left(N''(\mathrm{y}_i^{(j)}) \mathrm{v}_i^{(j)} \mathrm{p}_{u,i}^{(j)} \right)_{i=1}^{N_x}$
r_{T}	$-(\mathrm{M}\mathrm{y}^{(N_t)} - \mathrm{y}_d)$	$-\mathrm{M}\mathrm{v}^{(N_t)}$

In practice a rather large number N_x of FE basis functions is needed to provide a satisfactory accuracy of the FE Galerkin method. When applying the LSNCG procedure for the numerical solution of $(\hat{\mathbf{P}})$ we have to solve linear systems of equations of this high order N_x again and again as the above full-discretizations show. Notice that the Newton method to solve the nonlinear equations (3.16) might even need several iterations. This requires a considerable numerical effort and motivates the next chapter where we investigate a reduced-order modeling approach.

4 Reduced-Order Modeling Using POD

In this chapter we turn to the method of proper orthogonal decomposition (POD) to obtain a reduced-order model (ROM) for each given PDE. The idea is to replace the local FE basis functions from Section 3.2 by (only a few) global and problem-dependent POD basis functions to discretize the spatial variable. To achieve that the ROMs are completely independent of the large dimension N_x we consider two variants of the empirical interpolation method (EIM).

Let us mention that POD is also called Principal Component Analysis (PCA) or Karhunen-Loève Decomposition. In brief, the relevant theory about POD can be found in [17, 22, 23, 34, 43]. For more details we refer the reader to the lecture notes [48] which give a profound introduction. Let us start with a citation from these notes: 'One of the central issues of POD is the reduction of data expressing their *essential information* by means of a few basis vectors' [48, p. 6].

Firstly, in Section 4.1 this data is given by the span of the whole solution trajectory $\{y(t) \,|\, t \in [0,T]\}$ of a parabolic PDE. Exemplarily, we consider the state equation. Then, it is about finding a POD basis $\{\psi_i\}_{i=1}^{\ell}$ of rank ℓ that describes the above span in a certain optimal manner. Secondly, we discuss the numerically feasible discrete version of the POD method in Section 4.2. The POD Galerkin schemes of the involved PDEs are derived in Section 4.3. Finally, the application of empirical interpolation is content of Section 4.4. We keep close to the references [34] and [48].

In the following X denotes either the space H or the space V.

4.1 Continuous Version of the POD Method

Let $y = y(u)$ be a weak solution to (SE) for some $u \in L^\infty(0,T;\mathbb{R}^{\mathsf{k}})$. Assume that y belongs to $C([0,T];V) \hookrightarrow C([0,T];X)$; see Remark 2.5(3). The data that is supposed to get described is given by

$$\mathcal{V} := \operatorname{span}\{y(t) \,|\, t \in [0,T]\} \subseteq V \subset X, \tag{4.1}$$

and $d := \dim \mathcal{V} \leq \infty$.

Definition 4.1. For $\ell \in \{1,\dots,d\}$ a POD basis $\{\psi_i\}_{i=1}^{\ell}$ of rank ℓ is the solution to the optimization problem

$$\begin{cases} \min\limits_{\psi_1,\dots,\psi_\ell \in X} \displaystyle\int_0^T \Big\| y(t) - \sum_{i=1}^{\ell} \langle y(t), \psi_i \rangle_X \, \psi_i \Big\|_X^2 \,\mathrm{d}t \\ \text{s.t.} \quad \langle \psi_i, \psi_j \rangle_X = \delta_{ij} \text{ for } 1 \leq i,j \leq \ell, \end{cases} \tag{4.2}$$

where δ_{ij} stands for the Kronecker delta, i.e. $\delta_{ij} = 0$ for $i \neq j$ and $\delta_{ii} = 1$.

To solve problem (4.2) we utilize the linear integral operator $\mathcal{R} : X \to \mathcal{V}$, defined by

$$\mathcal{R}\psi := \int_0^T \langle \psi, y(t) \rangle_X \, y(t) \, \mathrm{d}t \quad \text{for } \psi \in X.$$

The operator $\mathcal{R}$ is self-adjoint, compact and non-negative; see [48, Lemma 2.1.3]. From the Riesz-Schauder and Hilbert-Schmidt theorems, see e.g., [48, Theorems A.17, A.18], it follows: There exists a complete orthonormal basis $\{\psi_i\}_{i=1}^d$ of $\mathcal{V}$ with corresponding real and non-negative eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d \geq 0$, $\lim_{i\to\infty} \lambda_i = 0$, such that

$$\mathcal{R}\psi_i = \lambda_i \psi_i \quad \text{and} \quad \langle \psi_i, \psi_j \rangle_X = \delta_{ij} \quad \text{for } 1 \leq i, j \leq \ell. \tag{4.3}$$

A proof of the following theorem is given in [48, Theorem 2.1.5].

Theorem 4.2. *The POD basis of rank $\ell \leq d$ solving problem* (4.2) *is given by eigenfunctions corresponding to the ℓ largest eigenvalues solving the eigenvalue problem* (4.3)*. Moreover, the POD approximation error equals*

$$\int_0^T \Big\| y(t) - \sum_{i=1}^{\ell} \langle y(t), \psi_i \rangle_X \, \psi_i \Big\|_X^2 \, \mathrm{d}t = \sum_{i=\ell+1}^{\infty} \lambda_i. \tag{4.4}$$

Remark 4.3. Having a look at the bounded linear operator $\mathcal{Y} : L^2(0,T) \to X$,

$$\mathcal{Y}\phi := \int_0^T \phi(t) y(t) \, \mathrm{d}t \quad \text{for } \phi \in L^2(0,T),$$

and its Hilbert space adjoint $\mathcal{Y}^* : X \to L^2(0,T)$ we can see that $\mathcal{R} = \mathcal{Y}\mathcal{Y}^*$ holds. Defining $\mathcal{K} := \mathcal{Y}^*\mathcal{Y} : L^2(0,T) \to L^2(0,T)$,

$$\mathcal{K}\phi = \int_0^T \langle y(s), y(\cdot) \rangle_X \, \phi(s) \, \mathrm{d}s \quad \text{for } \phi \in L^2(0,T),$$

provides an alternative, the so-called method of snapshots, to compute the POD basis: The operator $\mathcal{K}$ has the same properties as $\mathcal{R}$ and by the arguments of singular value decomposition (SVD), it follows that $\mathcal{K}$ and $\mathcal{R}$ have the same eigenvalues $\{\lambda_i\}_{i=1}^d$. Therefore, we can first solve

$$\mathcal{K}\phi_i = \lambda_i \phi_i, \quad \langle \phi_i, \phi_j \rangle_{L^2(0,T)} = \delta_{ij}, \quad 1 \leq i, j \leq \ell,$$

for the ℓ largest eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_\ell \geq 0$. Then, if $\lambda_i \neq 0$, we set (by SVD)

$$\psi_i = \frac{1}{\sqrt{\lambda_i}} \mathcal{Y}\phi_i = \frac{1}{\sqrt{\lambda_i}} \int_0^T \phi_i(t) y(t) \, \mathrm{d}t \quad \text{for } i = 1, \ldots, \ell.$$

Remark 4.4. (1) For a given rank ℓ the POD basis is chosen such that the error between the state y and the ℓ-th partial sum $\sum_{i=1}^{\ell} \langle y(\cdot), \psi_i \rangle_X \, \psi_i$ is minimal in the $L^2(0,T;X)$-norm.

(2) The POD basis depends on the chosen control u.

(3) Equation (4.4) says that the approximation quality of the POD method depends on the decay rate of the eigenvalues. It is decisive that they do not decay too slow.

4.2 Discrete Version of the POD Method

In numerics the whole trajectory $\{y(t) \,|\, t \in [0,T]\}$ is not available. Hence, a discrete version of the POD method is needed. In a first step we treat the situation where the FE approximations $V^h \ni y^h(t) \approx y(t) \in V$ are given. Secondly, we consider the fully discrete case.

4.2.1 Spatial Discretization

Assume that the FE approximation

$$y^h(t) = \sum_{i=1}^{N_x} \mathrm{y}_i(t)\varphi_i,\ \mathrm{y}(t) = (\mathrm{y}_1(t), \dots, \mathrm{y}_{N_x}(t))^T,\ t \in [0,T],$$

for the state y from Section 3.2 is given. The data set which is to be described is defined by

$$\mathcal{V} := \operatorname{span}\{\mathrm{y}(t) \,|\, t \in [0,T]\} \subset \mathbb{R}^{N_x},$$

with $d := \dim \mathcal{V} \leq N_x$.

We replace the inner product $\langle \cdot, \cdot \rangle_X$ in the N_x-dimensional FE subspace V^h by a weighted inner poduct in $\mathbb{R}^{N_x}$ for the FE coefficient vectors: Suppose that

$$u^h(\boldsymbol{x}) = \sum_{i=1}^{N_x} \mathrm{u}_i\varphi_i(\boldsymbol{x}) \quad \text{and} \quad v^h(\boldsymbol{x}) = \sum_{i=1}^{N_x} \mathrm{v}_i\varphi_i(\boldsymbol{x})$$

are two arbitrary elements in the FE space V^h with corresponding coefficient vectors $\mathrm{u} := (\mathrm{u}_1, \dots, \mathrm{u}_{N_x})^T$ and $\mathrm{v} := (\mathrm{v}_1, \dots, \mathrm{v}_{N_x})^T$. It follows

$$\langle u^h, v^h \rangle_X = \mathrm{u}^T \mathrm{W} \mathrm{v} =: \langle \mathrm{u}, \mathrm{v} \rangle_{\mathrm{W}} \ \text{ with } \begin{cases} \mathrm{W} = \mathrm{M} & \text{in case of } X = H, \\ \mathrm{W} = \mathrm{M} + \mathrm{A} & \text{in case of } X = V, \end{cases} \tag{4.5}$$

where A denotes the symmetric positive definite stiffness matrix $\mathrm{A} := \big(\langle \nabla\varphi_i, \nabla\varphi_j \rangle_H\big)_{i,j=1}^{N_x}$ and M the symmetric positive definite mass matrix from Section 3.2. The associated induced norm is $\|\mathrm{u}\|_W := (\langle \mathrm{u}, \mathrm{u} \rangle_{\mathrm{W}})^{1/2}$.

Now, the POD method is formulated for the FE coefficient vectors.

Definition 4.5. For $\ell \in \{1, \dots, d\}$ a POD basis $\{\mathrm{u}_i\}_{i=1}^{\ell}$ of rank ℓ is the solution to the minimization problem

$$\begin{cases} \min\limits_{\mathrm{u}_1, \dots, \mathrm{u}_\ell \in \mathbb{R}^{N_x}} \displaystyle\int_0^T \Big\| \mathrm{y}(t) - \sum_{i=1}^{\ell} \langle \mathrm{y}(t), \mathrm{u}_i \rangle_{\mathrm{W}}\, \mathrm{u}_i \Big\|_{\mathrm{W}}^2 \\ \text{s.t.} \quad \langle \mathrm{u}_i, \mathrm{u}_j \rangle_{\mathrm{W}} = \delta_{ij} \text{ for } 1 \leq i, j \leq \ell. \end{cases} \tag{4.6}$$

Let the operator $\mathcal{R}^h : \mathbb{R}^{N_x} \to \mathbb{R}^{N_x}$ be defined by

$$\mathcal{R}^h \mathrm{u} := \int_0^T \langle \mathrm{y}(t), \mathrm{u} \rangle_{\mathrm{W}}\, \mathrm{y}(t) \quad \text{for } \mathrm{u} \in \mathbb{R}^{N_x}.$$

It is a linear, bounded, non-negative and self-adjoint operator; see [48, Lemma 1.42].

Thus, by the following theorem, which is Theorem 1.4.3 in [48], we obtain an analogon to Theorem 4.2.

Theorem 4.6. *The POD basis of rank $\ell \leq d$ solving problem* (4.6) *is given by the eigenvectors corresponding to the ℓ largest eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_\ell > 0$ which solve the symmetric $N_x \times N_x$ eigenvalue problem*

$$\mathcal{R}^h \mathrm{u}_i = \lambda_i \mathrm{u}_i, \quad \langle \mathrm{u}_i, \mathrm{u}_j \rangle_{\mathrm{W}} = \delta_{ij} \quad \textit{for } 1 \leq i, j \leq \ell.$$

The POD approximation error is quantified by

$$\int_0^T \Big\| \mathrm{y}(t) - \sum_{i=1}^{\ell} \langle \mathrm{y}(t), \mathrm{u}_i \rangle_{\mathrm{W}} \, \mathrm{u}_i \Big\|_{\mathrm{W}}^2 \, \mathrm{d}t = \sum_{i=\ell+1}^{d} \lambda_i.$$

Remark 4.7. Again, we could consider the method of snapshots by introducing operators according to $\mathcal{Y}$ and $\mathcal{K}$. We refer the reader to [48] for more details.

4.2.2 Temporal Discretization

The above approach is still not practible in numerical computations. But for the given equidistant time grid $\{t_j\}_{j=0}^{N_t}$ in $[0, T]$ the snapshots

$$\mathbb{R}^{N_x} \ni \mathrm{y}^{(j)} \approx \mathrm{y}(t_j), \; j = 0, \ldots, N_t,$$

are available; compare Section 3.2.
Here, let $\mathcal{V} := \operatorname{span}\{\mathrm{y}^{(j)} \,|\, j = 0, \ldots, N_t\} \subset \mathbb{R}^{N_x}$ with $d := \dim \mathcal{V} \leq \min\{N_x, N_t + 1\}$.

Definition 4.8. For $\ell \in \{1, \ldots, d\}$ a POD basis $\{\mathrm{u}_i\}_{i=1}^{\ell}$ of rank ℓ is the solution to the optimization problem

$$\begin{cases} \min\limits_{\mathrm{u}_1, \ldots, \mathrm{u}_\ell \in \mathbb{R}^{N_x}} \displaystyle\sum_{j=0}^{N_t} \alpha_j \Big\| \mathrm{y}^{(j)} - \sum_{i=1}^{\ell} \langle \mathrm{y}^{(j)}, \mathrm{u}_i \rangle_{\mathrm{W}} \, \mathrm{u}_i \Big\|_{\mathrm{W}}^2 \\ \text{s.t.} \quad \langle \mathrm{u}_i, \mathrm{u}_j \rangle_{\mathrm{W}} = \delta_{ij} \quad \text{for } 1 \leq i, j \leq \ell, \end{cases} \tag{4.7}$$

with trapezoidal weights $\alpha_0 = \alpha_{N_t} = \Delta t/2$, $\alpha_j = \Delta t$, $j = 1, \ldots, N_t - 1$.

Note that (4.7) is the (composed) trapezoidal discretization of (4.6). Applying the trapezoidal rule on $\mathcal{R}^h$ yields the linear, bounded, non-negative and self-adjoint operator $\mathcal{R}^{h,n} : \mathbb{R}^{N_x} \to \mathbb{R}^{N_x}$,

$$\mathcal{R}^{h,n} \mathrm{u} = \sum_{j=0}^{N_t} \alpha_j \, \langle \mathrm{y}^{(j)}, \mathrm{u} \rangle_{\mathrm{W}} \, \mathrm{y}^{(j)} \quad \text{for } \mathrm{u} \in \mathbb{R}^{N_x}.$$

It can be utilized to solve problem (4.7). The relationship between the eigenvalues of $\mathcal{R}^{h,n}$ and $\mathcal{R}^h$ is investigated in [30]. For the following theorem see [48, Section 1.4].

Theorem 4.9. *The POD basis of rank $\ell \leq d$ solving problem* (4.7) *is given by the eigenvectors corresponding to the ℓ largest eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_\ell > 0$ solving the symmetric eigenvalue problem*

$$\mathcal{R}^{h,n} \mathrm{u}_i = \lambda_i \mathrm{u}_i, \quad \langle \mathrm{u}_i, \mathrm{u}_j \rangle_{\mathrm{W}} = \delta_{ij} \quad \textit{for } 1 \leq i, j \leq \ell. \tag{4.8}$$

The POD approximation error equals

$$\sum_{j=0}^{N_t} \alpha_j \left\| \mathrm{y}^{(j)} - \sum_{i=1}^{\ell} \langle \mathrm{y}^{(j)}, \mathrm{u}_i \rangle_{\mathrm{W}} \, \mathrm{u}_i \right\|_{\mathrm{W}}^2 = \sum_{i=\ell+1}^{d} \lambda_i.$$

We discuss the **POD basis computation**. Consider the following matrices for a fixed POD rank $\ell \leq d$:

- The matrix W is either the matrix M or the sum M + A depending on the choice of the space X.
- The POD basis matrix $\Psi := [\mathrm{u}_1, \ldots, \mathrm{u}_\ell] \in \mathbb{R}^{N_x \times \ell}$.
- The diagonal matrix $\mathrm{D} := \mathrm{diag}(\alpha_0, \ldots, \alpha_{N_t}) \in \mathbb{R}^{(N_t+1)\times(N_t+1)}$ containing the trapezoidal weights.
- The snapshot matrix $\mathrm{Z} := [\mathrm{y}^{(0)}, \ldots, \mathrm{y}^{(N_t)}] \in \mathbb{R}^{N_x \times (N_t+1)}$.
- The matrix $\bar{\mathrm{Z}} := \mathrm{W}^{1/2}\mathrm{Z}\mathrm{D}^{1/2} \in \mathbb{R}^{N_x \times (N_t+1)}$.

Corollary 4.10. *1. The matrix representation of the operator $\mathcal{R}^{h,n}$ is given by*

$$\mathcal{R}^{h,n} = \mathrm{Z}\mathrm{D}\mathrm{Z}^T\mathrm{W}.$$

2. The solutions to the eigenvalue problem (4.8) *are given by $\mathrm{u}_i = \mathrm{W}^{-1/2}\bar{\mathrm{u}}_i$ with*

$$\bar{\mathrm{Z}}\bar{\mathrm{Z}}^T\bar{\mathrm{u}}_i = \lambda_i \bar{\mathrm{u}}_i, \quad \bar{\mathrm{u}}_i^T \bar{\mathrm{u}}_j = \delta_{ij}, \quad 1 \leq i, j \leq \ell. \tag{4.9}$$

Proof. 1. Having a look at the i-th component of the matrix multiplication $\mathrm{Z}\mathrm{D}\mathrm{Z}^T\mathrm{W}$, $i \in \{1, \ldots, N_x\}$, one can see the claimed equality; compare [48, Remark 1.4.4].

2. Multiply $\mathrm{Z}\mathrm{D}\mathrm{Z}^T\mathrm{W}\mathrm{u} = \lambda_i \mathrm{u}$ from the left by $\mathrm{W}^{1/2}$ and define $\bar{\mathrm{u}}_i := \mathrm{W}^{1/2}\mathrm{u}_i$. This yields $\langle \bar{\mathrm{u}}_i, \bar{\mathrm{u}}_j \rangle_{\mathbb{R}^{N_x}} = \langle \mathrm{u}_i, \mathrm{u}_j \rangle_{\mathrm{W}}$.

□

According to [43] we name the different possibilities for determination of the POD basis vectors $\{\mathrm{u}_i\}_{i=1}^{\ell}$:

(1) Solve the $N_x \times N_x$ eigenvalue problem (4.9) for the ℓ largest eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_\ell > 0$ and set $\mathrm{u}_i = \mathrm{W}^{-1/2}\bar{\mathrm{u}}_i$. We will not consider this approach in our numerical examples. It is often, especially in case of $N_x \gg N_t$, numerically expensive due to $\bar{\mathrm{Z}}\bar{\mathrm{Z}}^T = \mathrm{W}^{1/2}\mathrm{Z}\mathrm{D}\mathrm{Z}^T\mathrm{W}^{1/2}$.

(2) **Method of snapshots**: We first solve the $(N_t + 1) \times (N_t + 1)$ eigenvalue problem

$$\bar{\mathrm{Z}}^T\bar{\mathrm{Z}}\bar{\mathrm{v}}_i = \lambda_i \bar{\mathrm{v}}_i, \quad \bar{\mathrm{v}}_i^T \bar{\mathrm{v}}_j = \delta_{ij}, \quad 1 \leq i, j \leq \ell,$$

for the ℓ largest eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_\ell > 0$. Then we set (by SVD)

$$\mathrm{u}_i = \frac{1}{\sqrt{\lambda_i}} \mathrm{Z}\mathrm{D}^{1/2}\bar{\mathrm{v}}_i.$$

We have $\bar{\mathrm{Z}}^T\bar{\mathrm{Z}} = \mathrm{D}^{1/2}\mathrm{Z}^T\mathrm{W}\mathrm{Z}\mathrm{D}^{1/2}$. Hence, this approach does not require the expensive computation of $\mathrm{W}^{1/2}$ whereas $\mathrm{D}^{1/2} = \operatorname{diag}(\alpha_1^{1/2}, \ldots, \alpha_{N_t}^{1/2})$ is numerically cheap to compute. We call this variant 'eigs' in Chapter 6 and it turns out to be the faster variant. But this strategy is less stable than the following because the multiplication $\bar{Z}^T\bar{Z}$ squares the condition number compared to $\bar{Z}$.

(3) **SVD**: Compute the SVD $\bar{\mathrm{Z}} = \bar{\mathrm{U}}\Sigma\bar{\mathrm{V}}^T$ with orthogonal matrices $\bar{\mathrm{U}} = [\bar{\mathrm{u}}_1, \ldots, \bar{\mathrm{u}}_{N_x}] \in \mathbb{R}^{N_x \times N_x}$ and $\bar{\mathrm{V}} = [\bar{\mathrm{v}}_1, \ldots, \bar{\mathrm{v}}_{N_t+1}] \in \mathbb{R}^{(N_t+1)\times(N_t+1)}$. Thus, $\Sigma = \bar{\mathrm{U}}^T\bar{\mathrm{Z}}\bar{\mathrm{V}} \in \mathbb{R}^{N_x\times(N_t+1)}$ is of form

$$\Sigma = \left(\begin{array}{ccc|c} \sigma_1 & & & 0 \\ & \ddots & & \vdots \\ & & \sigma_d & 0 \\ \hline 0 & \ldots & 0 & 0 \end{array}\right)$$

with singular values $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_d > 0$. This gives $\mathrm{u}_i = \mathrm{W}^{-1/2}\bar{\mathrm{u}}_i$ and $\lambda_i = \sigma_i^2$. More details about SVD can be found, for instance, in [10]. In Chapter 6 we call this variant 'SVD'.

Remark 4.11. (1) For our numerical computations we made the choice of using the same, only one, POD basis for the occuring PDEs. To obtain adequate results we will have to take both state and adjoint snapshots to set up the POD basis. The adjoint snapshots $\{\mathrm{p}^{(j)}\}_{j=0}^{N_t}$ are incorporated as follows: Replace the above snapshot matrix Z by

$$[\mathrm{y}^{(0)}, \ldots, \mathrm{y}^{(N_t)}, \mathrm{p}^{(0)}, \ldots, \mathrm{p}^{(N_t)}] \in \mathbb{R}^{N_x\times(2N_t+2)},$$

and the time-weights matrix D by $\begin{pmatrix} \mathrm{D} & 0 \\ 0 & \mathrm{D} \end{pmatrix} \in \mathbb{R}^{(2N_t+2)\times(2N_t+2)}$.

(2) Let the POD basis matrix $\Psi \in \mathbb{R}^{N_x\times\ell}$ be given. The corresponding $\boldsymbol{x}$-dependent POD basis functions equal

$$\psi_j(\boldsymbol{x}) = \sum_{i=1}^{N_x} \Psi_{ij}\varphi_i(\boldsymbol{x}), \quad j = 1, \ldots, \ell.$$

4.3 POD Galerkin Discretization

The FE Galerkin method for discretizing the spatial variable of (**P**) was introduced in Section 3.2. Now, we apply a POD Galerkin discretization. Note that in discrete version, the POD basis functions themselves are linear combinations of the underlying FE ansatz functions; see Remark 4.11(2).

Suppose that we have determined a POD basis of rank $\ell \leq d$ solving problem (4.2) with $\mathcal{V}$ given as in (4.1). We define the ℓ-dimensional POD subspace by

$$V^\ell := \operatorname{span}\{\psi_1, \ldots, \psi_\ell\} \subset V.$$

It is a Hilbert space endowed with the inner product in V.

We proceed analogously to Section 3.2 and start with a detailed consideration of the state equation.

State equation:
The set $\{\psi_i\}_{i=1}^d$ is an orthonormal basis of $\mathcal{V}$. Hence, the Fourier expansion

$$y(t) = \sum_{i=1}^{d} \langle y(t), \psi_i \rangle_V \psi_i$$

is given for all $t \in [0,T]$. We introduce the function $y^\ell(t) \in V^\ell$ by

$$y^\ell(t) := \sum_{i=1}^{\ell} \underbrace{\langle y^\ell(t), \psi_i \rangle_V}_{=:\mathrm{y}_i^\ell(t)} \psi_i \quad \text{for all } t \in [0,T]. \tag{4.10}$$

It is an approximation for $y(t)$ if $\ell < d$. The POD based variational formulation of (SE) now reads: Find a function $y^\ell(t) = \sum_{i=1}^{\ell} \mathrm{y}_i^\ell(t)\psi_i \in V^\ell$ such that

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}\langle c_p y^\ell(t), \psi_j \rangle_H + a(y^\ell(t), \psi_j) + \langle \mathcal{N}(y^\ell(t)), \psi_j \rangle_{V',V} &= \langle (F + \mathcal{B}u)(t), \psi_j \rangle_{V',V}, \\ \langle y^\ell(0), \psi_j \rangle_H &= \langle y_0, \psi_j \rangle_H, \end{aligned} \tag{4.11}$$

holds for $j = 1, \ldots, \ell$, a.e. in $(0,T]$. System (4.11) is a low-dimensional approximation and thus a reduced-order model (ROM) for (3.13). We assume that it admits a unique solution y^ℓ in $H^1(0,T;V^\ell)$. As stated in [24, Remark 3.4], this is not a-priori clear. The solution is denoted by $y^\ell(u)$. It belongs to $W(0,T)$ due to $V^\ell \hookrightarrow V$.

Let us derive the semi-discrete scheme for (4.11). We insert the POD Galerkin ansatz (4.10) for y^ℓ. This yields the following ℓ-dimensional nonlinear ODE system for the vector $\mathrm{y}^\ell(t) := \big(\mathrm{y}_1^\ell(t), \ldots, \mathrm{y}_\ell^\ell(t)\big)^T$:

$$\begin{aligned} c_p \mathrm{M}^\ell \dot{\mathrm{y}}^\ell(t) + \mathrm{S}^\ell \mathrm{y}^\ell(t) + \mathrm{N}^\ell(\mathrm{y}^\ell(t)) &= \mathrm{F}^\ell(t) + \mathrm{B}^\ell u(t) \quad \text{a.e. in } (0,T], \\ \mathrm{M}^\ell \mathrm{y}^\ell(0) &= \mathrm{y}_0^\ell, \end{aligned} \tag{4.12}$$

with the matrices and vectors

$$\begin{aligned} \mathrm{M}_{ij}^\ell &:= \langle \psi_i, \psi_j \rangle_H, \quad (\mathrm{y}_0^\ell)_j := \langle y_0, \psi_j \rangle_H, \\ \mathrm{S}_{ij}^\ell &:= a(\psi_i, \psi_j) = \langle \nabla\psi_i, \nabla\psi_j \rangle_H + q\langle \psi_i, \psi_j \rangle_{L^2(\Gamma)}, \\ \mathrm{B}_{jk}^\ell &:= \langle \chi_k, \psi_j \rangle_{L^2(\Gamma)} = \int_{\Gamma_k} \psi_j(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}, \\ (\mathrm{F}^\ell(t))_j &:= \langle F(t), \psi_j \rangle_{V',V} = \int_\Omega f(t,\boldsymbol{x})\psi_j(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}, \\ \big(\mathrm{N}^\ell(\mathrm{y}^\ell)\big)_j &:= \int_\Omega N\big(\sum_{i=1}^{\ell} \mathrm{y}_i^\ell \psi_i(\boldsymbol{x})\big)\psi_j(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \quad \text{for } \mathrm{y}^\ell \in \mathbb{R}^\ell, \end{aligned}$$

for $1 \le i,j \le \ell$, $1 \le k \le \mathsf{k}$ and f.a.a. $t \in (0,T]$.

For the linearized state equation and the adjoint equations we directly name the POD based semi-discretizations.

Linearized state equation:
With

$$v^\ell(t) := \sum_{i=1}^{\ell} \mathrm{v}_i^\ell(t)\psi_i,\ \mathrm{v}^\ell(t) := \left(\mathrm{v}_1^\ell(t), \dots, \mathrm{v}_\ell^\ell(t)\right)^T,\ \text{ for } t \in [0,T]$$

we obtain

$$\begin{aligned} c_p \mathrm{M}^\ell \dot{\mathrm{v}}^\ell(t) + \left(\mathrm{S}^\ell + (\mathrm{N}^\ell)'(\mathrm{y}^\ell(t))\right)\mathrm{v}^\ell(t) = \mathrm{B}^\ell s(t) \quad \text{a.e. in } (0,T], \\ \mathrm{v}^\ell(0) = 0. \end{aligned} \tag{4.13}$$

Adjoint equations:
The adjoint equations are treated via equation (3.22) with the ansatz

$$p^\ell(t) := \sum_{i=1}^{\ell} \mathrm{p}_i^\ell(t)\psi_i,\ \mathrm{p}^\ell(t) := \left(\mathrm{p}_1^\ell(t), \dots, \mathrm{p}_\ell^\ell(t)\right)^T,\ \text{ for } t \in [0,T].$$

We define $\left(\mathrm{r}_\mathrm{Q}^\ell(t)\right)_j := \int_\Omega r_Q(t,\boldsymbol{x})\psi_j(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}$ and $(\mathrm{r}_\mathrm{T}^\ell)_j := \langle r_T, \psi_j\rangle_H,\ j = 0, \dots, N_t$. This gives

$$\begin{aligned} -c_p \mathrm{M}^\ell \dot{\mathrm{p}}^\ell(t) + \left(\mathrm{S}^\ell + (\mathrm{N}^\ell)'(\mathrm{y}^\ell(t))\right)\mathrm{p}^\ell(t) = \mathrm{r}_\mathrm{Q}^\ell(t) \quad \text{a.e. in } [0,T), \\ c_p \mathrm{M}^\ell \mathrm{p}^\ell(T) = \mathrm{r}_\mathrm{T}^\ell. \end{aligned} \tag{4.14}$$

For time integration of the ODE systems (4.12), (4.13) and (4.14) we apply the implicit Euler method analogously to the solution of the FE based semi-discretizations.

We define a reduced cost functional based on a POD Galerkin approximation of the state variable by

$$\hat{J}^\ell(u) := J(y^\ell(u), u). \tag{4.15}$$

Furthermore, the above POD based variational formulations combined provide a POD Galerkin approximation of the gradient $\hat{J}'(u)$ and of the Hessian $\hat{J}''(u)$. These three approximations can now be utilized for numerical application of the (globalized) Newton-CG strategy. This yields a control $\bar{u}^\ell$ that is interpreted as suboptimal control for (**P**).

Remark 4.12. Suppose that we have computed the POD basis matrix $\Psi \in \mathbb{R}^{N_x \times \ell}$ as described in Section 4.2.2. Let $t \in [0,T]$. Similar to Remark 4.11(2), we write

$$y^\ell(t) = \sum_{i=1}^{\ell} \mathrm{y}_i^\ell(t)\psi_i = \sum_{i=1}^{\ell} \mathrm{y}_i^\ell(t) \sum_{j=1}^{N_x} \Psi_{ji}\varphi_j = \sum_{j=1}^{N_x} \left(\Psi \mathrm{y}^\ell(t)\right)_j \varphi_j.$$

Thus, the FE coefficients of $y^\ell(t)$ are obtained by multiplying $\Psi \mathrm{y}^\ell(t) \in \mathbb{R}^{N_x}$. In the same way we deduce

$$\begin{aligned} &\mathrm{M}^\ell = \Psi^T \mathrm{M}\Psi, \quad \mathrm{S}^\ell = \Psi^T \mathrm{S}\Psi, \quad \mathrm{B}^\ell = \Psi^T \mathrm{B}, \quad \mathrm{y}_0^\ell = \Psi^T \mathrm{y}_0, \quad \mathrm{r}_\mathrm{T}^\ell = \Psi^T \mathrm{r}_\mathrm{T}, \\ &\mathrm{F}^\ell(t) = \Psi^T \mathrm{F}(t), \quad \mathrm{r}_\mathrm{Q}^\ell(t) = \Psi^T \mathrm{r}_\mathrm{Q}(t) \quad \text{f.a.a. } t \in [0,T], \\ &\mathrm{N}^\ell(\mathrm{y}^\ell) = \Psi^T \mathrm{N}(\Psi \mathrm{y}^\ell) \quad \text{for all } \mathrm{y}^\ell \in \mathbb{R}^\ell, \end{aligned} \tag{4.16}$$

4.4 Empirical Interpolation Methods

The solution of system (4.12) requires the evaluation of the nonlinear term $\mathrm{N}^\ell(\mathrm{y}^\ell)$ with Jacobian $(\mathrm{N}^\ell)'(\mathrm{y}^\ell)$ for vectors $\mathrm{y}^\ell \in \mathbb{R}^\ell$. The Jacobian additionally occurs in the semi-discretizations (4.13) and (4.14). From the last equality in (4.16), it follows

$$\mathrm{N}^\ell(\mathrm{y}^\ell) = \underbrace{\Psi^T}_{\ell\times N_x}\underbrace{\mathrm{N}(\Psi\mathrm{y}^\ell)}_{N_x\times 1}, \quad (\mathrm{N}^\ell)'(\mathrm{y}^\ell) = \underbrace{\Psi^T}_{\ell\times N_x}\underbrace{\mathrm{N}'(\Psi\mathrm{y}^\ell)}_{N_x\times N_x}\underbrace{\Psi}_{N_x\times\ell}, \quad \text{for any } \mathrm{y}^\ell \in \mathbb{R}^\ell. \tag{4.17}$$

This illustrates that the ROMs are actually not independent of the FE dimension N_x. As done in [34] the evaluation of the nonlinear terms can be described as follows: First, $\mathrm{y}^\ell \in \mathbb{R}^\ell$ is expanded to a vector $\Psi\mathrm{y}^\ell$ of the given full dimension N_x. Second, the nonlinearity $\mathrm{N}(\Psi\mathrm{y}^\ell)$ or the Jacobian $\mathrm{N}'(\Psi\mathrm{y}^\ell) \in \mathbb{R}^{N_x\times N_x}$, respectively, is evaluated. Finally, the result is reduced back to the low dimension ℓ.

To reduce this computational complexity we investigate variants of the empirical interpolation method (EIM). It was originally proposed in [3]. Here, we use the EIM developed in [33] and the discrete empirical interpolation method (DEIM) as introduced in [7]. The following argumentation is based on [7, 33, 34, 48].

We directly use the approximation

$$\mathrm{N}(\mathrm{y}) \approx \mathrm{M}N(\mathrm{y}) \quad \text{for } \mathrm{y} \in \mathbb{R}^{N_x}$$

from Section 3.2. This gives

$$\mathrm{N}^\ell(\mathrm{y}^\ell) \approx \Psi^T\mathrm{M}N(\Psi\mathrm{y}^\ell) \quad \text{for } \mathrm{y}^\ell \in \mathbb{R}^\ell. \tag{4.18}$$

Let $\mathrm{y}^\ell : [0,T] \to \mathbb{R}^\ell$. EIM is used to approximate the term

$$n(t) := N(\Psi\mathrm{y}^\ell(t)) \in \mathbb{R}^{N_x} \quad \text{for } t \in [0,T].$$

We apply a Galerkin ansatz of form

$$n(t) \approx \sum_{i=1}^{\ell^{EIM}} c_i(t)\phi_i = \Phi c(t) =: \hat{n}(t), \tag{4.19}$$

with $\ell^{EIM} \ll N_x$ linearly independent basis functions $\phi_1, \dots, \phi_{\ell^{EIM}} \in \mathbb{R}^{N_x}$, the coefficient vector $c(t) := (c_1(t), \dots, c_{\ell^{EIM}}(t))^T \in \mathbb{R}^{\ell^{EIM}}$ and the basis matrix $\Phi := [\phi_1, \dots, \phi_{\ell^{EIM}}] \in \mathbb{R}^{N_x\times\ell^{EIM}}$.

The computation of the matrix Φ is done by EIM or DEIM while the coefficient vector is determined using interpolation. The EIM and DEIM algorithms do not only return the matrix Φ but also an index vector $\wp \in \mathbb{R}^{\ell^{EIM}}$. This vector contains the needed interpolation indices associated with Φ. Before we come to the algorithms we discuss the computation of $c(t)$ assuming Φ and $\wp$ are given.

The system $n(t) = \Phi c(t)$ is overdetermined. We choose ℓ^{EIM} distinguished rows of $n(t)$ and Φ such that the resulting square matrix is invertible. Then, $c(t)$ is uniquely determined. The chosen rows are those corresponding to the indices in $\wp$. For this reason, for any matrix $B \in \mathbb{R}^{N_x\times\ell^{EIM}}$ we denote by $B_{\{\wp\}} \in \mathbb{R}^{\ell^{EIM}\times\ell^{EIM}}$ the submatrix consisting of the rows of B corresponding to the indices in $\wp$. In order to simplify notation, we consider

the matrix $P = [e_{\wp_1}, \dots, e_{\wp_{\ell^{EIM}}}] \in \mathbb{R}^{N_x \times \ell^{EIM}}$, where $e_{\wp_i} = (0, \dots, 0, 1, 0, \dots, 0)^\top \in \mathbb{R}^{N_x}$ is the $\wp_i$-th unit vector. The equality $B_{\{\wp\}} = P^T B$ holds for any $B \in \mathbb{R}^{N_x \times \ell^{EIM}}$. And $P^T n(t) = (P^T \Phi) c(t)$ yields

$$c(t) = (P^T\Phi)^{-1} P^T n(t) = (P^T\Phi)^{-1} \underbrace{\left(P^T N(\Psi \mathrm{y}^\ell(t))\right)}_{=\left(N(\Psi \mathrm{y}^\ell(t))\right)_{\{\wp\}}} \quad \text{for } t \in [0,T].$$

The matrix P^T can be moved into the nonlinearity N since it evaluates pointwise. Thus, we arrive at

$$c(t) = (P^T\Phi)^{-1} N(P^T\Psi \mathrm{y}^\ell(t)) \overset{(4.19)}{\Rightarrow} \hat{n}(t) = \Phi(P^T\Phi)^{-1} N(P^T\Psi \mathrm{y}^\ell(t)).$$

For the extension to general nonlinear functions we refer to [7]. We can easily show that the approximation is exact at the interpolation points:

$$P^T\hat{n}(t) = P^T\Phi(P^T\Phi)^{-1} N(P^T\Psi \mathrm{y}^\ell(t)) = N(P^T\Psi \mathrm{y}^\ell(t)) = P^T n(t).$$

Now, approximation (4.18) yields

$$\mathrm{N}^\ell(\mathrm{y}^\ell(t)) \approx \Psi^T \mathrm{M} \Phi \, (P^T\Phi)^{-1} N(P^T\Psi \mathrm{y}^\ell(t)).$$

For the Jacobian we obtain

$$(\mathrm{N}^\ell)'(\mathrm{y}^\ell(t)) \approx \Psi^T \mathrm{M} \Phi \, (P^T\Phi)^{-1} \operatorname{diag}\left(N'((P^T\Psi \mathrm{y}^\ell(t))_1), \dots, N'((P^T\Psi \mathrm{y}^\ell(t))_{\ell^{EIM}})\right) P^T\Psi.$$

The matrices

$$\Psi^T \mathrm{M} \Phi \in \mathbb{R}^{\ell \times \ell^{EIM}}, \quad \mathrm{P}^T\Phi \in \mathbb{R}^{\ell^{EIM} \times \ell^{EIM}}, \quad P^T\Psi \in R^{\ell^{EIM} \times \ell},$$

are independent of the full dimension N_x. They can be precomputed (offline) so that N and N' have to be evaluated only at the ℓ^{EIM} interpolation points (online).

It remains to discuss the EIM and DEIM algorithms. Let $\mathrm{y}^{(j)}$, $j = 0, \dots, N_t$, be the state snapshots that get computed to set up the POD basis. While solving the fine full-discretization of (SE) we also obtain the corresponding nonlinear snapshots $N(\mathrm{y}^{(j)}) \in \mathbb{R}^{N_x}$, $j = 0, \dots, N_t$. They are stored and utilized via the matrix

$$\mathrm{E} := [N(\mathrm{y}^{(0)}), \dots, N(\mathrm{y}^{(N_t)})] \in \mathbb{R}^{N_x \times (N_t+1)}.$$

We denote by $\|\cdot\|_\infty$ the maximum norm in $\mathbb{R}^{N_x}$ and the operation $\arg\max$ returns the index where the maximal entry occurs. Algorithm 4 is Algorithm 2 in [34] and Algorithm 5 is Algorithm 2 in [33]. They are both adjusted to our data and notation. While the EIM implementation is just based on a greedy algorithm the DEIM implementation uses a combination of a POD approach and a greedy algorithm.

The DEIM algorithm requires ℓ^{EIM} at the beginning. Then, the POD method in $\mathbb{R}^{N_x}$ is applied to the nonlinear snapshot ensemble, the matrix E, for computation of the matrix Φ; see [48, Section 1.1-1.2]. In a second step, the index vector $\wp$ is determined.

The EIM algorithm requires a tolerance ϵ^{EIM} for the accepted interpolation error. The number ℓ^{EIM} is chosen appropriately. The columns of Φ and the interpolation indices are

both chosen successively.

Algorithm 4 Discrete empirical interpolation method (DEIM)

Require: A number ℓ^{EIM} and the matrix $\mathrm{E} = [N(\mathrm{y}^{(0)}), \ldots, N(\mathrm{y}^{(N_t)})] \in \mathbb{R}^{N_x \times (N_t+1)}$;

1: Compute POD basis matrix $\Phi = [\phi_1, \ldots, \phi_{\ell^{EIM}}]$ for E;
2: $\mathrm{idx} \leftarrow \arg\max_{j=0,\ldots,N_t} |(\phi_1)_j|$;
3: $U = [\phi_1]$ and $\wp = \mathrm{idx}$;
4: **for** $\ell = 2$ to ℓ^{EIM} **do**
5: $\quad u \leftarrow \phi_\ell$;
6: $\quad$ Solve $U_{\{\wp\}} c = u_{\{\wp\}}$;
7: $\quad r \leftarrow u - Uc$;
8: $\quad \mathrm{idx} \leftarrow \arg\max_{j=0,\ldots,N_t} |(r)_j|$;
9: $\quad U \leftarrow [U, u]$ and $\wp \leftarrow [\wp, \mathrm{idx}]$;
10: **end for**
11: **return** Φ and $\wp$;

Algorithm 5 Empirical interpolation method (EIM)

Require: a maximal number ℓ_{max}^{EIM}, a constant $\epsilon^{EIM} > 0$ and the matrix $\mathrm{E} = [N(\mathrm{y}^{(0)}), \ldots, N(\mathrm{y}^{(N_t)})] \in \mathbb{R}^{N_x \times (N_t+1)}$;

1: Set $\ell = 1$;
2: $k \leftarrow \arg\max_{j=0,\ldots,N_t} \|N(\mathrm{y}^{(j)})\|_\infty$;
3: $\xi \leftarrow N(\mathrm{y}^{(k)})$;
4: $\mathrm{idx} \leftarrow \arg\max_{i=1,\ldots,N_x} |\xi_i|$;
5: $\phi_1 \leftarrow \xi / \xi_{\mathrm{idx}}$;
6: $\Phi = [\phi_1]$ and $\wp = \mathrm{idx}$;
7: Solve $\Phi_{\{\wp\}}\, c_j = N(\mathrm{y}^{(j)})_{\{\wp\}}$ for $j = 0, \ldots, N_t$;
8: $k \leftarrow \arg\max_{j=0,\ldots,N_t} \|N(\mathrm{y}^{(j)}) - \Phi c_j\|_\infty$;
9: **while** $\|N(\mathrm{y}^{(k)}) - \Phi c_k\|_\infty > \epsilon^{EIM}$ and $\ell < \ell_{max}^{EIM}$ **do**
10: $\quad \ell \leftarrow \ell + 1$;
11: $\quad \xi \leftarrow N(\mathrm{y}^{(k)})$;
12: $\quad \mathrm{idx} \leftarrow \arg\max_{i=1,\ldots,N_x} |(\xi - \Phi c_k)_i|$;
13: $\quad \phi_\ell \leftarrow (\xi - \Phi c_k) / (\xi - \Phi c_k)_{\mathrm{idx}}$;
14: $\quad \Phi \leftarrow [\Phi, \phi_\ell]$ and $\wp \leftarrow [\wp, \mathrm{idx}]$;
15: $\quad$ Solve $\Phi_{\{\wp\}}\, c_j = N(\mathrm{y}^{(j)})_{\{\wp\}}$ for $j = 0, \ldots, N_t$;
16: $\quad k \leftarrow \arg\max_{j=0,\ldots,N_t} \|N(\mathrm{y}^{(j)}) - \Phi c_j\|_\infty$;
17: **end while**
18: **return** Φ, $\wp$ and $\ell^{EIM} = \ell$;

Let us briefly compare the computational expenses of the two methods as done in [33]:

(1) Offline computation:
Comparing line 6 in Algorithm 4 with line 7/15 in Algorithm 5 shows: The EIM algorithm requires to solve $N_t + 1$ linear systems of equations in each iteration and in the DEIM algorithm only one solve has to be done. But the DEIM algorithm requires a POD basis computation.

(2) Online computation:
The matrix $\Phi_{\{\wp\}}$ obtained by the EIM algorithm is a lower triangular one. The matrix $\Phi_{\{\wp\}}$ computed with DEIM has no special structure. To reduce the costs for computing the DEIM coefficient vector one can precompute an LU decomposition of $\Phi_{\{\wp\}}$. Then, the evaluation of the nonlinearity using EIM compared to DEIM needs two solves instead of one.

5 Trust Region POD

The ROMs derived in Section 4.3 depend on a certain control. Thus, they might be poor if this control is chosen badly. A globalized Newton-CG method using inaccurate approximations can not deliver satisfactory results.

To overcome this difficulty, we pursue the approach of successively improving the POD basis in the course of optimization. An alternative is to compute a better initializing control for POD optimization. This is done in [16] by using optimality system POD (OS-POD); see also [31].

In [2] a trust region POD (TR-POD) approach has been proposed that was likewise used in [41]. These references show how to realize updating the POD basis. There, a trust region method with the non-quadratic model function $m^{(k)}(d) := \hat{J}^{\ell}(u^{(k)} + d)$ is used. Here, we embed the POD based Newton-CG strategy in a trust region framework. In doing so, we can ensure convergence to a stationary point of the reduced cost functional.

In this chapter we proceed as follows: In Section 5.1 we first recall the principles of trust region methods that use a quadratic approximation for the cost functional. In a second step, we investigate the generalization to quadratic model functions with inexact gradient information. This ensures the well understanding of the TR-POD algorithm, which is defined in Section 5.2. We do not go into detail about convergence theory. This has been done in [13, 40]. A very good introduction to trust region methods is given in [47, Chapter II, Section 14], or see also [26, Section 3.3].

5.1 Trust Region Methods

Let us specify the setting for this section: Analogously to Section 3.1 we consider the unconstrained minimization problem

$$\min_{u \in U} \hat{J}(u), \tag{5.1}$$

where U denotes a real Hilbert space. The cost functional $\hat{J} : U \to \mathbb{R}$ is assumed to satisfy the following assumption (equal to (**A4**)):

(**A7**) $\hat{J}$ is twice continuously Fréchet differentiable and bounded from below.

Trust region methods proceed as follows for the solution of problem (5.1): At a current iterate $u^{(k)}$ the cost functional is replaced by a 'simple' model function which is then approximately minimized in a so-called trust region to find the next iterate.

We introduce a quadratic model for $\hat{J}$ at the current iterate $u^{(k)}$ by

$$m^{(k)}(d) := \hat{J}^{(k)} + \langle g^{(k)}, d \rangle_U + \frac{1}{2} \langle H^{(k)} d, d \rangle_U, \tag{5.2}$$

where $\hat{J}^{(k)} := \hat{J}(u^{(k)})$, $g^{(k)} := \hat{J}'(u^{(k)})$ and $H^{(k)} := \hat{J}''(u^{(k)})$. As noted in Remark 3.1, the global minimizer of $m^{(k)}$ is the Newton step if $H^{(k)}$ is positive definite.

The model $m^{(k)}(d)$ is the quadratic Taylor expansion of $\hat{J}(u^{(k)} + \cdot)$ at $d = 0$. Therefore, it is a good approximation to $\hat{J}(u^{(k)} + d)$ in a small region around $d = 0$. We refer to this region in which we can trust the model $m^{(k)}$ to adequately approximate $\hat{J}(u^{(k)} + \cdot)$ as trust region. It is given by $\{d \in U \mid \|d\|_U \leq \Delta^{(k)}\}$ with trust region radius $\Delta^{(k)}$.

In the following we allow the operator $H^{(k)}$ to be a self-adjoint approximation to the Hessian $\hat{J}''(u^{(k)})$. Hence, in addition to (**A7**) we have to demand

(**A8**) It exists $C_H > 0$ such that $\|H^{(k)}\|_{\mathcal{L}(U,U)} \leq C_H$ for all k.

To find the next iterate one approximately solves the

trust region subproblem

$$\min m^{(k)}(d) \quad \text{s.t.} \quad \|d\|_U \leq \Delta^{(k)}.$$

We refer to the trial step $d_t^{(k)}$ as the solution to the trust region subproblem. The associated trial point is given by $u^{(k)} + d_t^{(k)}$. Having determined the trial step $d_t^{(k)}$, two choices have to be made: Do we accept the step and/or do we change the trust region radius? These questions are answered by estimating the quality of the trial step. We introduce the predicted and actual reduction by

$$\begin{aligned} \mathsf{pred}^{(k)}(d) &:= m^{(k)}(0) - m^{(k)}(d), \\ \mathsf{ared}^{(k)}(d) &:= \hat{J}^{(k)} - \hat{J}(u^{(k)} + d). \end{aligned}$$

Then, the ratio

$$\rho^{(k)}(d_t^{(k)}) := \frac{\mathsf{ared}^{(k)}(d_t^{(k)})}{\mathsf{pred}^{(k)}(d_t^{(k)})}$$

is used to decide whether the reduction $\mathsf{ared}^{(k)}(d_t^{(k)})$ in the cost function compared to the reduction $\mathsf{pred}^{(k)}(d_t^{(k)})$ in the model function is satisfactory or not.

A basic trust region algorithm is defined in Algorithm 6 according to [41, Algorithm 5.1]. We do not specify a termination criterion. In practice the criterion from Algorithm 3 is added. As done in [41] the trust region procedure can be divided into the following four major parts:

1. Line 3: Build up the model function $m^{(k)}(d)$. This requires the computation of the current gradient and the setup of the current Hessian application.

2. Line 4: Computation of a trial step $d_t^{(k)}$.

3. Line 5: $\hat{J}(u^{(k)} + d_t^{(k)})$ has to be computed to evaluate $\rho^{(k)}(d_t^{(k)})$.

4. Lines 6 to 12: Update of the current iterate and the trust region radius:
   The ratio $\rho^{(k)}(d_t^{(k)})$ is used to decide whether the trial point $u^{(k)} + d_t^{(k)}$ is accepted as new iterate or not and whether the trust region radius $\Delta^{(k)}$ has to be changed or not. At this point it is distinguished between the following cases:

   (a) *Accept the trial step* if $\rho^{(k)}(d_t^{(k)}) \geq \eta_1$ (e.g. $\eta_1 = 0.2$):
   We say that the step is successful. The decrease in the cost function compared

Algorithm 6 (Basic trust region algorithm)

Require: An initial trust region radius $\Delta^{(0)} > 0$, an initial point $u^{(0)} \in U$ and constants $\eta_1, \eta_2, \gamma_1, \gamma_2, \gamma_3$ satisfying

$$0 < \eta_1 \leq \eta_2 < 1, \quad 0 < \gamma_1 \leq \gamma_2 < 1 \leq \gamma_3;$$

1: Build up the model function $m^{(k)}(d)$;
2: Determine an approximate solution $d_t^{(k)} \in U$ to the trust region subproblem;
3: Compute $\hat{J}(u^{(k)} + d_t^{(k)})$ and $\rho^{(k)}(d_t^{(k)})$;
4: **if** $\rho^{(k)} \geq \eta_2$ **then**
5: Set $u^{(k+1)} = u^{(k)} + d_t^{(k)}$ and $\Delta^{(k+1)} \in [\Delta^{(k)}, \gamma_3 \Delta^{(k)}]$;
6: **else if** $\eta_1 \leq \rho^{(k)} < \eta_2$ **then**
7: Set $u^{(k+1)} = u^{(k)} + d_t^{(k)}$ and $\Delta^{(k+1)} \in [\gamma_2 \Delta^{(k)}, \Delta^{(k)}]$;
8: **else if** $\rho^{(k)} < \eta_1$ **then**
9: Set $u^{(k+1)} = u^{(k)}$ and $\Delta^{(k+1)} \in [\gamma_1 \Delta^{(k)}, \gamma_2 \Delta^{(k)}]$;
10: **end if**
11: Set $k = k + 1$ go back to step 1.

to the decrease in the model function is satisfactory. More precisely this means that $\mathsf{ared}^{(k)}(d_t^{(k)})$ is at least a fixed fraction of $\mathsf{pred}^{(k)}(d_t^{(k)})$.

- If further $\rho^{(k)}(d_t^{(k)}) \geq \eta_2$ (e.g. $\eta_2 = 0.8$): The approximation quality of the model function on the trust region is as good that the trust region radius can be increased, at most by the factor γ_3 to be precise.
- If $\rho^{(k)}(d_t^{(k)}) < \eta_2$: The radius should be kept or decreased, at most by the factor γ_2.

(b) *Reject the trial step* if $\rho^{(k)}(d_t^{(k)}) < \eta_1$:
The iteration is unsuccessful. The approximation quality of the model function on the trust region is poor. Hence, the trust region radius has to be decreased, at most by the factor γ_1 and at least by the factor γ_2.

Global convergence results for Algorithm 6 are based on a sufficient predicted reduction, i.e. a sufficient decrease in the model function. Then, this sufficient decrease carries over to the cost function in case of an accepted trial step $d_t^{(k)}$ due to $\mathsf{ared}^{(k)}(d_t^{(k)}) \geq \eta_1 \mathsf{pred}^{(k)}(d_t^{(k)})$.

The sufficient predicted reduction has to be guaranteed by the approximative solutions to the trust region subproblems. Here, the Cauchy steps are of central importance.

Definition 5.1. The Cauchy step $d_C^{(k)}$ is the unique solution to

$$\min m^{(k)}(d) \quad \text{s.t.} \quad d = -tg^{(k)},\ t \geq 0,\ \|d\|_U \leq \Delta^{(k)}. \tag{5.3}$$

For the following remark see [41, Remark 5.1.1] and [47, Chapter II, Lemma 14.6].

Remark 5.2. Definition 5.1 means that the Cauchy step $d_C^{(k)}$ is the unique minimizer of $m^{(k)}$ along the steepest descent direction $-tg^{(k)}$ s.t. the trust region bound. It is given by

$d_C^{(k)} = -t_k g^{(k)}$ with

$$t_k = \begin{cases} \dfrac{\|g^{(k)}\|_U^2}{\langle H^{(k)} g^{(k)}, g^{(k)} \rangle_U}, & \text{if } t_k \|g^{(k)}\|_U \leq \Delta^{(k)} \text{ and } \langle H^{(k)} g^{(k)}, g^{(k)} \rangle_U > 0, \\ \dfrac{\Delta^{(k)}}{\|g^{(k)}\|_U}, & \text{else.} \end{cases} \tag{5.4}$$

Moreover, the Cauchy step $d_C^{(k)}$ satisfies

$$\mathsf{pred}^{(k)}(d_C^{(k)}) \geq \frac{1}{2} \|g^{(k)}\|_U \min \left\{ \Delta^{(k)}, \frac{\|g^{(k)}\|_U}{\|H^{(k)}\|_{\mathcal{L}(U,U)}} \right\}. \tag{5.5}$$

Estimate (5.5) provides a sufficient predicted reduction in order to prove global convergence. In addition, the Cauchy step is not costly to compute. But taking it as trial step in each trust region iteration s a steepest descent algorithm which is known to often perform poorly; see [14, Chapter 8].

The crucial aspect is that we can take the Cauchy step to decide whether any other approximative solution to the trust region subproblem is acceptable in the sense of yielding global convergence. The use of (inexact) Newton steps constitutes a natural choice for other trial steps. In fact, Algorithm 6 is globally convergent if the predicted reduction of the trial steps is at least a fixed fraction of the predicted reduction obtained by the Cauchy steps. According to [47] this requirement results in the following condition:

Fraction of Cauchy decrease: $\exists\, \alpha_{fcd} \in (0,1],\ \beta_{fcd} \geq 1$:

$$\|d_t^{(k)}\|_U \leq \beta_{fcd}\, \Delta^{(k)}, \quad \mathsf{pred}^{(k)}(d_t^{(k)}) \geq c_{fcd}\, \mathsf{pred}^{(k)}(d_C^{(k)}).$$

We name a convergence result for Algorithm 6 in Theorem 5.3. It is proven in [47, Theorem 14.10] for the Euclidean space $U = \mathbb{R}^m$. We assume that the result can be extended to a general Hilbert space what is justified by the proceeding in [13, 41] and the remarks in [9, Section 8.3.1]. In addition, we refer to [19] where an interior-point trust region algorithm for infinite-dimensional problems is considered and global convergence results are proven.

Theorem 5.3. *Assume that* (**A7**)-(**A8**) *hold. Let the sequence* $\{u^{(k)}\}_{k\in\mathbb{N}}$ *be generated by Algorithm 6 where the trial steps satisfy the fraction of Cauchy decrease condition. Then, it holds*

$$\lim_{k\to\infty} \|\hat{J}'(u^{(k)})\|_U = 0.$$

Remark 5.4. The most intuitive approach for the use of Newton steps to solve the trust region subproblems is proposed in [47, Chapter II, Section 14.3]: The Newton step is taken as trial step whenever it exists and whenever it satisfies the fraction of Cauchy decrease condition. If the constant β_{fcd} is larger than one, the Newton step is even allowed to lie outside the current trust region. A more elaborated method is the Dogleg method that performs a smooth transition from the steepest descent to the Newton direction; see [26]. For both methods it can be shown that they become the Newton method, if the iterates converge to a minimum that satisfies the second-order sufficient optimality condition and if the exact Hessian is used.

We now turn to trust region methods that use a quadratic model function with inexact gradient information. Thus, we keep the model function (5.2) but we assume

$$(m^{(k)})'(0) = g^{(k)} \neq \hat{J}'(u^{(k)}).$$

The trust region subproblems can still be solved approximately with Cauchy steps. This delivers the lower bound (5.5) for the predicted decrease. Hence, we can still use the fraction of Cauchy decrease condition to guarantee a sufficient predicted decrease for other trial points.

But in addition to the above theory, we now have to monitor the deviation $\hat{J}'(u^{(k)}) - g^{(k)}$ of the model gradient from the exact one to obtain a convergence result. In literature several error conditions can be found; see [35] or [44]. As done in [2, 13, 40] we choose the relative error condition proposed by Carter in [6]:

Carter condition:

$$\frac{\left\|\hat{J}'(u^{(k)}) - g^{(k)}\right\|_U}{\left\|g^{(k)}\right\|_U} \leq \zeta, \quad \zeta \in (0,1).$$

Using the triangle inequality it is easy to see that a convergence result for $\{g^{(k)}\}_{k\in\mathbb{N}}$ directly carries over to $\{\hat{J}'(u^{(k)})\}_{k\in\mathbb{N}}$, if the sequences fulfill the Carter condition.

Lemma 5.5. *Let $\{g^{(k)}\}_{k\in\mathbb{N}}$ and $\{\hat{J}'(u^{(k)})\}_{k\in\mathbb{N}}$ be sequences in U that satisfy the Carter condition for some $\zeta \in (0,1)$ for all k. Assume that $\lim_{k\to\infty} \|g^{(k)}\|_U = 0$ holds. It follows*

$$\lim_{k\to\infty} \|\hat{J}'(u^{(k)})\|_U = 0.$$

Proof. See Lemma 5.5 in [13]. □

In the next theorem we name a convergence result similar to that of Theorem 5.3.

Theorem 5.6. *Let* (**A7**)-(**A8**) *be satisfied. Assume that $\{u^{(k)}\}_{k\in\mathbb{N}}$ is generated by Algorithm 6 where the trial steps satisfy the fraction of Cauchy decrease condition. If the Carter condition is satisfied for some $\zeta \in (0, 1-\eta_2)$ for all k, it holds*

$$\lim_{k\to\infty} \|\hat{J}'(u^{(k)})\|_U = 0.$$

Proof. In [41, Theorem 5.1.9] the above convergence result is stated for trust region methods that use non-quadratic model functions (this requires a curvature measure to guarantee a sufficient predicted decrease in the sense of (5.5)). Again, we assume that the result can be generalized from the Euclidean to a general Hilbert space. □

5.2 Trust Region POD Algorithm

We derive the TR-POD algorithm for the solution (computation of stationary points) of the reduced problem ($\hat{\mathbf{P}}$) with Hilbert space $U = L^2(0,T;\mathbb{R}^k)$. In Remark 3.4(1) we have shown that the LSNCG method can be applied to ($\hat{\mathbf{P}}$) since the iterates belong to

$L^\infty(0,T;\mathbb{R}^{\mathsf{k}})$ if only the initial iterate belongs to $L^\infty(0,T;\mathbb{R}^{\mathsf{k}})$. The argumentation here is the same.

Recall the approximative model gradient $g^{(k)}$ and the approximative Hessian $H^{(k)}$ from Section 5.1. In the TR-POD approach these are computed from the POD Galerkin approximation of the gradient $\hat{J}'(u^{(k)})$ and of the Hessian $\hat{J}''(u^{(k)})$. Since they depend on the current POD rank ℓ the quadratic model function does so, too. We slightly modify notations to express the dependency:

$$m_\ell^{(k)}(d) := \hat{J}^{(k)} + \langle g_\ell^{(k)}, d\rangle_U + \frac{1}{2}\langle H_\ell^{(k)} d, d\rangle_U. \tag{5.6}$$

Of course, the model function $m_\ell^{(k)}$ depends as well on the control that is utilized to set up the POD basis. In iteration k of the presented adaptive TR-POD algorithm the current iterate $u^{(k)}$ is taken. This requires an update of the POD basis in each iteration. Consequently, the TR-POD algorithm given like that is not very efficient. Further below, we explain the intention of our proceeding.

We define the TR-POD algorithm in Algorithm 7; compare [41, Algorithm 5.3]. Notice

Algorithm 7 (TR-POD algorithm)

Require: An inital trust region radius $\Delta^{(0)} > 0$, an initial control $u^{(0)} \in L^\infty(0,T;\mathbb{R}^{\mathsf{k}})$, an initial number ℓ_0 and a maximal number ℓ_{max} of POD basis functions, $k = 0$, constants $\eta_1, \eta_2, \gamma_1, \gamma_2, \gamma_3, \zeta, \alpha_{fcd}, \beta_{fcd}$ satisfying

$$0 < \eta_1 \le \eta_2 < 1, \quad 0 < \gamma_1 \le \gamma_2 < 1 \le \gamma_3, \quad 0 < \zeta < 1 - \eta_2, \quad \alpha_{fcd} \in (0,1], \beta_{fcd} \ge 1;$$

1: Snapshot generation: Compute state $y(u^{(k)})$ and adjoint state $p(u^{(k)})_1$;
2: Compute a POD basis of rank ℓ_{max} and choose $\ell_0 \le \ell \le \ell_{max}$ such that

$$\|\hat{J}'(u^{(k)}) - g_\ell^{(k)}\|_U \le \zeta \|g_\ell^{(k)}\|_U;$$

3: Build up the model function $m_\ell^{(k)}(d)$;
4: Determine an approximate solution $d_t^{(k)} \in L^\infty(0,T;\mathbb{R}^{\mathsf{k}})$ to the trust region subproblem which satisfies the fraction of Cauchy decrease condition with $\alpha_{fcd}, \beta_{fcd}$;
5: Compute $\hat{J}(u^{(k)} + d_t^{(k)})$ and $\rho^{(k)}(d_t^{(k)})$;
6: **if** $\rho^{(k)} \ge \eta_2$ **then**
7: Set $u^{(k+1)} = u^{(k)} + d_t^{(k)}$ and $\Delta^{(k+1)} \in [\Delta^{(k)}, \gamma_3 \Delta^{(k)}]$
set $k = k+1$, $\ell_0 = \ell$ and go back to step 1;
8: **else if** $\eta_1 \le \rho^{(k)} < \eta_2$ **then**
9: Set $u^{(k+1)} = u^{(k)} + d_t^{(k)}$ and $\Delta^{(k+1)} \in [\gamma_2 \Delta^{(k)}, \Delta^{(k)}]$;
set $k = k+1$, $\ell_0 = \ell$ and go back to step 1;
10: **else if** $\rho^{(k)} < \eta_1$ **then**
11: Set $u^{(k+1)} = u^{(k)}$ and $\Delta^{(k+1)} \in [\gamma_1 \Delta^{(k)}, \gamma_2 \Delta^{(k)}]$;
set $k = k+1$ and go to step 3;
12: **end if**

that the POD basis needs not to be updated in case of an unsuccessful iteration. Two steps have to be further specified.

Managing the gradient accuracy in the sense of Carter (line 2):
In the first iteration as well as after every successful iteration the number ℓ of POD basis functions has to be chosen such that the Carter condition is satisfied. Since we compute state and adjoint snapshots (line 1) we cheaply obtain the gradient $\hat{J}'(u^{(k)})$ via formula (2.20). First, we compute a POD basis of rank ℓ_{max}. Let $\ell = \ell_0$ and define

$$\zeta_\ell^{(k)} := \frac{\|\hat{J}'(u^{(k)}) - g_\ell^{(k)}\|_U}{\|g_\ell^{(k)}\|_U}.$$

As long as $\zeta_\ell^{(k)} < 1 - \eta_2$ is not satisfied, we set $\ell = \ell + 1$; compare [40, Section 5.2.3]. Note that the number ℓ_0 gets updated itself. We expect system dynamics to become more complex when moving towards an optimal control. Hence, in iteration k we assume to need at least as many POD ansatz functions as in the previous iteration $k-1$. The computation of $g_\ell^{(k)}$ requires the solution of the ROMs of rank ℓ of (SE) and (AE_1). To avoid redundant computations one should try to improve the choice of ℓ_0 in each iteration and/or to increase the number ℓ more suitably instead of just setting $\ell = \ell + 1$, if the Carter condition is not satisfied. The reader might question if we can always find an ℓ such that $\zeta_\ell^{(k)} < 1 - \eta_2$ is fulfilled. An analytical estimate for an other optimal control problem can be found in [41, Definition 3.3.1, Theorem 4.2.13]. It requires to compute the POD basis from the actual iterate $u^{(k)}$ and the use of both state and adjoint snapshots. The accuracy of the model Hessian, being the POD Galerkin approximation of $\hat{J}''(u^{(k)})$, depends as well on the chosen POD basis rank ℓ. Notice that this accuracy is not monitored within the presented TR-POD strategy.

Solution of the trust region subproblem (line 3):
We compute the Cauchy step using formula (5.4) and an inexact Newton step (if it exists) by applying Algorithm 2 on $A = H_\ell^{(k)}$ and $b = g_\ell^{(k)}$. Whenever the latter one satisfies the fraction of Cauchy decrease condition we accept it as trial step. If not, we take the Cauchy step; compare Remark 5.4.

If the inexact Newton steps are accepted throughout the iterations, the TR-POD algorithm simply performs a POD based Newton-CG method where the POD models are adaptively updated during optimization. The important thing is: The Carter condition controls the number of POD basis functions such that the trust region framework guarantees convergence to a stationary point of $\hat{J}$ by Theorem 5.6.

In this thesis there lies no prior interest in the efficiency in terms of computing time of the derived TR-POD algorithm. In each iteration FE based computations are involved in the following steps:

1. Line 1: Computation of state and adjoint snapshots for an update of the POD basis.

2. Line 2: Computation of the FE gradient $\hat{J}'(u^{(k)})$ which is needed to test the Carter condition.

3. Line 5: The computation of the actual reduction $\mathsf{ared}^{(k)}(d_t^{(k)})$ requires the evaluation of $\hat{J}(u^{(k)} + d_t^{(k)})$. Hence, the full-order state equation has to be solved for the trial

point $u^{(k)} + d_t^{(k)}$. If the iteration is successful, delivering $u^{(k+1)} = u^{(k)} + d_t^{(k)}$, the snapshots of the state $y(u^{(k+1)})$ are already given.

Since the FE gradient is available we utilize it for a termination criterion analogous to that in the LSNCG algorithm. But notice that the FE evaluations are not involved in optimization. The optimization procedure is completely carried out by using the POD Galerkin approximations, i.e. the ROMs, and yields an optimal control $\bar{u}^{TR}$. Only due to the required actual reduction in line 5, the algorithm returns an associated optimal FE state $\bar{y}^{TR} = y(\bar{u}^{TR})$ and the FE value $\hat{J}(\bar{u}^{TR})$ of the reduced cost functional.

We pursue the aim of showing that the presented TR-POD strategy works very well for the solution of the considered optimal control problem. We hope that our results will motivate the investigation of error estimates that allow to remove as many full-order evaluations in the derived TR-POD algorithm as possible. We have the following modifications in mind:

1. Line 1: The POD basis is only updated when needed.

2. Line 2: If the POD basis is updated in line 1, the Carter condition could be used to monitor the gradient accuracy. But in this case, the better FE gradient can actually be used as gradient of the model function. The number ℓ of POD basis functions must still be chosen appropriately for an accurate POD Galerkin approximation of the Hessian. If the POD basis is kept, an error estimate to replace the Carter condition must be available.

3. Line 5: An estimate for the ratio of actual and predicted reduction that gets along without the FE state associated with the trial point $u^{(k)} + d_t^{(k)}$ has to be derived.

6 Numerical Experiments

In this chapter we present all our numerical results. For implementation we use the MATLAB software package (R2013a). Throughout, the considered spatial domain is the unit square $\Omega := (0,1) \times (0,1) \subset \mathbb{R}^2$ and the considered final time T equals 1. Recall the equidistant time grid $0 = t_0 < t_1 < \ldots < t_{N_t} = T$ from Section 3.2.

We mainly focus on the nonlinearity

$$\boxed{N(y) = y^3 \quad \text{with} \quad N'(y) = 3y^2,\ N''(y) = 6y.}$$

In addition, in the last example we investigate the nonlinear function

$$\boxed{N(y) = -0.5y^3 \quad \text{with} \quad N'(y) = -1.5y^2,\ N''(y) = -3y.}$$

We take a look at the numerical implementation and summarize the basic choices which need to be made:

FE assembly process: We choose piecewise linear elements for the FE method. The maximum edge length h_{max} for the triangulation of Ω can be specified. To generate the grid we use the MATLAB PDE toolbox function `initmesh` that controls the triangle size such that no side exceeds the chosen length h_{max}.
To assemble the mass matrix M and the stiffness matrices A and K we use the PDE toolbox function `assema`. Recall the FE nodes P_i, $i = 1, \ldots, N_x$, and the FE interpolation operator I from Section 3.2. The initial vector y_0 is approximated by $\mathrm{M}(y_0(P_i))_{i=1}^{N_x}$. This is equivalent to the choice of $y^h(0) = Iy_0$. The interpolant approximates the initial function such that the values of Iy_0 and y_0 are the same at the FE nodes. The vector y_d is computed analogously.

Division of the boundary: The boundary of the unit square gets divided into its four boundary parts. Only once, for the solution of a linear-quadratic optimal control problem, each of these four boundary parts is again divided equidistantly into 10 subsegments. This gives $\mathsf{k} = 40$ segments all in all. But apart from that, the considered $\mathsf{k} = 4$ boundary parts $\Gamma_1, \ldots, \Gamma_4$ of the unit square are numbered counterclockwise beginning at the bottom.

POD: We refer to Section 4.2 where we discussed the POD basis computation in detail. The two principal choices are:
1. Weighting matrix: We can set $X = H$ or $X = V$ so that the weighting matrix W is either given by $\mathrm{W} = \mathrm{M}$ or by $\mathrm{W} = \mathrm{M} + \mathrm{A}$.
2. 'eigs' or 'SVD': For 'eigs' we use the MATLAB function `eigs` to compute just the ℓ largest eigenvalues with corresponding eigenvectors of the matrix $\bar{\mathrm{Z}}^T\bar{\mathrm{Z}}$. The matrix $\bar{\mathrm{Z}}^T\bar{\mathrm{Z}}$ is symmetric and positive semi-definite. Thus, in theory its eigenvalues are real and non-negative. Due to numerical rounding errors it can be that

the eigenvalues computed by `eigs` are negative or complex if they are in the range of machine precision. Let us mention that the machine precision of MATLAB is $2.2204 \cdot 10^{-16}$. In order not to obtain complex or negative eigenvalues we take the absolute value of the real parts of the eigenvalues computed by `eigs`. Consequently, it can occur that the eigenvalues start increasing once they became very small.
The variant 'SVD' requires the computation of $\mathrm{W}^{1/2}$ (the matrix W is symmetric positive definite). This is done with the MATLAB function `sqrtm`. To compute the SVD of $\bar{Z}$ we use the MATLAB routine `svd`. Of course, we do not compute the inverse $\mathrm{W}^{-1/2}$ in order to obtain $\mathrm{u}_i = \mathrm{W}^{-1/2}\bar{\mathrm{u}}_i$. We solve the linear system $\mathrm{W}^{1/2}\mathrm{u}_i = \bar{\mathrm{u}}_i$.

As mentioned in Remark 4.4(3), the POD approximation error depends on the decay of the given eigenvalues. To visualize the decay rate we plot the normalized eigenvalues $\bar{\lambda}_i := \lambda_i / \sum_{i=1}^{d} \lambda_i$. The eigenvalues obtained by 'eigs' are denoted like this. In case of 'SVD' the eigenvalues are given as squared singular values. A normalized squared singular value $\bar{\sigma}_i^2$ is defined analogously to $\bar{\lambda}_i$.

The following criterion is often used for an appropriate choice of the POD basis rank ℓ: The sum $\sum_{i=1}^{d} \lambda_i$ is usually referred to as 'total energy' contained in the system or the snapshot ensemble. Hence, the quotient

$$\mathcal{E}(\ell) := \frac{\sum_{i=1}^{\ell} \lambda_i}{\sum_{i=1}^{d} \lambda_i} = \frac{\sum_{i=1}^{\ell} \lambda_i}{\mathrm{trace}(\bar{Z}^T \bar{Z})} \tag{6.1}$$

describes the percentage of captured energy; see [41, 42, 48]. The chosen number ℓ of POD basis functions should provide that $\mathcal{E}(\ell)$ is sufficiently close to one, for instance $\mathcal{E}(\ell) \geq 0.99$. This is related to the description of expressing essential information (see the beginning of Chapter 4).

W-orthonormality of the POD basis vectors when using 'eigs': It is well-known that 'eigs' is the preferable method in terms of cost, especially in case of $N_t \ll N_x$. But it is less stable than 'SVD'; compare [34, 42, 43]. The problem is that the W-orthonormality of the POD basis vectors can get lost critically when increasing the number ℓ of POD basis functions. With the aim of improving the variant 'eigs' we test the application of a Gram-Schmidt orthogonalization procedure, refering to it as 'gs'. This means that we apply a Gram-Schmidt algorithm (with respect to the weighted inner product $\langle \cdot, \cdot \rangle_{\mathrm{W}}$) to the POD basis vectors obtained by 'eigs'. Notice that 'gs' can as well be used to add basis vectors to a given POD basis.

Accuracy: We measure the accuracy of the POD Galerkin method with respect to the FE solutions. A time-averaged absolute and relative error between a FE state y^h and a POD state y^ℓ is defined by

$$e^y_{abs}(\ell) := \sum_{j=0}^{N_t} \left\| y^h(t_j) - y^\ell(t_j) \right\|_H, \quad e^y_{rel}(\ell) := \sum_{j=0}^{N_t} \frac{\left\| y^h(t_j) - y^\ell(t_j) \right\|_H}{\left\| y^h(t_j) \right\|_H}. \tag{6.2}$$

The discrete H-norm is stated in equation (4.5). When solving the optimal control problem we use these error quantities with $y^h = \bar{y}^h$ and $y^\ell = \bar{y}^\ell$.

Empirical interpolation: We can use the EIM or the DEIM algorithm to reduce the complexity of evaluating the nonlinear terms in the ROMs. When using DEIM the POD basis computation is done by applying the MATLAB function `eigs` on the matrix $E^T E$ (in Algorithm 4). The corresponding eigenvalues are plotted in a normalized manner analogously to the POD eigenvalues. For comparison we also visualize the respective normalized squared singular values (of the matrix E computed with the MATLAB function `svd`).

State equation: The solution of the full- and reduced-order state equation requires a Newton iteration in each time step. It gets stopped as soon as the discrete H-norm of the residual is smaller or equal to 10^{-10}.

The chapter is divided into two major parts:

6.1 Solution of a Semilinear Heat Equation

First, we turn to the numerical solution of the state equation with $N(y) = y^3$. We apply the FE Galerkin technique and the POD Galerkin method. In this section we follow the lines of [42, Section 6.1] but for a semilinear instead of a linear heat equation.

Example I:
This example is constructed in such a way that we know the exact analytical solution. Thus, we can compare the computed FE solutions to the exact one. Here, we draw a comparison to the accuracy of the FE method when solving a linear heat equation. Turning to the POD Galerkin method we again compare the results for the semilinear equation to those for the linear one. We compare the different variants for computing a POD basis and test the application of a Gram-Schmidt orthogonalization when using 'eigs'. Moreover, we investigate both empirical interpolation methods.

Example II:
In our second test example the initial condition is given by a step function. It is very interesting to see how the ROMs behave with and without empirical interpolation.

6.2 Solution of the Optimal Control Problem

The numerical solution of ($\mathbf{P}$) via ($\hat{\mathbf{P}}$) is centered on a Newton-CG strategy. We apply the FE Galerkin discretization and pursue the POD reduced-order modeling approach with and whithout an update of the POD basis during optimization.

Example III: At the beginning of this example we briefly investigate the FE based solution of a linear-quadratic optimal control problem with $N(y) \equiv 0$. When solving the semilinear problem with $N(y) = y^3$ we start the optimization as close to the optimal control such that a globalization strategy for the (FE based) local inexact Newton method would not be required.

Example IV: This example coincides with Example III except for the desired final state y_d. Here, it is chosen such that negative curvature in the Hessian of the reduced cost functional occurs.

Example V: In contrast to all foregoing considerations the nonlinearity in this example is given by $N(y) = -0.5y^3$. Apart from that, the example setting is the same as in Example III.

6.1 Solution of a Semilinear Heat Equation

6.1.1 Example I

According to Example I in [42, Section 6.1] (which is Run 4.1 in [43]) we first investigate the numerical solution of a semilinear heat equation (SE) for which we know the exact analytical solution. The example is constructed by starting with the desired solution

$$\boxed{y_{ex}(t,\boldsymbol{x}) := t\cos(2\pi x_1)\cos(2\pi x_2) \quad \text{for } t\in[0,T], \boldsymbol{x}=(x_1,x_2)\in\Omega.}$$

Then the parameters and right-hand sides in (SE) are chosen appropriately: For $t\in[0,T]$ and $\boldsymbol{x}=(x_1,x_2)\in\Omega$ we set

$$\boxed{\begin{aligned} c_p := 1, \quad q := 0, \quad u := 0, \quad y_0(\boldsymbol{x}) &\equiv 0, \\ f(t,\boldsymbol{x}) := (1+8\pi^2 t)\cos(2\pi x_1)\cos(2\pi x_2) &+ N\big(y_{ex}(t,\boldsymbol{x})\big). \end{aligned}}$$

If we specify $N(y)\equiv 0$, this is exactly Example I from [42] since the control term in (SE) vanishes ($u=0$). The temperature distribution at the beginning equals zero. This provides the advantage that the inital value needs not to be approximated and delivers no error term. Over time the heat source f together with an isolation along the boundary produces a change in the temperature distribution.

FE method:
First, we are interested in testing if the FE solution to a semilinear heat equation with $N(y)=y^3$ deviates more from its exact solution than the FE solution to a linear equation with $N(y)\equiv 0$.

To measure the accuracy of any FE solution y^h we introduce a time-averaged absolute and relative error by

$$E_{abs}(y^h) := \sum_{j=0}^{N_t} \alpha_j \big\| \underbrace{I(y_{ex}(t_j))}_{\approx y_{ex}(t_j)} - y^h(t_j)\big\|_H, \quad E_{rel}(y^h) := \sum_{j=0}^{N_t} \alpha_j \frac{\big\|I(y_{ex}(t_j)) - y^h(t_j)\big\|_H}{\big\|I(y_{ex}(t_j))\big\|_H},$$

with trapezoidal weights $\alpha_0=\alpha_{N_t}=\frac{\Delta t}{2}$, $\alpha_j=\Delta t$, $j=1,\dots,N_t-1$.

Since the function f is continuous we can evaluate $\mathrm{F}(t)$ at any time $t\in[0,T]$. This allows to set $\mathrm{F}^{(j)}=\mathrm{F}(t_j)=\Big(\int_\Omega f(t_j,\boldsymbol{x})\varphi_i(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}\Big)_{i=1}^{N_x}$, $j=0,\dots,N_t$. The errors listed in Table 6.1 in [42] are based on computing these integrals by Gauss-Legendre quadrature; see [28]. In addition, we test the approximation $\mathrm{F}^{(j)}\approx \mathrm{M}(f(t_j,P_i))_{i=1}^{N_x}$, which is obtained by replacing the term $f(t_j,\boldsymbol{x})$ in the above integral by $I\big(f(t_j,\boldsymbol{x})\big)$. In Table 6.1 we present the absolute and relative errors of several FE solutions for different mesh and time step sizes. The vectors $\mathrm{F}^{(j)}$, $j=0,\dots,N_t$, are computed using Gauss-Legendre integration. The respective results for the second approximation technique are given in Table 6.2. These tables are structured as follows: In the first three rows we keep the time grid and show the effect of dividing the maximal edge length h_{max} of the FE triangulation in half. From the last but one to the last row the time grid gets refined.

Comparison of semilinear and linear: The errors are practically equal. Compared to the

			$N(y) \equiv 0$		$N(y) = y^3$	
h_{max}	Δt	N_x	$E_{abs}(y^h)$	$E_{rel}(y^h)$	$E_{abs}(y^h)$	$E_{rel}(y^h)$
0.1	0.01	177	$2.38 \cdot 10^{-3}$	$1.01 \cdot 10^{-2}$	$2.36 \cdot 10^{-3}$	$1.01 \cdot 10^{-2}$
0.05	0.01	727	$5.35 \cdot 10^{-4}$	$2.29 \cdot 10^{-3}$	$5.37 \cdot 10^{-4}$	$2.29 \cdot 10^{-3}$
0.025	0.01	2810	$1.33 \cdot 10^{-4}$	$5.61 \cdot 10^{-4}$	$1.33 \cdot 10^{-4}$	$5.61 \cdot 10^{-4}$
0.025	0.001	2810	$1.33 \cdot 10^{-4}$	$5.75 \cdot 10^{-4}$	$1.33 \cdot 10^{-4}$	$5.76 \cdot 10^{-4}$

Table 6.1: Example I: FE errors when using Gauss-Legendre quadrature to approximate $\mathrm{F}^{(j)}$, $j = 0, \ldots, N_t$.

			$N(y) \equiv 0$		$N(y) = y^3$	
h_{max}	Δt	N_x	$E_{abs}(y^h)$	$E_{rel}(y^h)$	$E_{abs}(y^h)$	$E_{rel}(y^h)$
0.1	0.01	177	$1.49 \cdot 10^{-2}$	$5.47 \cdot 10^{-2}$	$1.44 \cdot 10^{-2}$	$5.32 \cdot 10^{-2}$
0.05	0.01	727	$2.87 \cdot 10^{-3}$	$1.05 \cdot 10^{-2}$	$2.78 \cdot 10^{-3}$	$1.03 \cdot 10^{-2}$
0.025	0.01	2810	$5.79 \cdot 10^{-4}$	$2.24 \cdot 10^{-3}$	$5.73 \cdot 10^{-4}$	$2.22 \cdot 10^{-3}$
0.025	0.001	2810	$5.79 \cdot 10^{-4}$	$2.24 \cdot 10^{-3}$	$5.73 \cdot 10^{-4}$	$2.22 \cdot 10^{-3}$

Table 6.2: Example I: FE errors when using the approximation $\mathrm{F}^{(j)} \approx \mathrm{M}(f(t_j, P_i))_{i=1}^{N_x}$, $j = 0, \ldots, N_t$.

linear heat equation the nonlinearity in the semilinear equation results in an additional term which needs to be approximated. Here, this does not enlarge the errors.

Approximation of $\mathrm{F}^{(j)}$*,* $j = 0, \ldots, N_t$: The use of Gauss-Legendre quadrature is beneficial. Indeed, for coarse space grids more than for fine grids as one expects.

Order of the error: The initial function in (SE) needs not to be approximated when applying the FE Galerkin technique. Hence, we can name the following convergence result for the FE method in combination with the implicit Euler method. It can be looked up in [36]. In [12] it is nicely proven for (SE) with $N(y) \equiv 0$ and with Dirichlet boundary conditions ($q = 0$). It holds

$$\max \{ \|y_{ex}(t_j) - y^h(t_j)\|_H \mid j = 0, \ldots, N_t \} = \mathcal{O}(h_{max}^2 + \Delta t).$$

Taking the sum over all time steps gives

$$E_{abs}(y^h) = \mathcal{O}(h_{max}^2 + \Delta t). \tag{6.3}$$

This motivates the choice $\Delta t = h_{max}^2$ for balancing the error contributions from space and time discretization. The values in the first row of the above tables satisfy $\Delta t = 0.01 = h_{max}^2$. For all other rows the time step size Δt is larger than the squared maximal edge length h_{max}^2. Except for the first row in Table 6.2 the absolute error $E_{abs}(y^h)$ is smaller than the (dominating) value Δt. Refining the time grid from the last but one to the last row in the above tables does not reduce the errors. Actually, for $\Delta t = 0.01$ (row 3) we already have

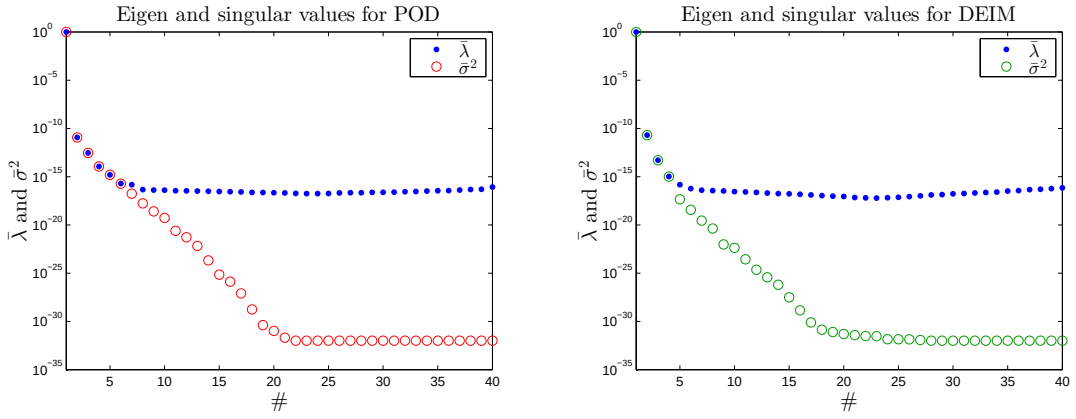


Figure 6.1: Example I, $N(y) = y^3$, $X = H$: Decay of normalized singular values and eigenvalues for the POD basis (left) and for the DEIM basis (right).

$E_{abs}(y^h) < 0.001$. Moreover, our results confirm spatial order two for the given equations: Bisecting the maximal edge length h_{max} in the FE triangulation (roughly) quarters the absolute error $E_{abs}(y^h)$.

POD method:
For the application of the POD Galerkin method we keep fixed the discretization parameters

$$\boxed{h_{max} = 0.025, \quad \Delta t = 0.01.}$$

This results in the (full) FE dimension $N_x = 2810$. The vectors $\mathrm{F}^{(j)}$, $j = 0, \ldots, N_t$, are computed using Gauss-Legendre quadrature. The time-averaged absolute error of the FE solutions for $N(y) \equiv 0$ and $N(y) = y^3$ equals $1.3 \cdot 10^{-4}$; see Table 6.1.

As described in Chapter 4, a POD Galerkin approximation consists in replacing the FE basis functions by a number $\ell \ll N_x$ of POD basis functions. The latter functions (their FE coefficients contained in the POD basis matrix Ψ) are computed from the snapshots $\mathrm{y}^{(j)}$, $j = 0, \ldots, N_t$, of the given FE solution. The corresponding nonlinear snapshots $N(\mathrm{y}^{(j)})$, $j = 0, \ldots, N_t$, are utilized to build up the EIM and DEIM bases. Recall that the nonlinear snapshots get computed anyway while solving the full-order state equation.

For $N(y) \equiv 0$ the considered example is identical to Example I from [42], as mentioned above. Whatever nonlinearity we choose, the exact solution is by construction always the same. Hence, the basic statements from [42] can be taken over and the basic characteristics of the POD ROMs are determined independently of the function N. For a good understanding we briefly quote the essence and refer the reader to [42] for a detailed consideration of the linear equation. In this example the solution space of (SE) is one-dimensional: At any time t the exact solution is the t-th multiple of $\cos(2\pi x_1)\cos(2\pi x_2)$. The snapshot matrix $[\mathrm{y}^{(0)}, \ldots, \mathrm{y}^{(N_t)}] \in \mathbb{R}^{N_x \times (N_t+1)}$ represents the solution space. Due to numerical inaccuracy this matrix has a rank greater than one. But the POD eigenvalues computed from these snapshots decay extremely rapidly. Of course, the same holds for the DEIM eigenvalues.

We present the results for the H-norm implementation in detail. The choice $X = H$ yields the weighting matrix $\mathrm{W} = \mathrm{M}$. First of all, we visualize the decay of the POD eigenvalues for $N(y) = y^3$ in the left plot in Figure 6.1. The normalized squared singular values

	$N(y) \equiv 0$			$N(y) = y^3$		
ℓ	$e^y_{abs}(\ell)$	$e^y_{rel}(\ell)$	λ_ℓ	$e^y_{abs}(\ell)$	$e^y_{rel}(\ell)$	λ_ℓ
1	$1.8 \cdot 10^{-07}$	$2.5 \cdot 10^{-06}$	$8.3 \cdot 10^{-02}$	$8.7 \cdot 10^{-07}$	$5.6 \cdot 10^{-06}$	$8.3 \cdot 10^{-02}$
2	$2.1 \cdot 10^{-08}$	$6.1 \cdot 10^{-07}$	$4.5 \cdot 10^{-14}$	$1.4 \cdot 10^{-07}$	$2.1 \cdot 10^{-06}$	$9.7 \cdot 10^{-13}$
3	$7.7 \cdot 10^{-09}$	$2.5 \cdot 10^{-07}$	$4.9 \cdot 10^{-16}$	$5.0 \cdot 10^{-08}$	$8.3 \cdot 10^{-07}$	$2.4 \cdot 10^{-14}$
4	$1.7 \cdot 10^{-09}$	$5.3 \cdot 10^{-08}$	$< 10^{-16}$	$1.5 \cdot 10^{-08}$	$4.2 \cdot 10^{-07}$	$9.4 \cdot 10^{-16}$
7	$1.1 \cdot 10^{-10}$	$2.2 \cdot 10^{-09}$	$< 10^{-16}$	$8.8 \cdot 10^{-10}$	$2.3 \cdot 10^{-08}$	$< 10^{-16}$
10	$1.3 \cdot 10^{-13}$	$3.9 \cdot 10^{-12}$	$< 10^{-16}$	$5.7 \cdot 10^{-11}$	$1.8 \cdot 10^{-09}$	$< 10^{-16}$

Table 6.3: Example I, $N(y) = y^3$, $X = H$: Absolute & relative errors and λ_ℓ for different ℓ for the linear ($N(y) \equiv 0$) and semilinear ($N(y) = y^3$) heat equation using 'eigs'.

$\bar{\sigma}_i^2$ obtained by 'SVD' and the normalized eigenvalues $\bar{\lambda}_i$ computed by 'eigs' are shown. In the beginning the values $\bar{\sigma}_i^2$ and $\bar{\lambda}_i$ are the same. Since the singular values are squared for comparison they keep decreasing while the eigenvalues computed by 'eigs' stagnate at the order of machine precision. The POD eigenvalues resulting from the linear equation with $N(y) \equiv 0$ decay even faster; compare the fourth and seventh column of Table 6.3.

Comparison of semilinear and linear case: We use the variant 'eigs' for the POD basis computation. The errors when solving the ROMs of different dimensions ℓ for the semilinear and the linear heat equation are stated in Table 6.3. For both equations the errors are considerably smaller than the error of the respective FE solution, even for $\ell = 1$. This is attributed to the theoretical dimension one of the solution space. The errors for the linear equation are smaller than those for the semilinear one.

Gram-Schmidt orthogonalization: We investigate the application of a Gram-Schmidt algorithm ('gs') to improve the variant 'eigs' in terms of W-orthonormality of the POD basis vectors.
Denote by I_ℓ the $\ell \times \ell$ identity matrix. We write $\Psi_{:,1:\ell}$ for the $N_x \times \ell$ submatrix (first ℓ columns) of the POD basis matrix Ψ. Analytically, the POD basis vectors are orthonormal with respect to the weighted inner product $\langle \cdot, \cdot \rangle_{\mathrm{W}}$, i.e. the equality

$$\Psi_{:,1:\ell}^T \mathrm{W} \Psi_{:,1:\ell} - I_\ell = 0 \quad \in \mathbb{R}^{\ell \times \ell}$$

is satisfied for any POD basis rank ℓ. To measure W-orthonormality of the numerically computed POD basis vectors we estimate the spectral norm of the matrix $\Psi_{:,1:\ell}^T \mathrm{W} \Psi_{:,1:\ell} - I_\ell$. This is done with the MATLAB routine `normest`.
We compare the three variants 'eigs', 'eigs'&'gs' and 'SVD'. The results for the linear and the semilinear heat equation for the numbers $\ell = 1, 4, 10$ are given in Table 6.4. It can be seen that 'gs' improves the W-orthonormality of the POD basis vectors obtained by 'eigs' in a very satisfactory manner.
This can be confirmed by the absolute errors of the POD solutions computed by 'eigs'&'gs'. In Table 6.5 we compare these errors to those obtained by 'gs' and 'SVD' for different numbers ℓ. At the beginning, the errors are equal for the different variants. In fact, this holds at least as long as the smallest POD eigenvalue λ_ℓ is still above machine precision.

	$N(y) \equiv 0$			$N(y) = y^3$		
ℓ	'eigs'	'eigs'&'gs'	'SVD'	'eigs'	'eigs'&'gs'	'SVD'
1	$8.9 \cdot 10^{-16}$	$2.2 \cdot 10^{-16}$	$1.8 \cdot 10^{-15}$	$6.7 \cdot 10^{-16}$	$2.2 \cdot 10^{-16}$	$2.2 \cdot 10^{-16}$
4	$9.7 \cdot 10^{-03}$	$4.6 \cdot 10^{-16}$	$2.9 \cdot 10^{-15}$	$4.8 \cdot 10^{-04}$	$8.9 \cdot 10^{-16}$	$3.1 \cdot 10^{-15}$
10	$1.0 \cdot 10^{+00}$	$9.8 \cdot 10^{-08}$	$4.2 \cdot 10^{-15}$	$1.0 \cdot 10^{+00}$	$1.5 \cdot 10^{-13}$	$3.6 \cdot 10^{-15}$

Table 6.4: Example I, $X = H$: Spectral norm $\left\|\Psi_{:,1:\ell}^T \mathrm{W} \Psi_{:,1:\ell} - I_\ell\right\|_2$ for different ℓ for the linear ($N(y) \equiv 0$) and semilinear ($N(y) = y^3$) heat equation using 'eigs', 'eigs'&'gs' and 'SVD'

	$N(y) \equiv 0$			$N(y) = y^3$		
ℓ	'eigs'	'eigs'&'gs'	'SVD'	'eigs'	'eigs'&'gs'	'SVD'
1	$1.8 \cdot 10^{-07}$	$1.8 \cdot 10^{-07}$	$1.8 \cdot 10^{-07}$	$8.7 \cdot 10^{-07}$	$8.7 \cdot 10^{-07}$	$8.7 \cdot 10^{-07}$
4	$1.7 \cdot 10^{-09}$	$1.7 \cdot 10^{-09}$	$1.7 \cdot 10^{-09}$	$1.5 \cdot 10^{-08}$	$1.5 \cdot 10^{-08}$	$1.5 \cdot 10^{-08}$
10	$1.3 \cdot 10^{-13}$	$1.3 \cdot 10^{-13}$	$9.7 \cdot 10^{-14}$	$5.7 \cdot 10^{-11}$	$4.6 \cdot 10^{-11}$	$3.0 \cdot 10^{-11}$
12	$2.6 \cdot 10^{-12}$	$1.0 \cdot 10^{-14}$	$2.9 \cdot 10^{-15}$	$3.1 \cdot 10^{-11}$	$2.2 \cdot 10^{-11}$	$1.9 \cdot 10^{-11}$
15	$7.0 \cdot 10^{-12}$	$1.9 \cdot 10^{-15}$	$9.6 \cdot 10^{-16}$	$1.7 \cdot 10^{-11}$	$1.7 \cdot 10^{-11}$	$1.7 \cdot 10^{-11}$
20	$7.1 \cdot 10^{-12}$	$1.1 \cdot 10^{-15}$	$6.5 \cdot 10^{-16}$	$4.6 \cdot 10^{-11}$	$1.7 \cdot 10^{-11}$	$1.7 \cdot 10^{-11}$

Table 6.5: Example I, $X = H$: Absolute errors $e_{abs}^y(\ell)$ for different ℓ for the linear ($N(y) \equiv 0$) and semilinear ($N(y) = y^3$) heat equation using 'eigs', 'eigs'&'gs' and 'SVD'.

ℓ	ROM	ROM-EIM	ROM-DEIM
1	$8.7138 \cdot 10^{-7}$	$8.7138 \cdot 10^{-7}$	$8.7138 \cdot 10^{-7}$
2	$1.4397 \cdot 10^{-7}$	$1.4397 \cdot 10^{-7}$	$1.4397 \cdot 10^{-7}$
3	$4.9993 \cdot 10^{-8}$	$4.9993 \cdot 10^{-8}$	$4.9993 \cdot 10^{-8}$
4	$1.5237 \cdot 10^{-8}$	$1.5208 \cdot 10^{-8}$	$1.5207 \cdot 10^{-8}$

Table 6.6: Example I, $N(y) = y^3$, $X = H$: Absolute errors $e_{abs}^y(\ell)$ for $\ell = 1, \ldots, 4$ when solving the ROM with and without EIM or DEIM for $\ell^{EIM} = 4$ using 'eigs'&'gs'.

Recall that the eigenvalues of interest are given in Table 6.3. Once the eigenvalues are in the range of machine precision or below, the variants differ as follows:

- *Linear*: For 'eigs' the absolute errors oscillate around 10^{-13}, 10^{-12} for numbers $\ell \geq 10$. Using 'eigs'&'gs' the errors keep decreasing down to 10^{-15} and they are just a bit larger than those obtained by the variant 'SVD'.
- *Semilinear*: For all three variants the errors stagnate around 10^{-11}. Taking just 'eigs' gives slight oscillations but 'eigs'&'gs' is absolutely as good as 'SVD'. The reason why the errors stagnate and do not decrease as far as for the linear equation has to do with the used and practicable accuracy of the full- and reduced-order solvers for the semilinear equation. Remember that the Newton iteration in each time step is terminated when the discrete H-norm of the residual is smaller or equal to 10^{-10}.

Even for $\ell = 35$ our 'gs' algorithm applied to the POD basis matrix obtained by 'eigs' in Example I needs less than 0.05 seconds CPU time. Hence, we advise to use the variant 'eigs'&'gs' for the computation of a POD basis.

So far, we have excluded the question if it really makes sense to further increase the number ℓ of POD basis functions once the eigenvalues are close to machine precision. Since we compare a POD solution to the respective FE solution we are actually satisfied if the error between these two is smaller than the error that is made by the FE method itself. Nonetheless, it is interesting to see up to which level of accuracy with respect to the FE solution the POD solutions can be brought by increasing the number ℓ of POD basis functions. When the POD eigenvalues are in the range of machine precision this means that the essential information in the FE snapshot matrix is already captured by the previous POD basis functions. Let us illustrate this with the introduced energy term. For $\ell = 1$ we have $1 - \mathcal{E}(\ell) = 1.2 \cdot 10^{-11}$ for the semilinear heat equation and $1 - \mathcal{E}(\ell) = 5.5 \cdot 10^{-13}$ for the linear equation. Thus, we can capture almost all energy contained in the snapshot ensemble by only one single POD basis function.

Empirical interpolation methods: For the EIM algorithm we choose $\epsilon^{EIM} = 10^{-8}$ resulting in $\ell^{EIM} = 4$ interpolation points. This value is also chosen for DEIM in order to compare the results. The DEIM eigenvalues are shown in the right plot in Figure 6.1. Their decay confirms adequateness of using $\ell^{EIM} = 4$ interpolation points for DEIM. In Table 6.6 we compare the absolute errors when solving the ROM for the semilinear heat equation with and without EIM or DEIM for $\ell = 1, \ldots, 4$ and $\ell^{EIM} = 4$. We observe that EIM and DEIM perform both very well and that there is no difference in accuracy when comparing the two methods.

Comparison to $X = V$: The V-norm implementation yields the following differences: The weighting matrix $\mathrm{W} = \mathrm{M} + \mathrm{A}$ leads to greater eigenvalues and a slower decay compared to $\mathrm{W} = \mathrm{M}$. In case of the linear equation the effect on the accuracy of the POD solutions is small. The errors are nearly the same as for $X = H$. In Table 6.7 we name the absolute errors for the linear and the semilinear equation analogously to Table 6.5, i.e. for the variants 'eigs', 'eigs'&'gs' and 'SVD'. For the semilinear equation we observe that the absolute error reaches values in the range of 10^{-12} for all three variants. This is about one decimal power smaller than for the H-norm implementation. In case of 'eigs', the

	$N(y) \equiv 0$			$N(y) = y^3$		
ℓ	'eigs'	'eigs'&'gs'	'SVD'	'eigs'	'eigs'&'gs'	'SVD'
1	$1.8 \cdot 10^{-07}$	$1.8 \cdot 10^{-07}$	$1.8 \cdot 10^{-07}$	$8.7 \cdot 10^{-07}$	$8.7 \cdot 10^{-07}$	$8.7 \cdot 10^{-07}$
4	$2.2 \cdot 10^{-09}$	$2.2 \cdot 10^{-09}$	$1.7 \cdot 10^{-09}$	$3.4 \cdot 10^{-08}$	$3.4 \cdot 10^{-08}$	$3.4 \cdot 10^{-08}$
10	$2.1 \cdot 10^{-13}$	$1.4 \cdot 10^{-13}$	$1.0 \cdot 10^{-13}$	$9.3 \cdot 10^{-11}$	$3.2 \cdot 10^{-11}$	$2.1 \cdot 10^{-11}$
12	$7.9 \cdot 10^{-12}$	$1.0 \cdot 10^{-14}$	$2.9 \cdot 10^{-15}$	$3.9 \cdot 10^{-12}$	$5.7 \cdot 10^{-12}$	$5.2 \cdot 10^{-12}$
15	$1.9 \cdot 10^{-11}$	$4.4 \cdot 10^{-15}$	$5.6 \cdot 10^{-16}$	$1.7 \cdot 10^{-12}$	$1.2 \cdot 10^{-12}$	$1.2 \cdot 10^{-12}$
20	$1.8 \cdot 10^{-11}$	$1.4 \cdot 10^{-15}$	$5.6 \cdot 10^{-16}$	$3.4 \cdot 10^{-09}$	$1.2 \cdot 10^{-12}$	$1.2 \cdot 10^{-12}$

Table 6.7: Example I, $N(y) = y^3$, $X = V$: Absolute errors $e^y_{abs}(\ell)$ for different ℓ for the linear ($N(y) \equiv 0$) and semilinear ($N(y) = y^3$) heat equation using 'eigs', 'eigs'&'gs' and 'SVD'.

	FE	POD	EIM	DEIM	ROM	ROM-EIM	ROM-DEIM
$X = H$	2.83	0.19	0.02	0.06	0.21	0.07	0.07
$X = V$	–	0.19	–	–	0.15	0.05	0.06

Table 6.8: Example I, $N(y) = y^3$: CPU times of the FE and POD method measured in seconds for $\ell = \ell^{EIM} = 4$ using 'eigs'&'gs'.

oscillations in the error are more pronounced compared to Table 6.5 and for $\ell = 20$ it even increases back to 10^{-9}.

Computational efficiency: The full- and reduced-order solvers for the linear equation have to solve only one linear system of equations per time step. But the solvers for the semilinear equation perform a Newton method in each time iteration and the nonlinear term and its Jacobian have to be evaluated (N_x- or ℓ^{EIM}-dimensional). We observed that, on average, the reduced-order solver generally needs more Newton steps to satisfy the termination criterion than the full-order solver. For this example the iterations inside the reduced-order solver are not as many for the V-norm implementation as for the use of $X = H$. In fact, for $X = V$ the number of iterations carried out by the reduced-order solver is almost equal to that of the full-order solver. This results in a smaller CPU time when using $X = V$ compared to $X = H$. The performance with respect to computational time of the POD method with and without EIM or DEIM for $X = H$ and $X = V$ compared to the FE method is summarized in Table 6.8. For $X = H$ the required CPU time for the solution of the ROM is about 13 times smaller than the needed CPU time for the solution of the fine model. The use of EIM or DEIM reduces the CPU time again by a factor of 3. For $X = V$ the solution of the ROM is almost 19 times faster than the solution of the fine model. When adding EIM the required CPU time can even be reduced to amazing 0.05 seconds. For comparison we also state the CPU times associated with the solution of the linear heat equation: It takes about 0.70 seconds to solve the fine model and about 0.01 seconds to compute the solution to the ROM with $\ell = 4$ (for both $X = V$ and $X = H$).

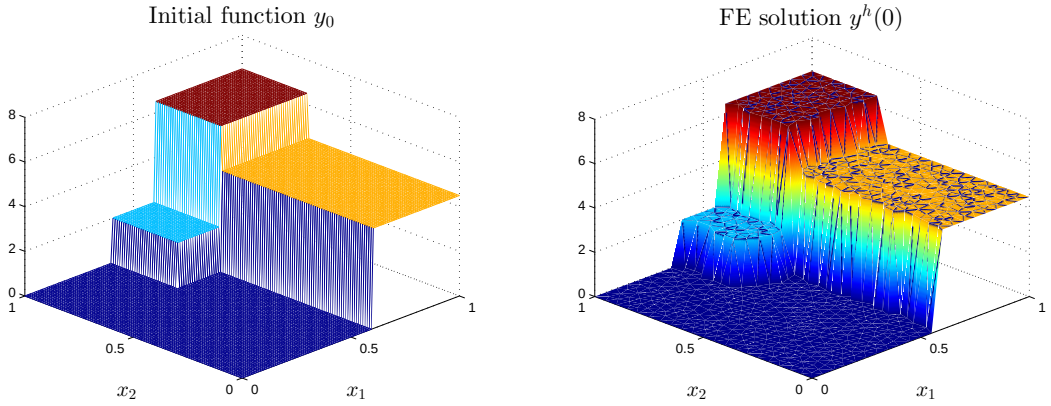


Figure 6.2: Example II: Exact initial function y_0 (left) and FE solution $y^h(0)$ (right).

6.1.2 Example II

Our second test example is more challenging for the POD reduced-order modeling approach and for the empirical interpolation methods than the first one. We consider a discontinuous initial value function y_0 and an external heating source is applied. Each boundary part of the unit square is heated or cooled separately but constantly. For $t \in [0, T]$ and $\boldsymbol{x} = (x_1, x_2) \in \Omega$ we specify

$$f(t, \boldsymbol{x}) \equiv 0, \quad N(y) := y^3, \quad \mathsf{k} := 4, \quad q := 0.01, \quad c_p := 10,$$

$$u(t) \equiv \begin{pmatrix} 0.5 \\ -4 \\ -0.5 \\ 4 \end{pmatrix}, \quad y_0(\boldsymbol{x}) := \begin{cases} 2 & \text{if } x_1 \in [0.4, 0.6) \text{ and } x_2 \in [0.7, 1], \\ 4.5 & \text{if } x_1 \in [0.6, 1] \text{ and } x_2 \in [0, 0.7), \\ 6 & \text{if } x_1 \in [0.6, 1] \text{ and } x_2 \in [0.7, 1], \\ 0 & \text{else,} \end{cases}.$$

The discretization parameters are

$$h_{max} := 0.05, \quad \Delta t := 0.0025.$$

We obtain the number $N_x = 727$ of FE nodes. The mesh and time grid sizes differ from those of Example I. The exact solution here is unknown. Therefore, we can not compute the error of any FE solution. This is why we choose the time step size Δt and the maximal edge length h_{max} of the FE triangulation such that $\Delta t = h_{max}^2$ is satisfied.

The left plot in Figure 6.2 shows the step function y_0. Contrary to Example I, this initial value function needs to be approximated when applying the FE Galerkin technique. In this thesis we use the approximation $y^h(0) = Iy_0 \approx y_0$, i.e. the (simple) interpolation in V^h. Alternatively, a Ritz projection could be used (leads to the solution of a Poisson problem); see [4, 12]. The interpolant $Iy_0 \in V^h$ is visualized in the right plot in Figure 6.2. Of course, an error in the initial condition is also made when applying the POD Galerkin method. Similar to FE error estimates, the error at the initial time influences the error of the POD solution at later time points; an error estimate can be found in [29, Theorem 10].

We compute the FE solution y^h. In Figure 6.3 we present the function $y^h(t) \in V^h$ for several time steps $t = t_j$. As it can be seen, the discontinuity at the initial time $t_0 = 0$

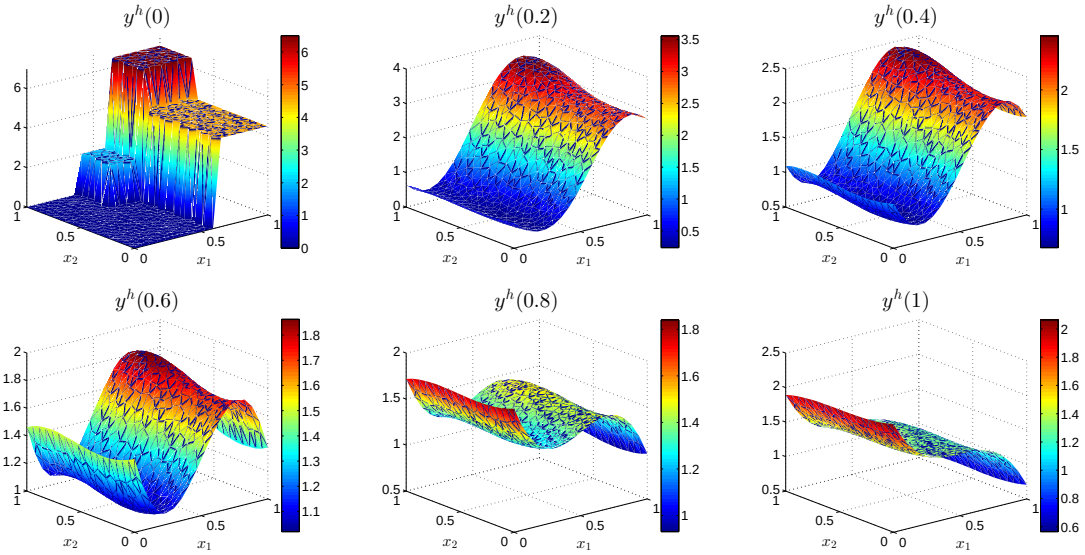


Figure 6.3: Example II: FE solution $y^h(t)$ at six different time steps $t = t_j$.

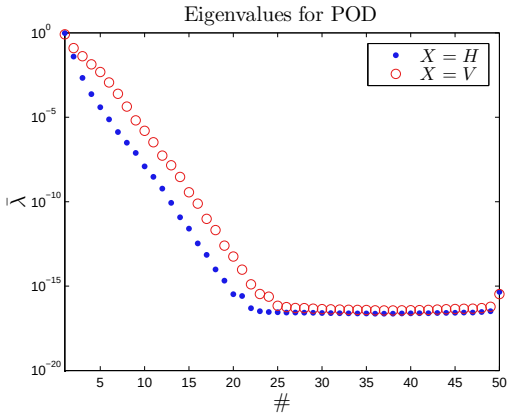


Figure 6.4: Example II: Decay of normalized eigenvalues for the POD basis using $X = H$ and $X = V$.

disappears for $t_j > 0$: The solution becomes smooth. This smoothing property (of any sharp features) is characteristic for the heat equation; see [4, p. 127].

We turn to the POD method. The snapshots of the FE state y^h are utilized to set up the POD basis and the associated nonlinear snapshots are taken for the empirical interpolation methods. Due to the results in Example I we use the variant 'eigs'&'gs' for the POD basis computation. The observations made in the first example motivate a comparison between the H- and the V-norm implementation.

In Figure 6.4 we show the decay of the first 50 normalized POD eigenvalues computed with $X = H$ and with $X = V$. Remember that the space $X = H$ yields the weighting matrix $\mathrm{W} = \mathrm{M}$ and that $X = V$ results in $\mathrm{W} = \mathrm{M} + \mathrm{A}$. As mentioned in Example I, the eigenvalues obtained by the weighting matrix $\mathrm{W} = \mathrm{M} + \mathrm{A}$ are larger and decay slower than those computed with $\mathrm{W} = \mathrm{M}$. In case of $X = V$ the eigenvalues stop decreasing as from $\ell = 25$. For $X = H$ the stagnation starts slightly earlier. Once the eigenvalues

	$X = H$			$X = V$		
ℓ	$e^y_{abs}(\ell)$	$e^y_{rel}(\ell)$	λ_ℓ	$e^y_{abs}(\ell)$	$e^y_{rel}(\ell)$	λ_ℓ
1	$4.9 \cdot 10^{-01}$	$2.9 \cdot 10^{-01}$	$2.9 \cdot 10^{+00}$	$5.2 \cdot 10^{-01}$	$3.3 \cdot 10^{-01}$	$1.4 \cdot 10^{+01}$
2	$7.6 \cdot 10^{-02}$	$4.2 \cdot 10^{-02}$	$2.8 \cdot 10^{-01}$	$1.1 \cdot 10^{-01}$	$6.9 \cdot 10^{-02}$	$2.5 \cdot 10^{+00}$
6	$2.6 \cdot 10^{-03}$	$1.4 \cdot 10^{-03}$	$2.8 \cdot 10^{-05}$	$3.7 \cdot 10^{-03}$	$2.2 \cdot 10^{-03}$	$1.9 \cdot 10^{-02}$
7	$\mathbf{1.2 \cdot 10^{-03}}$	$6.3 \cdot 10^{-04}$	$5.0 \cdot 10^{-06}$	$\mathbf{1.6 \cdot 10^{-03}}$	$8.3 \cdot 10^{-04}$	$4.6 \cdot 10^{-03}$
8	$\mathbf{7.1 \cdot 10^{-04}}$	$4.0 \cdot 10^{-04}$	$1.5 \cdot 10^{-06}$	$1.1 \cdot 10^{-03}$	$6.7 \cdot 10^{-04}$	$8.2 \cdot 10^{-04}$
12	$2.4 \cdot 10^{-05}$	$1.3 \cdot 10^{-05}$	$3.1 \cdot 10^{-09}$	$3.2 \cdot 10^{-05}$	$1.9 \cdot 10^{-05}$	$1.7 \cdot 10^{-06}$
14	$4.5 \cdot 10^{-06}$	$2.5 \cdot 10^{-06}$	$8.6 \cdot 10^{-11}$	$6.6 \cdot 10^{-06}$	$4.0 \cdot 10^{-06}$	$6.9 \cdot 10^{-08}$
17	$3.1 \cdot 10^{-07}$	$1.8 \cdot 10^{-07}$	$4.9 \cdot 10^{-13}$	$3.6 \cdot 10^{-07}$	$2.1 \cdot 10^{-07}$	$4.0 \cdot 10^{-10}$
19	$5.6 \cdot 10^{-08}$	$3.2 \cdot 10^{-08}$	$1.2 \cdot 10^{-14}$	$6.6 \cdot 10^{-08}$	$3.9 \cdot 10^{-08}$	$1.3 \cdot 10^{-11}$
22	$8.8 \cdot 10^{-09}$	$5.0 \cdot 10^{-09}$	$4.0 \cdot 10^{-16}$	$4.5 \cdot 10^{-09}$	$2.7 \cdot 10^{-09}$	$5.2 \cdot 10^{-14}$
25	$8.6 \cdot 10^{-10}$	$5.4 \cdot 10^{-10}$	$< 10^{-16}$	$5.4 \cdot 10^{-10}$	$3.4 \cdot 10^{-10}$	$1.6 \cdot 10^{-15}$
30	$1.4 \cdot 10^{-10}$	$9.3 \cdot 10^{-11}$	$< 10^{-16}$	$1.4 \cdot 10^{-10}$	$9.3 \cdot 10^{-11}$	$7.6 \cdot 10^{-16}$
35	$1.4 \cdot 10^{-10}$	$9.3 \cdot 10^{-11}$	$1.2 \cdot 10^{-15}$	$1.4 \cdot 10^{-10}$	$9.3 \cdot 10^{-11}$	$5.0 \cdot 10^{-15}$

Table 6.9: Example II: Absolute & relative errors and λ_ℓ for different ℓ for $X = H$ and $X = V$.

stagnate, which occurs in the range of machine precision, the accuracy of the POD solutions can usually no longer be significantly improved by further increasing the POD basis rank ℓ.

This can be confirmed by the values from Table 6.9. In this table we name the absolute and relative errors of the POD solutions for different numbers ℓ for $X = H$ and $X = V$. The associated (non-normalized) eigenvalues λ_ℓ are also listed. The absolute error of the FE solution ($E_{abs}(y^h)$) is assumed to be $2.5 \cdot 10^{-3} = \Delta t = h^2_{max}$. For both spaces $X = H$ and $X = V$ the absolute error $e^y_{abs}(\ell)$ drops beneath $2.5 \cdot 10^{-3}$ when using $\ell = 7$ POD basis functions. In contrast to Example I, the errors for $X = V$ do not become smaller than those for $X = H$. In both cases the absolute errors stagnate at $1.4 \cdot 10^{-10}$. On the one hand, for $\ell \leq 19$ the errors for $X = V$ are slightly larger than those for $X = H$. On the other hand, for $\ell \leq 12$ the reduced-order solver using $X = V$ needs less Newton iterations compared to $X = H$. But the difference is so small that the resulting CPU times are merely distinguishable. Here, neither the V-norm implementation nor the H-norm implementation is superior. The two spaces lead to comparable results.

As already observed, the POD eigenvalues are larger and their decay is slower when using $X = V$ compared to $X = H$. This gives smaller energy ratios $\mathcal{E}(\ell)$. In Table 6.10 we state the values $1 - \mathcal{E}(\ell)$ of unmodelled energy for different numbers ℓ for $X = H$ and $X = V$. In addition, we name the values of unmodelled energy for the DEIM eigenvalues (last row). For $\ell = 7$ the errors of the POD solutions computed with $X = H$ and $X = V$ are nearly equal and assumed to undermatch the error of the FE solution. But the value $1 - \mathcal{E}(7)$ of unmodelled energy when using $X = V$ is about two powers larger compared to $X = H$. This shows that the quotient $\mathcal{E}(\ell)$, describing the percentage of captured energy by the first ℓ POD basis functions, must be interpreted in consideration of the used space X.

	ℓ or ℓ^{EIM}				
	1	2	6	7	8
$1-\mathcal{E}(\ell)$, $X=H$	$9.0\cdot10^{-2}$	$2.8\cdot10^{-3}$	$2.2\cdot10^{-6}$	$4.0\cdot10^{-7}$	$1.3\cdot10^{-7}$
$1-\mathcal{E}(\ell)$, $X=V$	$2.4\cdot10^{-1}$	$9.9\cdot10^{-2}$	$2.9\cdot10^{-4}$	$5.1\cdot10^{-5}$	$1.0\cdot10^{-5}$
$1-\mathcal{E}(\ell^{EIM})$ for DEIM	$2.4\cdot10^{-2}$	$9.7\cdot10^{-3}$	$4.5\cdot10^{-5}$	$1.9\cdot10^{-6}$	$3.4\cdot10^{-7}$

Table 6.10: Example II: Unmodelled energy $1-\mathcal{E}(\ell)$ for different ℓ for POD using $X=H$ and $X=V$ and for DEIM.

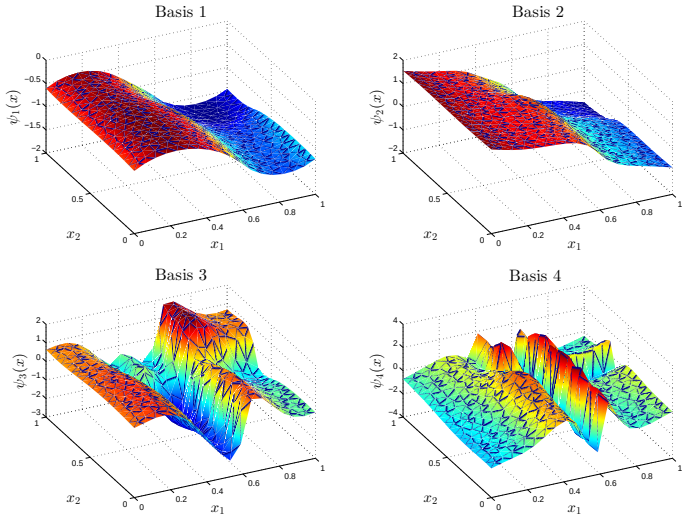


Figure 6.5: Example II: First four POD basis functions computed with $X=H$.

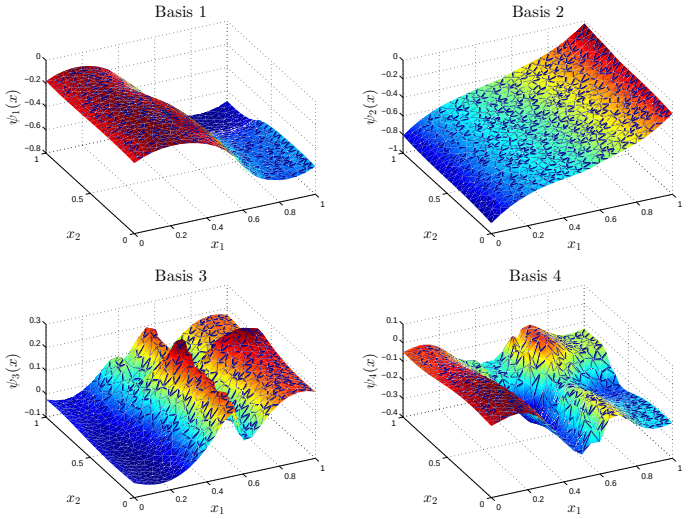


Figure 6.6: Example II: First four POD basis functions computed with $X=V$.

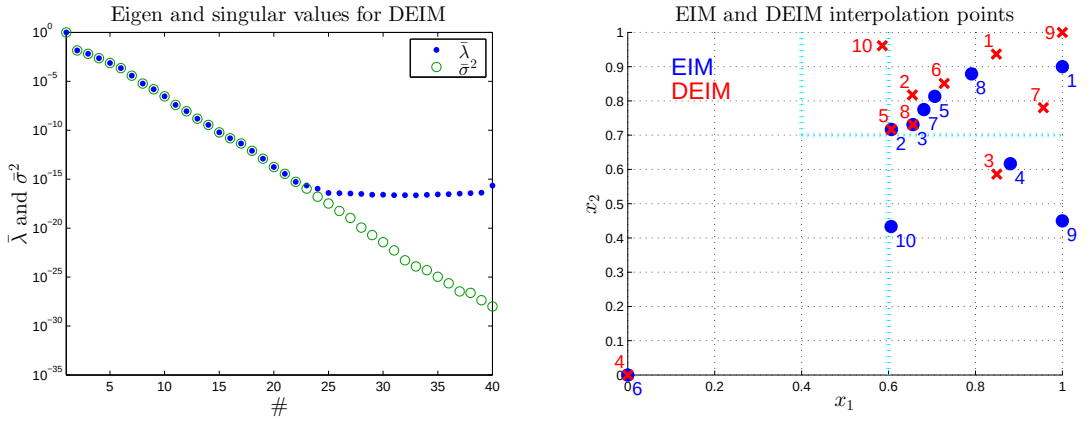


Figure 6.7: Example II: Decay of normalized eigenvalues and squared singular values for the DEIM basis (left) and the first ten EIM and DEIM interpolation points (right).

We present the shape of the first four POD basis functions computed with $X = H$ and with $X = V$. Notice that the algebraic sign of the POD basis functions is not uniquely determined. Figure 6.5 shows the first four POD bases obtained for $X = H$. The first basis function represents the dynamics of the FE solution at positive times. In the second basis function we can make out slight different steps located as in the initial condition but the basic shape is still determined by the FE solution at positive times. The shape of the third and fourth basis function is clearly related to the inital condition: In the third one we recognize the different steps and the activity of the fourth one is concentrated on the step at $x_1 = 0.6$. The first four POD basis functions computed from $X = V$ can be seen in Figure 6.6. The shape of the first basis function is very similar to that in Figure 6.5. The second basis function here is smoother. In the shape of the third and fourth basis function we again recognize the jump discontinuities of the initial condition.

Empirical interpolation methods: For the investigation of the empirical interpolation methods we keep the space $X = H$ for the POD method fixed. We are especially interested in testing how good the POD method with and without EIM or DEIM can reproduce the FE solution at initial and at final time. Unless otherwise specified, the POD basis rank equals $\ell = 8$. This yields the absolute error $e^y_{abs}(8) = 7.1 \cdot 10^{-4}$ when solving the ROM (without EIM or DEIM).

The left plot in Figure 6.7 illustrates the decay of the first 40 normalized eigenvalues and squared singular values for DEIM. First, with Example I in mind we used the tolerance $\epsilon^{EIM} = 10^{-8}$ to build up the EIM basis. This gave $\ell^{EIM} = 30$ interpolation points. But the decay rate of the eigenvalues for DEIM let us doubt if the obtained number is not too large. And actually, many less interpolation points suffice to obtain satisfactory approximations.

The relative error in H-norm over time of several POD-EIM and POD-DEIM solutions with respect to the FE solution can be seen in Figure 6.8. The upper plots show the error curves of the POD-DEIM solutions while the lower plots present the error curves of the POD-EIM solutions. The black curve in the plots is the relative error curve of the POD solution. For both empirical interpolation methods, even for $\ell^{EIM} = 1$, the relative error

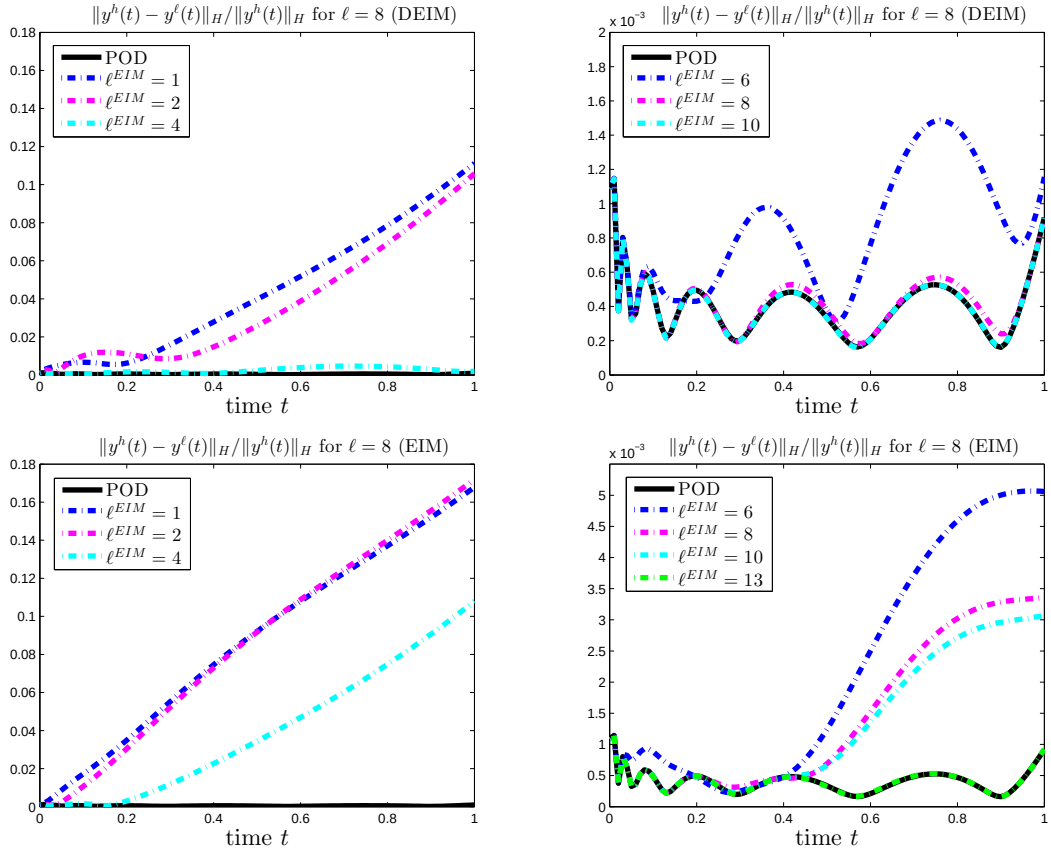


Figure 6.8: Example II, $X = H$: Relative errors over time of several POD-EIM and POD-DEIM solutions with $\ell = 8$.

at initial time $t = 0$ differs less than 10^{-14} from the relative error of the POD solution. Apparently, one interpolation point is only needed for an accurate POD-EIM or POD-DEIM approximation of the FE initial function. Over time the relative error increases up to about 10^{-1} for small numbers ℓ^{EIM}, as the left plots show. This increase is larger for EIM than for DEIM. For DEIM with $\ell^{EIM} = 10$ interpolation points the relative error curve is optically identical to that of the POD solution. For EIM one has to take at least $\ell^{EIM} = 13$ interpolation points so that the relative error curves are no longer visibly distinguishable. Comparing the results for DEIM and the values of unmodelled energy named in Table 6.10 we make an interesting observation: The POD value $1 - \mathcal{E}(8)$ is smaller than $1 - \mathcal{E}(\ell^{EIM})$ for $\ell^{EIM} \leq 9$ while the latter value goes below the first for $\ell^{EIM} = 10$. Of course, this should not be over-interpreted.

In the right plot in Figure 6.7 we compare the distribution of the first 10 interpolation points selected by EIM and DEIM. The methods perform very similar in their selection and it can be seen that the selected points cluster in the right upper part of the domain. The initial condition has its largest step in this corner and regarding Figure 6.3 we observe that the largest change in the FE solution over time takes place in this area. In case of DEIM the plotted 10 interpolation points (with associated basis) provide a satisfactory accuracy. But for EIM we have to take 3 more interpolation points. The two methods determine the same 11th and 12th point. The 13th point selected by EIM is close to

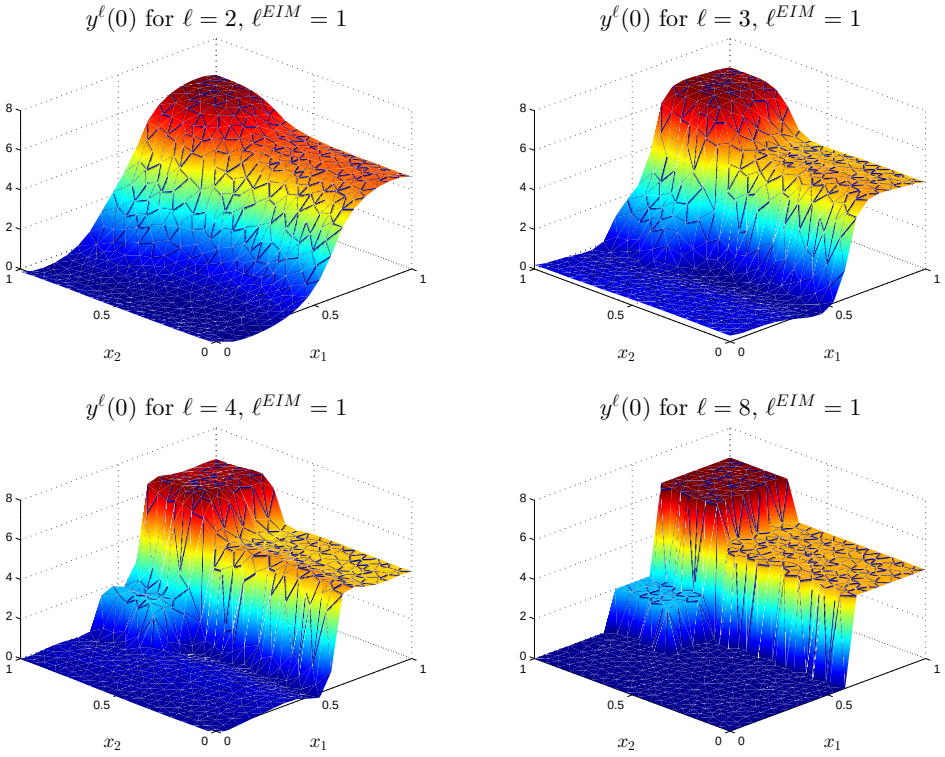


Figure 6.9: Example II, $X = H$: POD-DEIM solution $y^\ell(0)$ for $\ell = 2, 3, 4, 8$ with $\ell^{EIM} = 1$.

the 7th DEIM interpolation point.

In Figure 6.9 we show the POD-DEIM approximation of the inital function for different POD basis ranks ℓ when using $\ell^{EIM} = 1$ interpolation point for DEIM. The respective POD solutions at initial time computed without DEIM look identical. Remember that the FE solution at initial time can be seen in the right plot in Figure 6.2. As mentioned above, only one interpolation point for DEIM suffices here to obtain satisfactory results. It is the POD method itself that needs several basis functions to accurately reproduce the FE initial function. Regarding the shape of the first four POD basis functions in Figure 6.5 this becomes clear. We can clearly comprehend the contribution of the third and fourth basis function to the shape of the POD (-DEIM) solution at initial time.

From Figure 6.8 we know that one interpolation point does not suffice to obtain an accurate POD-DEIM approximation of the FE solution at final time. In Figure 6.10 we show the final deviation of the POD solution and of several POD-DEIM solutions from the FE solution. The plots illustrate how the increase of the number ℓ^{EIM} successively reduces the error and how the shape of the curves tends towards the shape of the error curve obtained for the POD solution. For $\ell^{EIM} = 8$ interpolation points the resulting error curve is no longer visually distinguishable from the curve associated with the POD solution, as to expect when taking a look at the upper right plot in Figure 6.8.

Computational efficiency: Let us finally sum up the computational performance of the POD method using $X = H$ and $\ell = 8$ with and without EIM or DEIM. The CPU times are shown in Table 6.11. For comparison the CPU time required for solving the fine

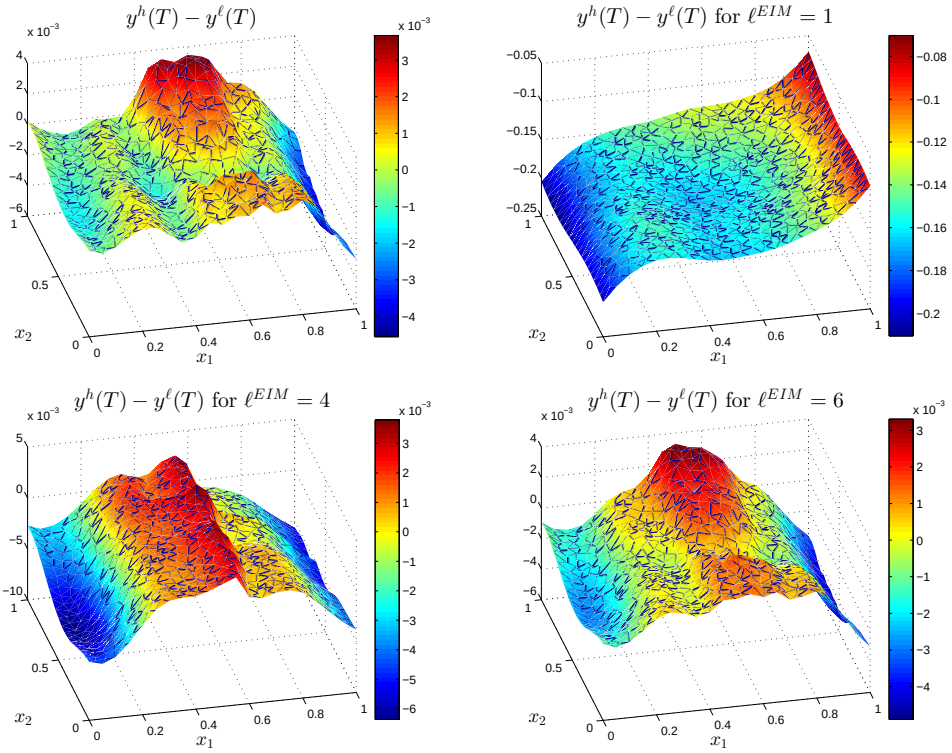


Figure 6.10: Example II, $X = H$: Error $y^h(T) - y^\ell(T)$ of the POD solution and of the POD-DEIM solution for $\ell^{EIM} = 1, 4, 6$ with $\ell = 8$.

	FE	POD	EIM	DEIM	ROM	ROM-EIM	ROM-DEIM
CPU time	3.03	0.20	0.05	0.06	0.51	0.33	0.32

Table 6.11: Example II, $X = H$: Performance of the FE and POD method measured in seconds for $\ell = 8$, with DEIM using $\ell^{EIM} = 10$ and with EIM using $\ell^{EIM} = 13$.

model is added. The solution of the ROM is about 6 times faster than the solution of the fine model. For a fair comparison we use $\ell^{EIM} = 10$ interpolation points for DEIM and $\ell^{EIM} = 13$ points for EIM. Then, both empirical interpolation methods deliver the same POD error. The speedup factor resulting from adding EIM or DEIM is about 1.6. Let us mention that the use of EIM with $\ell^{EIM} = 30$ interpolation points results in a CPU time of 0.46 seconds for solving ROM-EIM. This is almost the CPU time required for solving the ROM without EIM.

In Example II the reduction of computational costs when solving the ROM instead of the fine model is satisfactory. But it is not as large as in Example I. Here, the spatial dimension is reduced from $N_x = 727$ to $\ell = 8$. This yields a speedup factor of about 6. In Example I the FE dimension is given by $N_x = 2810$ and reduced down to $\ell = 4$. This leads to a factor of about 13 ($X = H$) or 19 ($X = V$).

By adding empirical interpolation to the ROMs a second speedup is achieved. Our results show that EIM and DEIM perform both very well. In Example II we obtain a

speedup factor of about 1.6 ($\ell^{EIM} = 10/13$) and in the first example we gain a factor of about 3 ($\ell^{EIM} = 4$).

For the solution of the optimal control problem in the next section we use the maximal edge length $h_{max} = 0.06$. This gives $N_x = 498$ spatial degrees of freedom. The time step size is set to $\Delta t = 0.004 \approx h_{max}^2$. In preparation for the solution of (**P**), we solve the state equation with specifications as in Example I and II using the modified discretization parameters. Unfortunately, a noteworthy speedup by adding empirical interpolation to the ROMs can not be achieved. For exemplification we name the CPU times measured in seconds analogous to Table 6.11 for Example II: ROM: 0.27, ROM-EIM: 0.24, ROM-DEIM: 0.23 (The absolute POD error of the corresponding solutions is smaller than 0.004). For the chosen mesh grid size the resulting N_x-dimensional evaluation of the nonlinearity and of its Jacobian seem comparatively not to be costly enough to make the use of empirical interpolation beneficial.

6.2 Solution of the Optimal Control Problem

The optimal boundary control problem (**P**) is numerically solved by applying a globalized Newton-CG method to the equivalent reduced problem ($\hat{\mathbf{P}}$). This strategy involves the repeated solving of several PDEs. We apply the FE Galerkin discretization and the POD Galerkin method. Using the latter technique, the POD basis is either fixed or updated adaptively. For the test examples of this section the procedure consists of three steps:

Step (1): **FE optimization: LSNCG-FE**
The LSNCG method (Algorithm 3 from Section 3) is applied by using the FE Galerkin discretization for computation of an optimal FE control $\bar{u}^h$ with associated optimal FE state $\bar{y}^h := y(\bar{u}^h)$ and associated optimal FE adjoint state $\bar{p}^h := p(\bar{u}^h)_1$.

Step (2): **POD optimization with a fixed POD basis: LSNCG-POD**
The optimization is carried out with the LSNCG strategy using the POD Galerkin method. For a chosen POD basis rank ℓ we obtain a suboptimal control $\bar{u}^\ell$ with associated suboptimal state $\bar{y}^\ell := y^\ell(\bar{u}^\ell)$.

Step (3): **POD optimization with an update of the POD basis: TR-POD**
The TR-POD algorithm (Algorithm 7 from Section 5.2) is applied and yields an optimal control $\bar{u}^{TR}$. The associated FE state is denoted by $\bar{y}^{TR} := y(\bar{u}^{TR})$. For comparison we also run a TR-FE optimization procedure which consists in applying the basic trust region algorithm (Algorithm 6 from Section 5.2) by using the FE Galerkin technique.

We define certain parameters, recall the input data of the algorithms and introduce the quantities we make use of:

Discretization parameters: The FE triangulation and the time grid are generated from the values

$$\boxed{h_{max} := 0.06, \quad \Delta t := 0.004.}$$

The dimension of the FE subspace V^h is then given by $N_x = 498$.

LSNCG algorithm:

- *Forcing sequence*: According to Remark 3.3, the iterates η_k are defined by

$$\eta_k := \min\left\{\left(\frac{\|\hat{J}'(u^{(k)})\|_U}{\|\hat{J}'(u^{(0)})\|_U}\right)^{\eta_a}, \eta_b \frac{\|\hat{J}'(u^{(k)})\|_U}{\|\hat{J}'(u^{(0)})\|_U}\right\} \quad \text{with } \eta_a := 1.5,\ \eta_b := 10^{-2}.$$

  Additionally, in step (1), we replace η_a by 1 or 2 and/or η_b by 10^{-1} in order to see the effect on the number of required inexact Newton steps for FE optimization and on the number of needed CG iterations. The differences are mentioned briefly.

- *Termination criterion*: We have to define a value $\tau_r \in (0,1)$ that specifies the desired relative reduction in the norm of the initial gradient and an absolute tolerance bound τ_a, $0 < \tau_a \ll 1$.

- *Initial point*: Throughout, we start the optimization at $u^{(0)} := 0$. Thus, at the beginning we always consider the uncontrolled situation where no external heating source is applied.

- *Negative curvature test*: We choose $\varepsilon_{cv} := 10^{-6}$.

- *Armijo algorithm*: We use the constant $\beta := 0.5$ for the backtracking strategy. The constant $\alpha \in (0,1)$ for the sufficient decrease condition (3.10) remains to be defined.

Gradient algorithm: Optionally, in step (1), we generate a reference optimal FE control u^{ref} by applying a simple gradient algorithm; compare Remark 3.4(2). We denote by $y^{ref} := y(u^{ref})$ the associated optimal FE state. The required parameters are taken over from the LSNCG method. The optimal LSNCG-FE control and state differ themselves from the unknown exact analytical values due to discretization errors. The absolute FE error when solving the given parabolic PDEs is of order $\mathcal{O}(h_{max}^2 + \Delta t)$; see Example I. A comparison of $(\bar{u}^h, \bar{y}^h)$ to (u^{ref}, y^{ref}) gives us at least a sense of the accuracy of the optimal FE solution pair $(\bar{u}^h, \bar{y}^h)$.
The difference between any computed control u and the optimal FE control $\bar{u}^h$ can be measured by

$$\varepsilon_{abs}^u(u) := \|\bar{u}^h - u\|_{U_h}, \quad \varepsilon_{rel}^u(u) := \frac{\|\bar{u}^h - u\|_{U_h}}{\|\bar{u}^h\|_{U_h}}. \tag{6.4}$$

The discrete norm $\|\cdot\|_{U_h}$ in U can be looked up in Section 3.2. Furthermore, we define

$$\Delta J(u) := \hat{J}(u) - \hat{J}(\bar{u}^h). \tag{6.5}$$

Any FE state y^h can be compared to the optimal FE state $\bar{y}^h$ by the time-averaged quantities

$$\varepsilon_{abs}^y(y^h) := \sum_{j=0}^{N_t} \|\bar{y}^h(t_j) - y^h(t_j)\|_H, \quad \varepsilon_{rel}^y(y^h) := \sum_{j=0}^{N_t} \frac{\|\bar{y}^h(t_j) - y^h(t_j)\|_H}{\|\bar{y}^h(t_j)\|_H}. \tag{6.6}$$

POD basis computation: The variant 'eigs'&'gs' is used. We denote by u_{bc} the control which is utilized to set up the POD basis. We can only take state snapshots

for the basis computation ('basis B_S') or we can add the snapshots from the adjoint state ("basis B_{SA}"). In the LSNCG-POD approach we choose a control u_{bc}, build up the ROMs and run the optimization. When pursuing the TR-POD strategy the control u_{bc} with corresponding POD basis is changed adaptively during optimization.

Accuracy of suboptimal controls: The accuracy of a suboptimal control $\bar{u}^\ell$ obtained by LSNCG-POD is measured with respect to the optimal FE control $\bar{u}^h$. We consider the time-averaged error quantities

$$e^u_{abs}(\ell) := \|\bar{u}^h - \bar{u}^\ell\|_{U_h}, \quad e^u_{rel}(\ell) := \frac{\|\bar{u}^h - \bar{u}^\ell\|_{U_h}}{\|\bar{u}^h\|_{U_h}}, \tag{6.7}$$

analogous to (6.4). In addition, we measure how good a suboptimal state $\bar{y}^\ell$ approximates the optimal FE state $\bar{y}^h$ via definition (6.2) with $y^h = \bar{y}^h$ and $y^\ell = \bar{y}^\ell$.

Trust region method: The *initial point*, the *termination criterion* and the parameter ε_{cv} are chosen as for the LSNCG algorithm. For the computation of an inexact Newton step in iteration k we also use $\eta_{CG} := \eta_k$. It remains to specify:

- Initial trust region radius $\Delta^{(0)}$.
- Constants $\alpha_{fcd} \in (0, 1]$ and $\beta_{fcd} \geq 1$ for the fraction of Cauchy decrease condition. We choose $\alpha_{fcd} := 1$ and $\beta_{fcd} := 1$.
- Constants $0 < \eta_1 \leq \eta_2 < 1$ and $0 < \gamma_1 \leq \gamma_2 < 1 \leq \gamma_3$. These are used to control and modify the trust region radius.
- **TR-POD**: An initial number ℓ_0 and a maximal number ℓ_{max} of POD basis functions.

Cost functional: Recall that the cost functional can be written as $J(y, u) = J_1(y) + J_2(u)$, with

$$J_1(y) = \frac{1}{2}\|y(T) - y_d\|^2_H, \quad J_2(u) = \frac{1}{2}\sum_{k=1}^{\mathsf{k}} \gamma_k \|u_k\|^2_{L^2(0,T)}.$$

6.2.1 Example III

This example is constructed with the aid of Example III in [42] (which is Run 4.2 in [43]). Basically, the examples coincide except for the control space and the added nonlinearity (instead of $N(y) \equiv 0$). The modification of the control space is explained below. First, we consider the linear function $N(y) \equiv 0$ and test the FE based Newton-CG procedure for solving the resulting linear-quadratic optimal control problem. Afterwards, we exclusively focus on the solution of the semilinear optimal control problem. The specifications for the example are directly named for the semilinear problem. The admissible set U_{ad} is not defined; see the argumentation in Remark 2.20(1). Let $t \in [0, T]$, $\boldsymbol{x} = (x_1, x_2) \in \Omega$.

(SE): $f(t, \boldsymbol{x}) \equiv 0, \quad N(y) := y^3, \quad q := 0.01, \quad c_p := 10, \quad \mathsf{k} := 4,$ $y_0(\boldsymbol{x}) := 3 - 4(x_2 - 0.5)^2.$
Cost functional: $y_d(\boldsymbol{x}) := 2 + 2\lvert 2x_1 - x_2\rvert, \quad \gamma := 0.01, \ \gamma_k := 4/\mathsf{k} \cdot \gamma.$

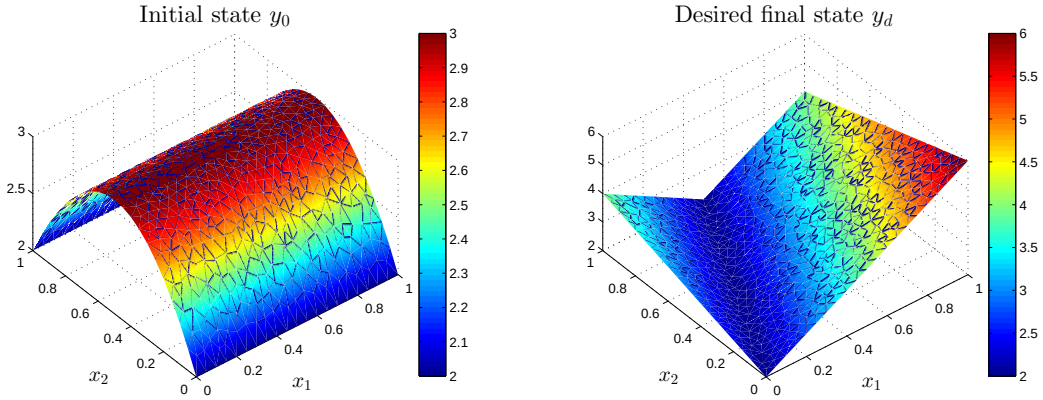


Figure 6.11: Example III: Initial function y_0 (left) and desired final state function y_d (right).

In Figure 6.11 we visualize the initial condition y_0 (left plot) and the desired final state y_d (right plot). Due to the smoothing property of the heat equation, see Example II, any solution to (SE) at time $t > 0$ can not possess the kink of the desired final state function.

Linear-quadratic optimal control problem:
As already mentioned, we pursue the FE based Newton-CG strategy for the solution of a linear-quadratic optimal control problem with $N(y) \equiv 0$. In this work the results in [42, 43], where linear-quadratic problems have been studied, are extended. We do not only move on to semilinear problems but we also consider an other control space. We explain the connection to our ansatz. In [42, 43] the control space equals $L^2(\Sigma)$. For the required discretization controls are expanded in the FE basis: Denote by N_b the number of FE boundary nodes and by $\varphi_{i_1}, \ldots, \varphi_{i_{N_b}}$ the associated FE basis functions (those basis functions which are non-zero on the boundary). Then, any $\tilde{u} \in L^2(\Sigma)$ is approximated by

$$\tilde{u}^h(t) = \sum_{j=1}^{N_b} \tilde{\mathrm{u}}_j(t)\varphi_{i_j} \quad \text{for } t \in [0,T], \tag{6.8}$$

with N_b degrees of freedom. We define the continuous linear operator B_2 analogously to the operators B_s from the proof of Theorem 2.4. It maps from U to $L^2(0,T;L^\infty(\Gamma)) \hookrightarrow L^2(0,T;L^2(\Gamma))$ and is given by $(B_2u)(t) = \sum_{k=1}^{\mathsf{k}} u_k(t)\chi_k$ for all $u \in U$, a.e. in $[0,T]$. Compared to formula (6.8) we have k degrees of freedom and the shape functions are 'on-off' functions instead of the FE 'hat' functions. Because of the different control spaces the cost functional in [42, 43] differs from our cost functional in the second summand. There, it is given by $\gamma/2\,\|\tilde{u}\|^2_{L^2(\Sigma)}$. We insert $\tilde{u} = B_2u$ and obtain

$$\gamma/2\,\|\tilde{u}\|^2_{L^2(\Sigma)} = 1/2 \sum_{k=1}^{\mathsf{k}} \Big(\gamma \int_{\Gamma_k} 1\,\mathrm{d}\boldsymbol{x}\Big)\|u_k\|^2_{L^2(0,T)}.$$

This is the reason why we set $\gamma_k = 4/\mathsf{k} \cdot \gamma$.

The discretization parameters here are the same as in Example III in [42]. This gives $N_b = 68$ boundary nodes. Firstly, we set $\mathsf{k} = 40$ in order to show how the thereby obtained

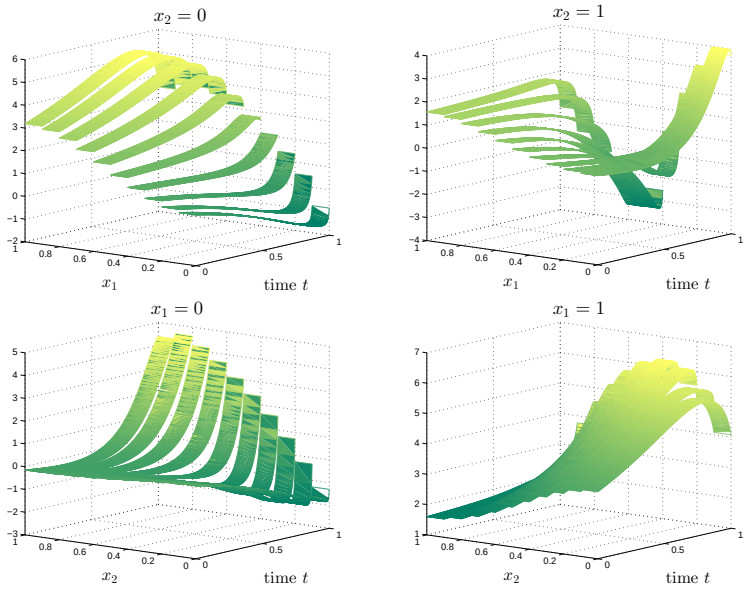


Figure 6.12: Example III with $N(y) \equiv 0$ and $\mathsf{k} = 40$: Optimal FE control $\bar{u}^h$ expanded in the control shape functions $\sum_{k=1}^{\mathsf{k}} \bar{u}_k^h(t)\chi_k(\boldsymbol{x})$ for $\boldsymbol{x} = (x_1, 0) \in \Gamma$ (upper left), $\boldsymbol{x} = (x_1, 1) \in \Gamma$ (upper right), $\boldsymbol{x} = (0, x_2) \in \Gamma$ (lower left), $\boldsymbol{x} = (1, x_2) \in \Gamma$ (lower right).

optimal FE control $\bar{u}^h$ compared to [42, Example III] looks like. Taking $\mathsf{k} = 40$ means that each boundary part of the unit square is again divided into ten subsegments. Secondly, we decrease the number of boundary parts to $\mathsf{k} = 4$ in order to illustrate the impact on the shape of the optimal FE state $\bar{y}^h$ at final time.

Let $N(y) \equiv 0$. In this case the gradient $\hat{J}'(\cdot)$ is linear and the Hessian $\hat{J}''(\cdot)$ is constant, as noted in Remark 2.16. Consequently, the Newton method finds the solution in one iteration and a globalization strategy (line search or trust region) is dispensable. Recall that the reduced cost functional $\hat{J}$ is convex for linear functions N. Hence, the first-order necessary optimality condition (2.24) is sufficient for global optimality. Remember that the state space is given by $W(0,T)$ for the linear-quadratic problem and that the admissible set U_{ad} can equal the entire Hilbert space U. Hence, in case of $U_{\mathsf{ad}} = U$ a stationary point of $\hat{J}$ is indeed an optimal control for problem (**P**). As fixed above, the initial iterate is taken to be zero. We terminate the CG algorithm as soon as $\|\hat{J}''(0)d^{(0)} + \hat{J}'(0)\|_{U_h} \leq 10^{-4}$ holds. Then, the optimal FE control $\bar{u}^h$ equals $d^{(0)}$.

The value of the reduced cost functional at the initial iterate is given by $\hat{J}(0) = 0.67475$. Let $\mathsf{k} = 40$. Then, the FE optimization takes about 3.64 seconds CPU time. Only 5 CG iterations are needed. We obtain $\hat{J}(\bar{u}^h) = 0.19959$ with $J_1(\bar{y}^h) = 0.07782$. The optimal FE control $\bar{u}^h$ is visualized in Figure 6.12. We plot the control expanded in the control shape functions. In doing so, we can compare the optimal FE control $\bar{u}^h$ to the optimal FE control shown in Figure 6.18 in [42] (or Figure 2 in [43]). The latter is obtained by a primal-dual active set strategy. Now, we run the optimization for $\mathsf{k} = 4$. The computation of the optimal FE control $\bar{u}^h$ needs 3.40 seconds CPU time. As for $\mathsf{k} = 40$, the CG algorithm performs 5 iterations. The minimal value of the reduced cost functional is given by $\hat{J}(\bar{u}^h) = 0.26777$ with $J_1(\bar{y}^h) = 0.15637$. Thus, the difference between $\bar{y}^h(T)$

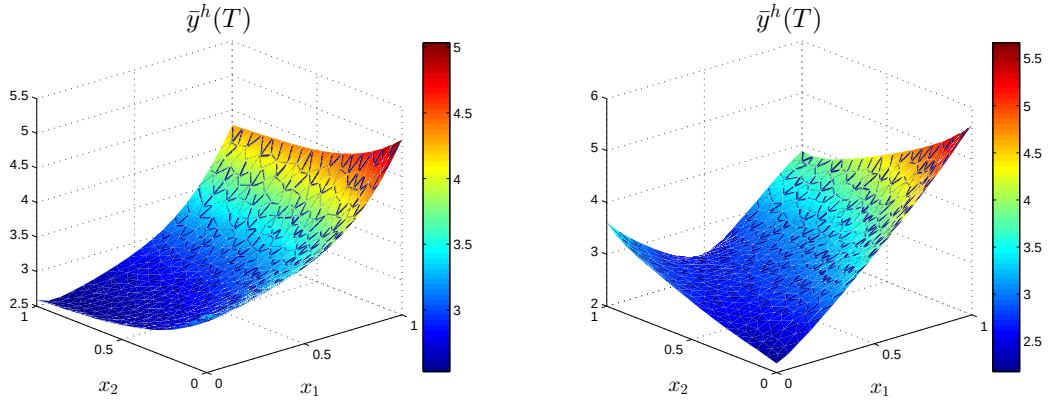


Figure 6.13: Example III with $N(y) \equiv 0$: Final optimal FE state $\bar{y}^h(T)$ for $\mathsf{k} = 4$ (left) and $\mathsf{k} = 40$ (right).

and the desired state y_d, measured by J_1, is twice as large as for $\mathsf{k} = 40$. The optimal FE states at final time are shown in Figure 6.13. For $\mathsf{k} = 40$ (right plot) the function $\bar{y}^h(T)$ has a clear parabolic shape on the boundary part Γ_3. But when taking $\mathsf{k} = 4$ (left plot) the optimal FE state $\bar{y}^h(T)$ is strictly monotone on Γ_3. Notice that the different boundary parts of the unit square are heated or cooled as complete parts over time in case of $\mathsf{k} = 4$.

To sum up, the Newton-CG strategy for solving the considered linear-quadratic optimal boundary control problem works very well and extremely efficiently. The latter follows from the fact that the CG algorithm converges rapidly. When permitting a larger number k of boundary segments the desired final state y_d can be reached far better. Nevertheless, we choose $\mathsf{k} = 4$ for the solution of the semilinear optimal control problem. On the one hand, this is closer to practical applications. On the other hand, this allows a more concrete handling of the control variable.

Semilinear optimal control problem:
We denote by y^h the FE state associated with the initial control $u^{(0)} = 0$. In Figure 6.14 we visualize the FE state $y^h(t) \in V^h$ at several time steps $t = t_j$. This illustrates how the state evolves over time when the boundary parts are neither heated nor cooled. If the reader is interested in the respective temperature distribution over time which results from the linear equation with $N(y) \equiv 0$, we refer to Figure 6.15 in [42]. The uncontrolled final FE state $y^h(T)$ is shown in the lower right plot in Figure 6.14. It differs completely in shape and size from the desired final state y_d (right plot in Figure 6.11). In this uncontrolled case the value of the reduced cost functional is given by $\hat{J}(0) = J_1(y^h) = 1.76667$.

We specify the parameters for the LSNCG algorithm by

$$\boxed{\tau_a := 10^{-3},\ \tau_r := 10^{-3},\ \alpha := 10^{-2}}$$

Step (1):
We directly summarize the LSNCG-FE run in Table 6.12. In the first column of this table we name the iteration number k. The second column contains the reduced cost functional value $\hat{J}(u^{(k)})$ and the third column the U_h-norm of the gradient $\hat{J}'(u^{(k)})$. In the remaining columns we list the value η_k of the forcing sequence, the number of needed CG iterations

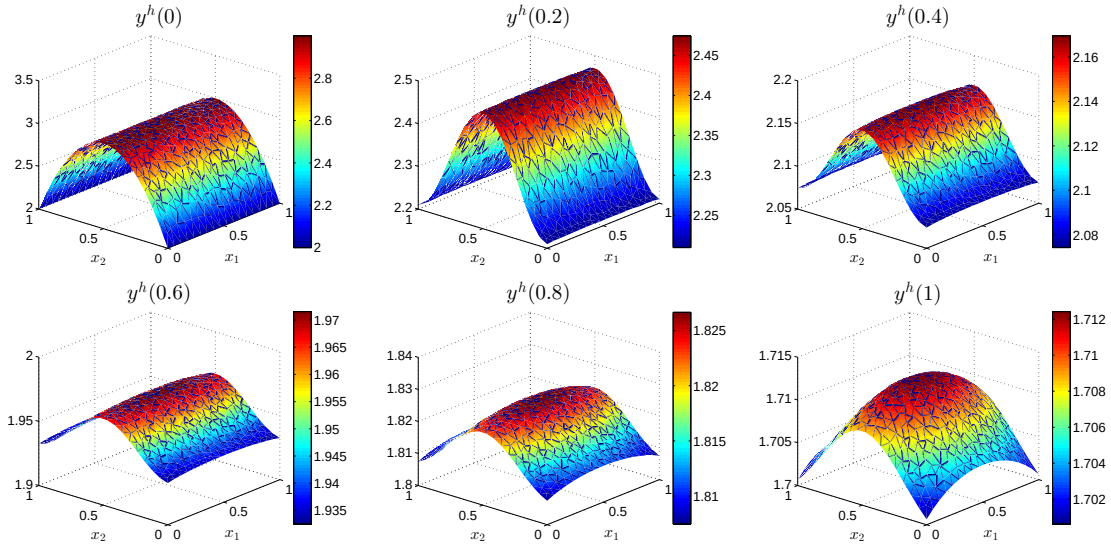


Figure 6.14: Example III: FE state $y^h(t)$ associated with $u^{(0)} = 0$ at six different time steps $t = t_j$.

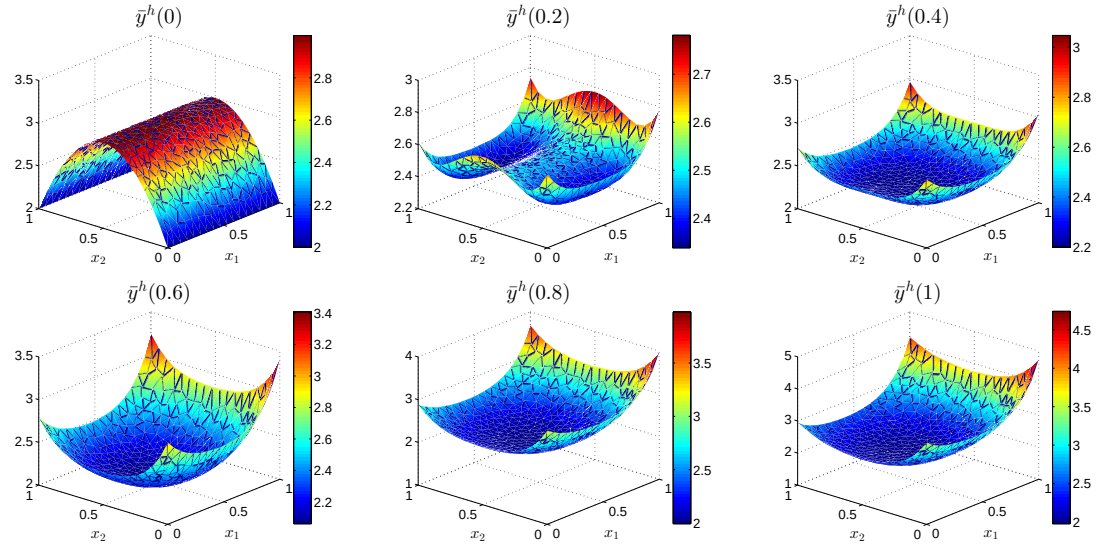


Figure 6.15: Example III: Optimal FE state $\bar{y}^h(t)$ at six different time steps $t = t_j$.

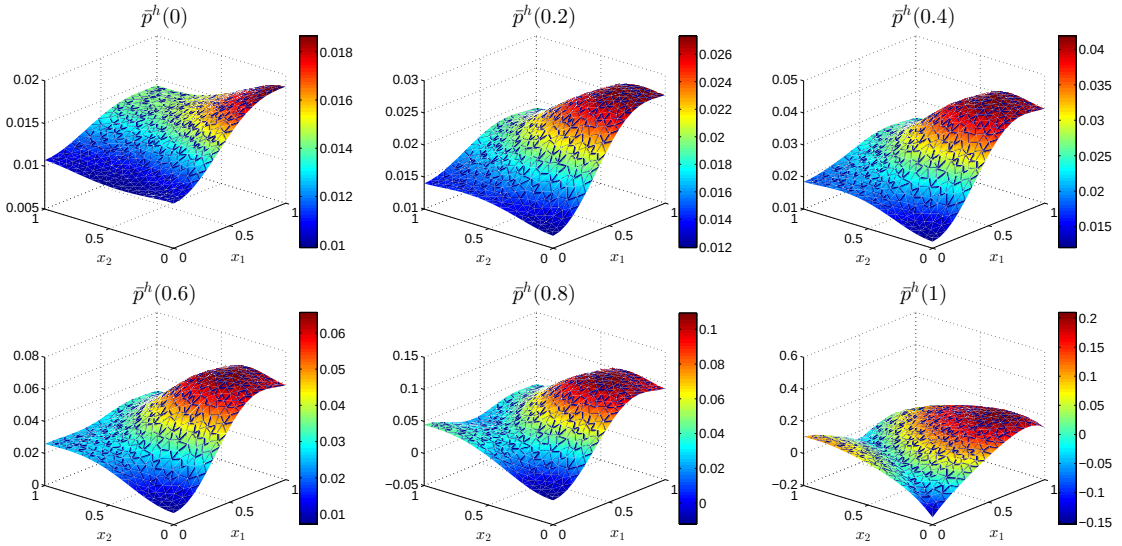


Figure 6.16: Example III: Optimal FE adjoint state $\bar{p}^h(t)$ at six different time steps $t = t_j$.

k	$\hat{J}(u^{(k)})$	$\|\hat{J}'(u^{(k)})\|_{U_h}$	η_k	#CG it	$\|d^{(k)}\|_{U_h}$	t_k
0	1.76667	$2.4 \cdot 10^{-1}$	0.0100	3	6.59	1
1	0.93123	$3.7 \cdot 10^{-2}$	0.0016	4	1.62	1
2	0.90240	$2.3 \cdot 10^{-3}$	0.0001	5	0.11	1
3	0.90237	$9.1 \cdot 10^{-6}$				

Table 6.12: Example III: Iterations of the LSNCG-FE run.

and the U_h-norm of the obtained inexact Newton step $d^{(k)}$. The last column contains the determined Armijo step length t_k. The FE optimization takes 25.10 seconds CPU time. As we can see, the LSNCG-FE procedure works also very efficiently for the solution of the given semilinear optimal control problem. The CG algorithm needs only 3, 4 and 5 iterations to satisfy the termination criterion. Moreover, a variation of η_b to 10^{-1} and/or a change of η_a to 1 or 2 does not result in a different number of (inexact) Newton iterations. It is only the number of CG iterations which changes to one more or one less, as the case may be. Negative curvature is not detected and the Armijo algorithm does not have to reduce the step length 1. Consequently, the line search part of the algorithm, being the globalization of the local inexact Newton method, could be omitted for this example. The minimal reduced cost functional value is given by $\hat{J}(\bar{u}^h) = 0.90237$.

In Figure 6.15 we present the optimal FE state $\bar{y}^h$. The peaks that $\bar{y}^h(t_j)$ possesses in the corners of the unit square at time steps $t_j > 0$ are conspicuous. The optimal FE adjoint state $\bar{p}^h$ is illustrated in Figure 6.16. The values of the adjoint state are in the range of -0.15 to 0.21 whereas the state values lie between 1.9 and 4.8. The four components of the optimal FE control $\bar{u}^h$ over time are shown in the left plot in Figure 6.17. We observe that most activity in the components is located towards the end of the time interval.

We compare the optimal LSNCG-FE pair $(\bar{u}^h, \bar{y}^h)$ with the reference optimal FE pair (u^{ref}, y^{ref}). Remember that the reference optimal FE control with associated state is

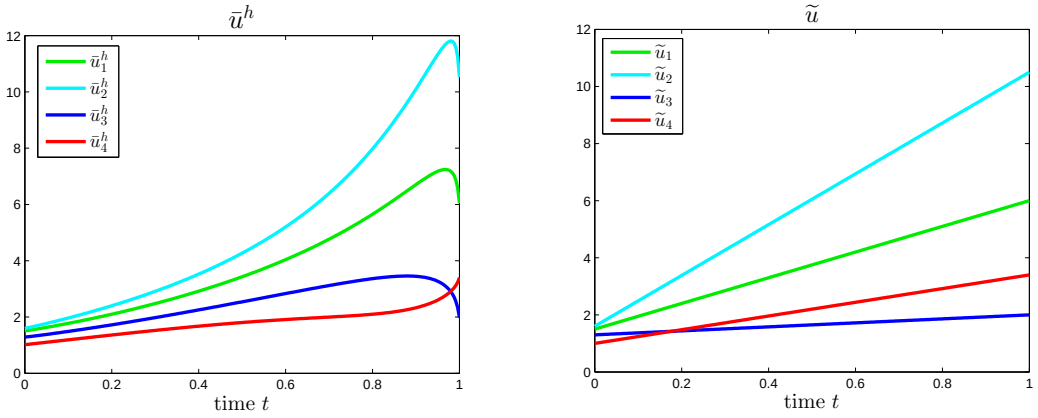


Figure 6.17: Example III: Optimal FE control $\bar{u}^h$ (left) and control $\widetilde{u}$ (right).

h_{max} (N_x)	Δt	$\varepsilon^u_{abs}(u^{ref})$	$\varepsilon^u_{rel}(u^{ref})$	$\varepsilon^y_{abs}(y^{ref})$	$\varepsilon^y_{rel}(y^{ref})$	$\Delta J(u^{ref})$
0.1 (177)	0.01	$1.4 \cdot 10^{-1}$	$1.8 \cdot 10^{-2}$	$3.6 \cdot 10^{-3}$	$1.5 \cdot 10^{-3}$	$3.8 \cdot 10^{-4}$
0.06 (498)	0.004	$9.4 \cdot 10^{-2}$	$1.2 \cdot 10^{-2}$	$2.4 \cdot 10^{-3}$	$9.7 \cdot 10^{-4}$	$-1.3 \cdot 10^{-4}$
0.03 (1932)	0.001	$3.9 \cdot 10^{-2}$	$5.0 \cdot 10^{-3}$	$8.4 \cdot 10^{-4}$	$3.4 \cdot 10^{-4}$	$-1.6 \cdot 10^{-5}$

Table 6.13: Example III: Comparison of optimal LSNCG-FE solution $(\bar{u}^h, \bar{y}^h)$ with reference optimal FE solution (u^{ref}, y^{ref}).

computed with a gradient algorithm. The results are shown in Table 6.13, where we additionally name the values for two other mesh sizes. For all three pairs of discretization parameters we observe: The absolute difference $\varepsilon^y_{abs}(y^{ref})$ between the states is smaller than $h^2_{max} \leq \Delta t$. But the absolute difference between the controls is more than one decimal power larger. When turning to the LSNCG-POD approach in step (2) we also expect the errors of the suboptimal controls to be larger than those of the suboptimal states. Bear in mind that the absolute deviation $\varepsilon^u_{abs}(u^{ref})$ equals $9.4 \cdot 10^{-2}$ for the actually chosen mesh and time step size. The last column of the table contains the values $\hat{J}(u^{ref}) - \hat{J}(\bar{u}^h)$. For $h_{max} = 0.06$ and $\Delta t = 0.004$ this difference is about $-1.3 \cdot 10^{-4}$. Let us add that the difference $J_1(y^{ref}) - J_1(\bar{y}^h)$ is even of order $4.8 \cdot 10^{-3}$. The left plot in Figure 6.18 shows the relative deviation $|\bar{y}^h(T) - y^{ref}(T)|/|\bar{y}^h(T)|$.

Step (2):
We choose the space $X = H$ for the POD method. The difference to the V-norm implementation is so small that only the results for the H-norm implementation are presented. The POD basis is computed from both state and adjoint snapshots ('basis B_{SA}').

In [42, 43] a primal-dual active set strategy is applied for the solution of the considered linear-quadratic optimal boundary control problem (with control space $L^2(\Sigma)$). In these works it is shown that the control which is utilized for the POD basis computation has to be chosen 'well enough' in order to obtain good POD suboptimal controls.

This is why we investigate the use of four different controls u_{bc} which are more and less good in the sense of being similar to the optimal FE control $\bar{u}^h$. These controls are

	$u_{bc} = 0$			$u_{bc} = u^m$		
ℓ	$e^u_{abs}(\ell)$	$e^u_{rel}(\ell)$	$e^y_{abs}(\ell)$	$e^u_{abs}(\ell)$	$e^u_{rel}(\ell)$	$e^y_{abs}(\ell)$
1	$3.3 \cdot 10^{+0}$	$4.2 \cdot 10^{-1}$	$2.4 \cdot 10^{-1}$	$2.9 \cdot 10^{+0}$	$3.6 \cdot 10^{-1}$	$1.5 \cdot 10^{-1}$
5	$1.1 \cdot 10^{+0}$	$1.4 \cdot 10^{-1}$	$1.2 \cdot 10^{-1}$	$7.3 \cdot 10^{-1}$	$9.2 \cdot 10^{-2}$	$3.8 \cdot 10^{-2}$
15	$1.1 \cdot 10^{+0}$	$1.4 \cdot 10^{-1}$	$1.0 \cdot 10^{-1}$	$2.0 \cdot 10^{-1}$	$2.6 \cdot 10^{-2}$	$1.6 \cdot 10^{-2}$
21	$5.4 \cdot 10^{-1}$	$6.8 \cdot 10^{-2}$	$5.7 \cdot 10^{-2}$	$1.8 \cdot 10^{-1}$	$2.3 \cdot 10^{-2}$	$1.4 \cdot 10^{-2}$
32	$4.9 \cdot 10^{-1}$	$6.2 \cdot 10^{-2}$	$5.1 \cdot 10^{-2}$	$1.1 \cdot 10^{-1}$	$1.4 \cdot 10^{-2}$	$1.2 \cdot 10^{-2}$
33	$4.9 \cdot 10^{-1}$	$6.2 \cdot 10^{-2}$	$5.1 \cdot 10^{-2}$	$\mathbf{5.5 \cdot 10^{-2}}$	$6.9 \cdot 10^{-3}$	$8.0 \cdot 10^{-3}$
64	$2.4 \cdot 10^{-1}$	$3.0 \cdot 10^{-2}$	$2.8 \cdot 10^{-2}$	$\mathbf{2.5 \cdot 10^{-2}}$	$3.1 \cdot 10^{-3}$	$2.8 \cdot 10^{-3}$

Table 6.14: Example III, LSNCG-POD, $X = H$, 'basis B_{SA}': Absolute & relative errors in the control variable and absolute errors in the state variable for $u_{bc} = 0$ and $u_{bc} = u^m$ for different ℓ.

utilized to set up the POD basis for the LSNCG-POD strategy. The chosen controls are $u_{bc} = 0$, $u_{bc} = \bar{u}^h$, $u_{bc} = u^m$ with $u^m(t) \equiv (3.8, 5.1, 2.5, 1.8)^T$ and $u_{bc} = \widetilde{u}$ with

$$\widetilde{u}(t) := \begin{pmatrix} 1.5 + 4.5 \cdot t \\ 1.6 + 8.9 \cdot t \\ 1.3 + 0.7 \cdot t \\ 1.0 + 2.4 \cdot t \end{pmatrix}.$$

The control u^m is obtained by taking the componentwise mean value of the optimal FE control $\bar{u}^h$ (the vectors $\bar{u}^h(t) \in \mathbb{R}^4$ are given at all time steps $t = t_j$). The control $\widetilde{u}$ is shown in the right plot in Figure 6.17. Let us begin by testing the controls $u_{bc} = 0$ and $u_{bc} = u^m$. Obviously, these two differ a lot from the optimal FE control $\bar{u}^h$. We run the LSNCG-POD optimization for different POD basis ranks ℓ. The absolute and relative errors in the control variable and the absolute errors in the state variable for selected numbers ℓ are listed in Table 6.14. From the comparison with the reference optimal FE control u^{ref} we infer that a suboptimal control which satisfies $e^u_{abs}(\ell) < 10^{-1}$ is at the latest as close to the optimal FE control $\bar{u}^h$ as the discretization error. As expected, the errors in the control variable are larger than those in the state variable and we observe:

- $u_{bc} = 0$: Even when increasing the POD basis rank ℓ the value $e^u_{abs}(\ell)$ remains larger than 10^{-1} and $e^y_{abs}(\ell)$ stays above 10^{-2}.

- $u_{bc} = u^m$: The errors decrease faster than those obtained from $u_{bc} = 0$. The absolute error $e^u_{abs}(15)$ for $u_{bc} = u^m$ is smaller than $e^u_{abs}(64)$ for $u_{bc} = 0$. For $\ell = 33$ this absolute error is of order $5.5 \cdot 10^{-2}$ and the absolute error in the state variable is given by $8.0 \cdot 10^{-3}$. By adding more POD basis functions the absolute error in the control variable does not drop beneath 10^{-2} and the absolute error in the state variable stays larger than 10^{-3}.

We are interested in significantly reduced computing times. Hence, the aim is to keep the POD basis rank ℓ as small as possible. The choice of $u_{bc} = 0$ or $u_{bc} = u^m$ does not yield a satisfactory decrease in the error of the suboptimal controls when increasing the number ℓ

	$u_{bc} = \widetilde{u}$			$u_{bc} = \bar{u}^h$		
ℓ	$e^u_{abs}(\ell)$	$e^u_{rel}(\ell)$	$e^y_{abs}(\ell)$	$e^u_{abs}(\ell)$	$e^u_{rel}(\ell)$	$e^y_{abs}(\ell)$
1	$2.8 \cdot 10^{+0}$	$3.6 \cdot 10^{-1}$	$1.6 \cdot 10^{-1}$	$2.9 \cdot 10^{+0}$	$3.7 \cdot 10^{-1}$	$1.5 \cdot 10^{-1}$
3	$2.5 \cdot 10^{+0}$	$3.1 \cdot 10^{-1}$	$5.0 \cdot 10^{-2}$	$2.4 \cdot 10^{+0}$	$3.1 \cdot 10^{-1}$	$4.9 \cdot 10^{-2}$
5	$6.6 \cdot 10^{-1}$	$8.3 \cdot 10^{-2}$	$2.2 \cdot 10^{-2}$	$5.3 \cdot 10^{-1}$	$6.6 \cdot 10^{-2}$	$1.4 \cdot 10^{-2}$
6	$3.1 \cdot 10^{-1}$	$3.9 \cdot 10^{-2}$	$1.7 \cdot 10^{-2}$	$3.4 \cdot 10^{-1}$	$4.3 \cdot 10^{-2}$	$7.0 \cdot 10^{-3}$
7	$2.7 \cdot 10^{-1}$	$3.4 \cdot 10^{-2}$	$1.6 \cdot 10^{-2}$	$1.5 \cdot 10^{-1}$	$1.9 \cdot 10^{-2}$	$3.2 \cdot 10^{-3}$
9	$2.4 \cdot 10^{-1}$	$3.1 \cdot 10^{-2}$	$1.5 \cdot 10^{-2}$	$1.3 \cdot 10^{-1}$	$1.7 \cdot 10^{-2}$	$2.2 \cdot 10^{-3}$
10	$2.2 \cdot 10^{-1}$	$2.7 \cdot 10^{-2}$	$1.5 \cdot 10^{-2}$	$\mathbf{3.8 \cdot 10^{-2}}$	$4.8 \cdot 10^{-3}$	$7.8 \cdot 10^{-4}$
15	$1.7 \cdot 10^{-1}$	$2.1 \cdot 10^{-2}$	$1.1 \cdot 10^{-2}$	$\mathbf{8.6 \cdot 10^{-3}}$	$1.1 \cdot 10^{-3}$	$1.1 \cdot 10^{-4}$
19	$1.3 \cdot 10^{-1}$	$1.6 \cdot 10^{-2}$	$9.2 \cdot 10^{-3}$	$\mathbf{1.1 \cdot 10^{-3}}$	$1.4 \cdot 10^{-4}$	$1.8 \cdot 10^{-5}$
20	$\mathbf{7.2 \cdot 10^{-2}}$	$9.1 \cdot 10^{-3}$	$7.4 \cdot 10^{-3}$	$\mathbf{5.4 \cdot 10^{-4}}$	$6.8 \cdot 10^{-5}$	$1.0 \cdot 10^{-5}$
21	$\mathbf{5.3 \cdot 10^{-2}}$	$6.7 \cdot 10^{-3}$	$7.0 \cdot 10^{-3}$	$\mathbf{5.2 \cdot 10^{-4}}$	$6.6 \cdot 10^{-5}$	$7.9 \cdot 10^{-6}$
64	$\mathbf{8.7 \cdot 10^{-3}}$	$1.1 \cdot 10^{-3}$	$1.8 \cdot 10^{-3}$	$\mathbf{8.3 \cdot 10^{-6}}$	$1.0 \cdot 10^{-6}$	$4.1 \cdot 10^{-7}$

Table 6.15: Example III, LSNCG-POD, $X = H$, 'basis B_{SA}': Absolute & relative errors in the control variable and absolute errors in the state variable for $u_{bc} = \widetilde{u}$ and $u_{bc} = \bar{u}^h$ for different ℓ.

of POD basis functions. In Table 6.15 we state the errors for the controls $u_{bc} = \widetilde{u}$ and $u_{bc} = \bar{u}^h$ analogously to Table 6.14 for adjusted numbers ℓ. It can be seen:

- $u_{bc} = \widetilde{u}$: The errors are smaller than those obtained from $u_{bc} = u^m$. For $\ell = 20$ the error $e^u_{abs}(\ell)$ is of order $7.2 \cdot 10^{-2}$ and the corresponding absolute error in the state variable is again about one decimal power smaller. Here, the error $e^u_{abs}(\ell)$ goes below 10^{-2} but stays larger than 10^{-3} when further increasing the POD basis rank ℓ.

- $u_{bc} = \bar{u}^h$: In this case the POD basis is computed from the snapshots of the optimal FE state $\bar{y}^h$ and the optimal FE adjoint state $\bar{p}^h$. By utilizing these optimal snapshots a POD basis rank $\ell = 10$ suffices to obtain an absolute error $e^u_{abs}(\ell)$ of order $3.8 \cdot 10^{-2}$. The absolute error $e^y_{abs}(10)$ in the state variable is even given by $7.8 \cdot 10^{-4}$. The errors continue to decrease when further increasing the number ℓ such that $e^u_{abs}(64)$ equals $8.3 \cdot 10^{-6}$. Let us add that the error $e^u_{abs}(\ell)$ stays larger than 10^{-6}, even for $\ell = 90$ POD basis functions.
  In case of $u_{bc} = \bar{u}^h$ compared to the other controls less or equivalent good suboptimal controls deliver suboptimal states which are closer to the optimal FE state $\bar{y}^h$. The reader is asked to compare the errors for instance as follows: $u_{bc} = \widetilde{u}$, $\ell = 64$ to $u_{bc} = \bar{u}^h$, $\ell = 10$ or $u_{bc} = u^m$, $\ell = 15$ to $u_{bc} = \bar{u}^h$, $\ell = 6$.

The above comparison shows: The better the control u_{bc} is chosen, the faster and the further the errors of the suboptimal controls and states decrease when increasing the POD basis rank ℓ. But even for $u_{bc} = u^m$ the suboptimal controls get closer than $e^u_{abs}(\ell) = 10^{-1}$ to the optimal FE control $\bar{u}^h$ solely by increasing the number of used POD basis functions.

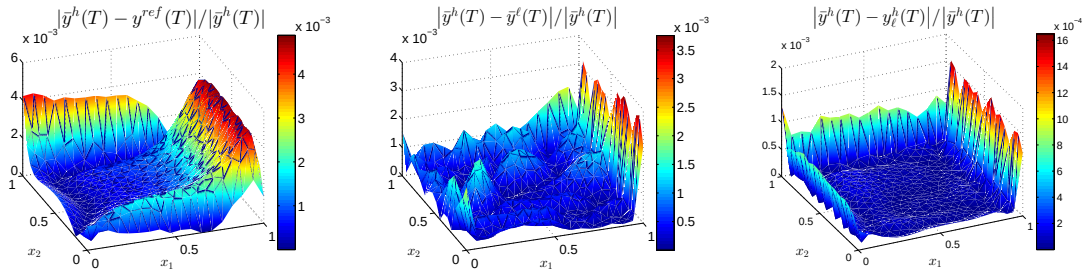


Figure 6.18: Example III: Relative deviations $|\bar{y}^h(T) - y^{ref}(T)|/|\bar{y}^h(T)|$ (left plot), $|\bar{y}^h(T) - \bar{y}^\ell(T)|/|\bar{y}^h(T)|$ (middle plot) and $|\bar{y}^h(T) - y^h_\ell(T)|/|\bar{y}^h(T)|$ (right plot). The suboptimal pair $(\bar{u}^\ell, \bar{y}^\ell)$ is obtained by LSNCG-POD with $X = H$ and 'basis B_{SA}' for $u_{bc} = \bar{u}^h$ and $\ell = 10$. We denote by y^h_ℓ the FE state associated with $\bar{u}^\ell$.

The state variable is controlled such that only its final deviation from the desired state y_d is measured by the cost functional. Let $(\bar{u}^\ell, \bar{y}^\ell)$ be the suboptimal solution obtained by LSNCG-POD when using $u_{bc} = \bar{u}^h$ and $\ell = 10$. In the middle plot in Figure 6.18 we present the relative deviation of $\bar{y}^\ell(T)$ from the final optimal FE state $\bar{y}^h(T)$. In the left graphic the relative deviation $|\bar{y}^h(T) - y^{ref}(T)|/|\bar{y}^h(T)|$ can be seen for comparison. The right plot shows the respective final relative deviation of the FE state y^h_ℓ associated with the suboptimal control $\bar{u}^\ell$. Please note the slight different scaling of the vertical axes. This shows that the POD suboptimal state $\bar{y}^\ell$ provides a very satisfactory approximation of the final optimal FE state $\bar{y}^h(T)$. The final FE state $y^h_\ell(T)$ is even closer to the final optimal FE state $\bar{y}^h(T)$.

Let us have a more detailed look at the errors in the state variable. For $u_{bc} \neq \bar{u}^h$ the errors of the suboptimal states are larger than those for $u_{bc} = \bar{u}^h$ when the errors of the corresponding suboptimal controls are in the same range (or even smaller for $u_{bc} \neq \bar{u}^h$). But how close are the associated FE states $y(\bar{u}^\ell)$ to the optimal FE state $\bar{y}^h$? For the given example we generally obtain $\varepsilon^y_{abs}(y(\bar{u}^\ell)) < e^y_{abs}(\ell)$ in case of $u_{bc} \neq \bar{u}^h$. In fact, the left value can even be about one decimal power smaller than the right one. In case of $u_{bc} = \bar{u}^h$ it occurs that $\varepsilon^y_{abs}(y(\bar{u}^\ell))$ is a bit larger than $e^y_{abs}(\ell)$. For exemplification we compare the suboptimal controls $\bar{u}^\ell$ for $\ell = 6$ ($u_{bc} = \bar{u}^h$) and $\ell = 15$ ($u_{bc} = u^m$). We have $e^u_{abs}(15) = 2.0 \cdot 10^{-1}$ and $e^u_{abs}(6) = 3.4 \cdot 10^{-1}$. Hence, the control for $\ell = 15$ is a bit closer to the optimal FE control $\bar{u}^h$. But the errors in the state variable are given by $e^y_{abs}(15) = 1.6 \cdot 10^{-2} > e^y_{abs}(6) = 7.0 \cdot 10^{-3}$. And we obtain that $\varepsilon^y_{abs}(y(\bar{u}^\ell))$ for $\ell = 15$ equals $2.9 \cdot 10^{-3}$ whereas the same value for $\ell = 6$ is given by $1.1 \cdot 10^{-2}$. Hence, the FE state $y(\bar{u}^\ell)$ for $\ell = 15$ is clearly closer to the optimal FE state $\bar{y}^h$ than the suboptimal state $\bar{y}^\ell$ for $\ell = 15$ or the FE state $y(\bar{u}^\ell)$ for $\ell = 6$. This is why we attach more importance to the error of the suboptimal control. Notice that in view of the heating process which is mathematically described by the optimal control problem the following counts: The control costs for a suboptimal control together with the deviation of the associated final FE state (not the suboptimal one) from the desired final state. Furthermore, from the pure mathematical point of view, a pair $(\bar{u}^\ell, \bar{y}^\ell)$ does not fulfill the side condition of the minimization problem (**P**), which is the semilinear heat equation.

Snapshot ensemble: It is essential to include adjoint snapshots in order to obtain satis-

ℓ	$e^u_{abs}(\ell)$	$e^u_{rel}(\ell)$	$e^y_{abs}(\ell)$
1	$2.9 \cdot 10^{+0}$	$3.7 \cdot 10^{-1}$	$1.5 \cdot 10^{-1}$
5	$2.6 \cdot 10^{+0}$	$3.2 \cdot 10^{-1}$	$5.1 \cdot 10^{-2}$
15	$4.1 \cdot 10^{-1}$	$5.2 \cdot 10^{-2}$	$1.4 \cdot 10^{-2}$
21	$2.9 \cdot 10^{-1}$	$3.7 \cdot 10^{-2}$	$8.4 \cdot 10^{-3}$
64	$\mathbf{9.5 \cdot 10^{-2}}$	$1.2 \cdot 10^{-2}$	$3.7 \cdot 10^{-3}$

Table 6.16: Example III, LSNCG-POD, $X = H$, 'basis B_S': Absolute & relative errors in the control variable and absolute errors in the state variable for $u_{bc} = \bar{u}^h$ for different ℓ.

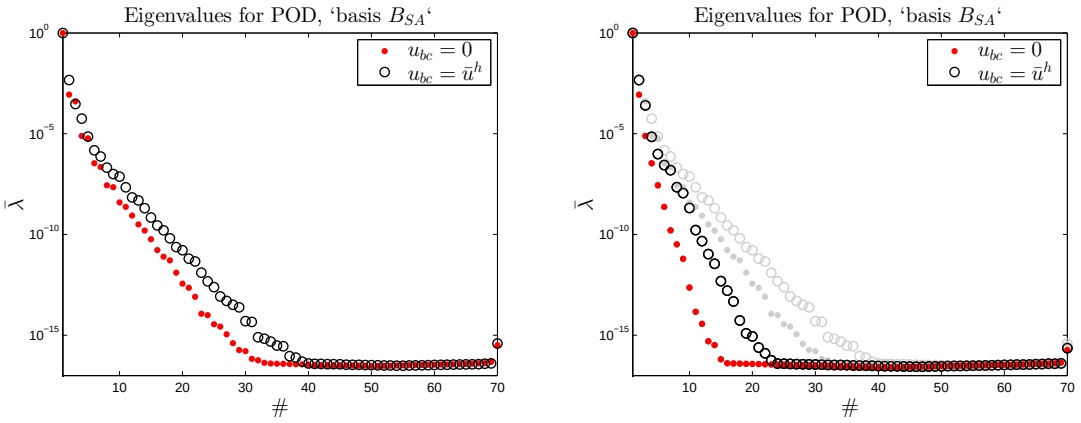


Figure 6.19: Example III: Decay of normalized eigenvalues for the POD basis computed from $u_{bc} = 0$ and from $u_{bc} = \bar{u}^h$ using $X = H$ and 'basis B_{SA}' (left) or 'basis B_S' (right).

factory results. In Table 6.16 we name the errors resulting from 'basis B_S' and $u_{bc} = \bar{u}^h$ for selected numbers ℓ of POD basis functions. Even for this best available control u_{bc} the errors are only a bit smaller than those resulting from $u_{bc} = 0$ and 'basis B_{SA}'. This observation is not surprising since the gradient $\hat{J}'(u)$, which is (negatively taken) the right-hand side in the Newton equation, depends directly on the adjoint state via formula (2.20).

Decay of POD eigenvalues: We present the decays of the POD eigenvalues computed from $u_{bc} = 0$ and from $u_{bc} = \bar{u}^h$. They are shown in Figure 6.19 for both variants 'basis B_{SA}' (left plot) and 'basis B_S' (right plot). For a better visual comparison we added the curves of the left plot to the right one in gray color. When we take the uncontrolled snapshots ($u_{bc} = 0$) for the POD basis computation the eigenvalues decay more rapidly than those obtained from the optimal FE control $\bar{u}^h$. We compare Figure 6.14 to Figure 6.15 and observe that the dynamics of the uncontrolled FE state compared to the optimal FE state $\bar{y}^h$ are less complex. Consequently, there is less 'information' incorporated in the corresponding snapshots. This results in a faster decay of the eigenvalues for 'basis B_S'. The same holds when using 'basis B_{SA}'. The adding of the adjoint snapshots leads to a considerable slower decay of the eigenvalues. We did not visualize the eigenvalues computed from $u_{bc} = u^m$ or $u_{bc} = \widetilde{u}$. Expectedly, they are located in between the plotted ones.

k	$\hat{J}^\ell(u^{(k)})$	$\|g_\ell^{(k)}\|_{U_h}$	η_k	#CG it	$\|d^{(k)}\|_{U_h}$	t_k
0	1.76780	$2.4 \cdot 10^{-1}$	0.0100	3	6.60	1
1	0.93070	$3.7 \cdot 10^{-2}$	0.0016	4	1.61	1
2	0.90230	$2.2 \cdot 10^{-3}$	0.0001	6	0.11	1
3	0.90227	$8.7 \cdot 10^{-6}$				

Table 6.17: Example III: Iterations of LSNCG-POD run 3.1.

Empirical interpolation methods: From the findings in Section 6.1, a noteworthy speedup by combining the ROMs with one of the two empirical interpolation methods is not to be expected. This can be confirmed by our observations. But the use of EIM or DEIM is not only not beneficial with respect to computing time, it is also critical concerning accuracy. We apply EIM or DEIM for the approximation of the nonlinear term $\mathrm{N}^\ell(\mathrm{y}^\ell)$ and of its Jacobian $\left(\mathrm{N}^\ell\right)'(\mathrm{y}^\ell)$. The latter is needed for the solution of the state equation (by a Newton method in each time iteration) and it also occurs in the discretized adjoint equations and the discretized linearized state equation. If we use EIM or DEIM only within the state solver, this leads to satisfactory suboptimal controls (compared to the results without EIM or DEIM). But if EIM or DEIM are added to all ROMs, the Armijo algorithm generally interferes in iteration $k = 1$ and reaches a predefined maximal number of iterations without satisfying the sufficient decrease condition (3.10). The obtained results are poor. Let us just mention the following: We used the given example setting with a 'nicer' desired final state y_d. The results for the ROM-EIM or ROM-DEIM implementation have then been satisfactory in terms of accuracy. But again, a remarkable acceleration compared to the standard POD approach could not be achieved.

Now, we take a detailed look at one selected LSNCG-POD run. For **LSNCG-POD run 3.1** we use $X = H$, 'basis B_{SA}', $u_{bc} = \bar{u}^h$ and $\ell = 10$. Remember that the relative deviations which are shown in Figure 6.18 belong to this run. The optimization is summarized in Table 6.17. For the POD Galerkin approximation of the gradient $\hat{J}'(u^{(k)})$ we use the notation $g_\ell^{(k)}$ which is known from the TR-POD strategy. Comparing this table to Table 6.12 emphasizes the good approximation quality of the POD method applied in optimal control. The optimization runs are almost one-to-one identical and the values differ only slightly. In all our LSNCG-POD runs (without EIM or DEIM) the Armijo algorithm does not have to reduce the step length 1 except for $\ell = 1$ and $u_{bc} \neq \bar{u}^h$. Basically, we observe the following: The more the control u_{bc} resembles the optimal FE control $\bar{u}^h$ and/or the larger the POD basis rank ℓ, the closer the inexact Newton steps computed in the LSNCG-POD run are to those of the LSNCG-FE run. We come back to this in step (3).

Computational efficiency: In Table 6.18 we compare the performance with respect to the computational time of LSNCG-POD run 3.1 and the LSNCG-FE procedure. The overall CPU time to compute the POD suboptimal control is 5.33 seconds. This is 4.7 times smaller than the CPU time required for the computation of the optimal FE control $\bar{u}^h$. Thereof, the snapshot generation takes 1.93 seconds. The solution of the full-order non-

	CPU time	
LSNCG-FE optimization	25.10	
CG algorithm	17.29	(4.44, 5.61, 7.24)
Snapshot generation	1.93	
- state	1.23	
- adjoint state	0.70	
POD basis ($\ell = 15$)	0.21	
LSNCG-POD optimization	3.19	
CG algorithm	1.77	(0.43, 0.50, 0.84)

Table 6.18: Example III: CPU times measured in seconds of LSNCG-POD run 3.1 compared to LSNCG-FE optimization (in brackets: CPU time needed by the CG algorithm in iteration $k = 0, 1, 2$).

linear state equation needs about 1.8 times as long as the solution of the full-order adjoint equation. The values in parenthesis are the CPU times needed by the CG algorithm to compute the inexact Newton steps. One CG iteration requires one application of the Hessian, i.e. the solution of (LSE) and (AE_2). When we apply the POD Galerkin method we have to solve two reduced-order linear equations (the evaluation of the Jacobian $(N^\ell)'(\cdot)$ still depends on the FE dimension N_x) per CG iteration instead of two full-order equations. This results in an average speedup factor of 9.8 for computing an inexact Newton step in LSNCG-POD run 3.1 compared to the LSNCG-FE run. For the first two iterations the factor is even larger than 10. For the last iteration it is just of order 8.6 because the CG algorithm in iteration $k = 2$ of LSNCG-POD run 3.1 needs one iteration more than in the LSNCG-FE run.

We investigate one last aspect in view of TR-POD optimization. In each TR-POD iteration we have to compute the quotient $\rho^{(k)}(d_t^{(k)})$. Before evaluating this term we test if the actual reducion $\mathsf{ared}^{(k)}(d_t^{(k)})$ and the predicted reduction $\mathsf{pred}^{(k)}(d_t^{(k)})$ are both still 'large enough'. This motivates to compute the difference $\Delta J(\bar{u}^\ell)$ for some suboptimal controls $\bar{u}^\ell$ to get a feeling for the tolerance we have to choose. The results are listed in Table 6.19. We additionally name the corresponding values of the reduced cost functional. From Table 6.13 we know that $\Delta J(u^{ref})$ equals $-1.3 \cdot 10^{-4}$. This means that the reference optimal FE control u^{ref} yields a smaller reduced cost functional value than the optimal (LSNCG-)FE control $\bar{u}^h$. This is as well the case for the suboptimal control $\bar{u}^7$ computed from $u_{bc} = \bar{u}^h$. And even for $u_{bc} = 0$ with $\ell = 64$ and for $u_{bc} = u^m$ with $\ell = 21$ we obtain $\Delta J(\bar{u}^\ell) < 0$. This is quite surprising. Actually, a computed control u can differ $\varepsilon^u_{abs}(u) = 2.4 \cdot 10^{-1}$ from the optimal FE control $\bar{u}^h$ and nevertheless yield a smaller reduced cost functional value. Apparently, even for the (very worse) control $u_{bc} = 0$ we can find an ℓ such that $\bar{u}^\ell$ differs only in the range of the discretization error from the optimal FE control $\bar{u}^h$. In practical applications, one can not compare a computed suboptimal control to an optimal FE control $\bar{u}^h$ because the latter one is simply not given (the computation is too expensive or even not feasible). A-posteriori error estimates are one possibility to estimate the distance; see [24, 42, 43, 46]. When applying the presented TR-POD strategy such an estimator is not needed since convergence is guaranteed. This property is certainly the main strength of the TR-POD procedure.

		$\hat{J}(\bar{u}^\ell)$	$\Delta J(\bar{u}^\ell)$
$u_{bc} = 0$	$\ell = 64$	0.90234	$-2.5 \cdot 10^{-5}$
$u_{bc} = u^m$	$\ell = 21$	0.90234	$-2.3 \cdot 10^{-5}$
$u_{bc} = \widetilde{u}$	$\ell = 9$	0.90269	$3.3 \cdot 10^{-4}$
$u_{bc} = \bar{u}^h$	$\ell = 5$	0.90371	$1.3 \cdot 10^{-3}$
$u_{bc} = \bar{u}^h$	$\ell = 7$	0.90214	$-2.3 \cdot 10^{-4}$
$u_{bc} = \bar{u}^h$	$\ell = 10$	0.90228	$-8.4 \cdot 10^{-5}$
$\hat{J}(\bar{u}^h)$		0.90237	

Table 6.19: Example III: Reduced cost functional values and difference $\Delta J(\bar{u}^\ell)$ for some suboptimal controls $\bar{u}^\ell$ obtained by LSNCG-POD with $X = H$ and 'basis B_{SA}'.

Step (3):
In step (1) and in step (2) for $\ell > 1$ the Newton-CG method could be applied without the added line search strategy. Hence, we also expect the TR-POD algorithm to accept the POD based inexact Newton steps throughout the iterations, if the POD basis rank ℓ is controlled appropriately. Remember that the number ℓ of used POD basis functions is determined such that gradient accuracy in the sense of Carter holds. But there is no guarantee that this number yields an equally good POD Galerkin approximation of the Hessian. We choose the initial trust region radius $\Delta^{(0)}$ such that the first inexact Newton step lies inside the trust region. For all TR-POD runs we use

$$\ell_{max} := 15, \quad \Delta^{(0)} := 8, \quad \eta_1 := 0.1,\ \eta_2 := 0.8,$$
$$\gamma_1 := 0.5,\ \gamma_2 := 1,\ \gamma_3 := 1.2.$$

TR-FE optimization: With the above setting the iterates are identical to those generated by the LSNCG-FE procedure. This should be the case for an appropriate choice of parameters because the trust region strategy just replaces the line search globalization. The two things we look at are the ratios $\rho^{(k)}(d_t^{(k)})$ and the numerical effort. We obtain $\rho^{(0)}(d_t^{(0)}) = 1.09$, $\rho^{(1)}(d_t^{(1)}) = 0.98$ and $\rho^{(2)}(d_t^{(2)}) = 0.29$. The latter value is that small since the last inexact Newton step is only of length $\|d_t^{(2)}\|_{U_h} = 0.11$ and hence close to the discretization error. But the first two values point out the good approximation quality of the quadratic model function (5.2). The TR-FE run takes 33.06 seconds. This shows that the used trust region globalization is more costly than the applied line search strategy (25.10 seconds). The main reason is the required testing of the fraction of Cauchy decrease condition including the computation of Cauchy steps.

We turn to the TR-POD method. According to step (2), we choose $X = H$ and use 'basis B_{SA}' for the POD method. For the first TR-POD run, **TR-POD run 3.1**, we choose the initial number $\ell_0 = 3$ of POD basis functions. In Table 6.20 we present

k	$\hat{J}(u^{(k)})$	$\|\hat{J}'(u^{(k)})\|_{U_h}$	#CG it	$\|d_t^{(k)}\|_{U_h}$	$\Delta^{(k)}$	$\rho^{(k)}(d_t^{(k)})$	ℓ	$\zeta_\ell^{(k)}$
0	1.76667	$2.4 \cdot 10^{-1}$	3	7.39	8.0	1.00	3	0.044
1	0.93774	$3.6 \cdot 10^{-2}$	4	2.23	9.6	0.90	5	0.118
2	0.90292	$5.7 \cdot 10^{-3}$	5	0.41	11.5	0.62	10	0.087
3	0.90224	$8.5 \cdot 10^{-4}$						

Table 6.20: Example III: Iterations of TR-POD run 3.1.

the iterations of this run. We name the reduced cost functional value $\hat{J}(u^{(k)})$ and the U_h-norm of the gradient $\hat{J}'(u^{(k)})$ (FE evaluations). Then, we state the needed number of (reduced-order) CG iterations for computation of the inexact Newton step. If this step is not accepted as trial step, we put the number in brackets. As well, the step length $\|d_t^{(k)}\|_{U_h}$ is named. Note that this step length corresponds to the product $t_k\|d^{(k)}\|_{U_h}$ in the LSNCG method, where t_k denotes the Armijo step length and $d^{(k)}$ the inexact Newton direction. In addition, we list the trust region radius $\Delta^{(k)}$ and the ratio $\rho^{(k)}(d_t^{(k)})$ of actual and predicted reduction. The last two columns contain the determined number ℓ of POD basis functions and the ratio $\zeta_\ell^{(k)}$. In every iteration the POD based inexact Newton step satisfies the fraction of Cauchy decrease condition and provides a successful step. In the first two iterations the quotient $\rho^{(k)}(d_t^{(k)})$ is close to one. The value $\rho^{(2)}(d_t^{(2)})$ equals 0.62 and is hence not as small as in the TR-FE run. Here, the corresponding inexact Newton step is still of order 0.41. We can see that the number of POD basis fucntions in increased from 3 to 5 to 10. This shows: In the beginning, very few POD basis functions suffice to fulfill the Carter condition. When moving towards the optimal control, meaning that the norm of the gradient decreases and that the state and adjoint state become richer in dynamical features, a higher POD basis rank is needed.

Shape of POD basis functions: We take a look at the change in shape of the first four POD basis functions during TR-POD run 3.1. The basis functions computed from $u_{bc} = u^{(0)} = 0$ are shown in Figure 6.20. In the shape of the upper basis functions we recognize the shape of the uncontrolled state $y^h(t)$ at earlier time steps $t = t_j$ (Figure 6.14). The shape of the third basis function corresponds to the shape of the uncontrolled adjoint state (not visualized). For comparison we plot the first four POD basis functions computed from $u_{bc} = \bar{u}^h$ in Figure 6.21. They are considered to be optimal. The first two basis functions reflect the dynamics of the optimal FE state $\bar{y}^h(t)$ at later time steps $t = t_j$ (Figure 6.15). The fourth basis function relates to the optimal FE adjoint state $\bar{p}^h$ (Figure 6.16). Analogously, we now visualize the first four POD basis functions computed from $u_{bc} = u^{(1)}$ in Figure 6.22. We have to look closely at the figures to detect the very small differences to the first four POD basis functions computed from $u_{bc} = \bar{u}^h$. When applying an OS-POD strategy one can observe a similar adaptation of shape of the POD basis functions; see [16, 31]. The control $u^{(1)}$ differs $\varepsilon_{abs}^u(u^{(1)}) = 2.23$ and $\varepsilon_{rel}^u(u^{(1)}) = 0.28$ from the optimal FE control $\bar{u}^h$. It is shown in the lower left plot in Figure 6.23.

In our second TR-POD run, **TR-POD run 3.2**, we modify TR-POD run 3.1 such that the POD basis computed from $u_{bc} = u^{(1)}$ in the second iteration is kept for further optimization steps. In Table 6.21 we summarize the results. It is only a little different to that

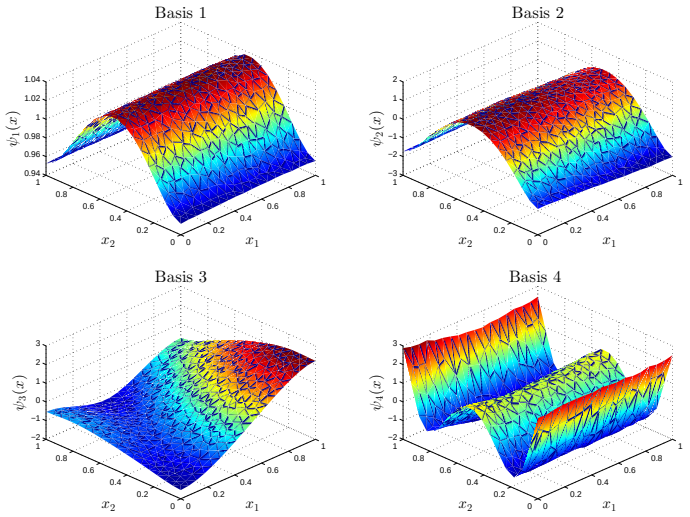


Figure 6.20: Example III: First four POD basis functions computed from $u_{bc} = 0$ with $X = H$ and 'basis B_{SA}'.

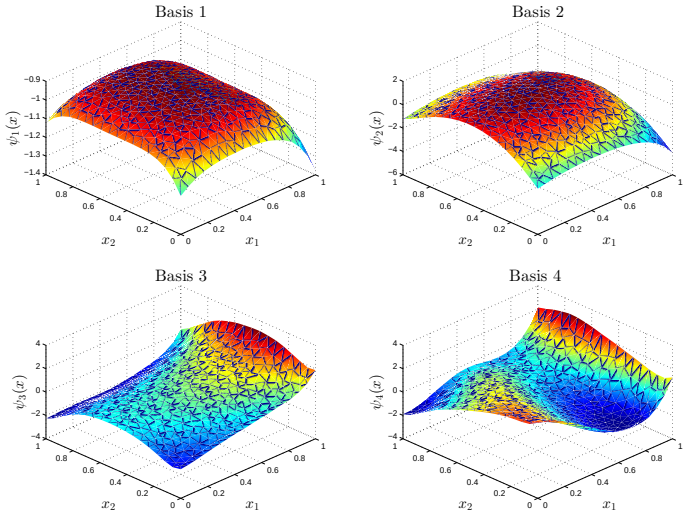


Figure 6.21: Example III: First four POD basis functions computed from $u_{bc} = \bar{u}^h$ with $X = H$ and 'basis B_{SA}'.

k	$\hat{J}(u^{(k)})$	$\|\hat{J}'(u^{(k)})\|_{U_h}$	#CG it	$\|d_t^{(k)}\|_{U_h}$	$\Delta^{(k)}$	$\rho^{(k)}(d_t^{(k)})$	ℓ	$\zeta_\ell^{(k)}$
2	0.90292	$5.7 \cdot 10^{-3}$	6	0.39	11.5	0.53	14	0.187
3	0.90238	$9.1 \cdot 10^{-4}$						

Table 6.21: Example III: Last optimization step of TR-POD run 3.2.

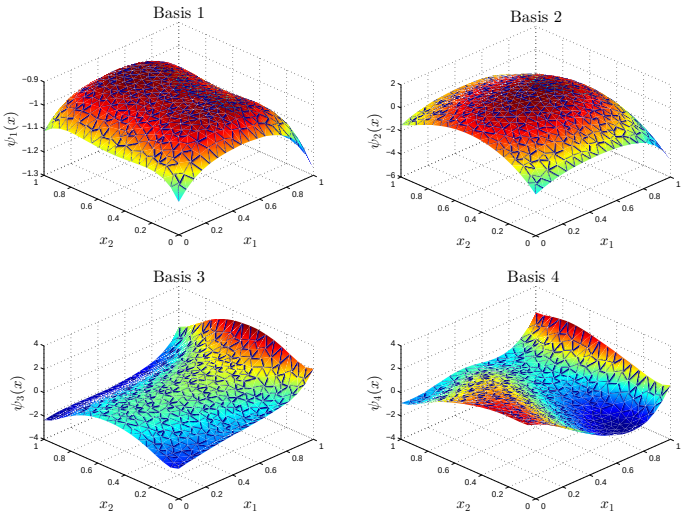


Figure 6.22: Example III: First four POD basis functions computed from $u_{bc} = u^{(1)}$ in TR-POD run 3.1.

of TR-POD run 3.1. The Carter condition is first fulfilled for $\ell = 14$ POD basis functions, instead of $\ell = 10$. In addition, the CG algorithm needs one iteration more. But this shows: The FE state and the FE adjoint state associated with the second control iterate $u^{(1)}$ have enough features in common with the optimal FE state and the optimal FE adjoint state in the sense that the POD optimization process can be continued and terminated utilizing the corresponding POD basis. Recall that the first POD based Newton step is computed from only amazing $\ell = 3$ POD basis functions (upper plots and lower left plot in Figure 6.20).

Gradient accuracy: We name the values $\zeta_\ell^{(k)}$ of TR-POD run 3.1 and 3.2 in Table 6.22. In addition, we state these values for $k = 0$ and different numbers ℓ in case of using 'basis B_S', i.e. when only utilizing state snapshots for the POD basis computation. We observe that the required gradient accuracy in the first iteration can only be reached when using 'basis B_{SA}'. This confirms the necessity of adding adjoint snapshots to the state snapshot ensemble.

Next, we consider **TR-POD run 3.3** where we start the optimization directly with $\ell_0 = 10$ POD basis functions. The iterations can be seen in Table 6.23. The reduced cost functional value $\hat{J}(u^{(1)})$ is just about $6{\cdot}10^{-3}$ smaller than that in TR-POD run 3.1 although 7 POD basis functions more are used. But the interesting thing is that the optimization is terminated within the third iteration because the actual reduction $\mathsf{ared}^{(2)}(d_t^{(2)})$ is only of order $1.1 \cdot 10^{-5}$ while the predicted reduction $\mathsf{ared}^{(2)}(d_t^{(2)})$ equals $1.2 \cdot 10^{-4}$. Actually, $\hat{J}(u^{(2)}) < \hat{J}(\bar{u}^h)$ already holds. But the gradient norm $\|\hat{J}'(u^{(2)})\|_{U_h}$ is still above the chosen tolerance bound $\tau_r\|\hat{J}'(u^{(0)})\|_{U_h} + \tau_a$. The reader might question if the constants τ_r and τ_a are adequately defined. But we can justify our choice. For consistence we take the same constants for the different considered strategies. If the constants are increased such that TR-POD run 3.3 accepts the third control iterate $u^{(2)}$, we obtain in some extent worser LSNCG-POD results (one iteration less is sometimes carried out). To survey the

TR-POD run 3.1						TR-POD run 3.2		‘basis B_S’	
$k=0$		$k=1$		$k=2$		$k=2$		$k=0$	
ℓ	$\zeta_\ell^{(k)}$	ℓ	$\zeta_\ell^{(k)}$	ℓ	$\zeta_\ell^{(k)}$	ℓ	$\zeta_\ell^{(k)}$	ℓ	$\zeta_\ell^{(k)}$
1	0.341	3	0.921	5	0.882	5	0.973	3	0.338
2	0.332	4	0.234	6	0.494	6	0.662	6	0.338
3	0.044	5	0.118	7	0.449	7	0.463	7	0.337
				8	0.253	8	0.464	10	0.323
				9	0.242	9	0.296	14	0.323
				10	0.087	10	0.294	15	0.317
						13	0.220	17	0.300
						14	0.187	30	0.283

Table 6.22: Example III: Values $\zeta_\ell^{(k)}$ of TR-POD run 3.1 and 3.2 and for ‘basis B_S’, $k=0$ for different ℓ.

k	$\hat{J}(u^{(k)})$	$\|\hat{J}'(u^{(k)})\|_{U_h}$	#CG it	$\|d_t^{(k)}\|_{U_h}$	$\Delta^{(k)}$	$\rho^{(k)}(d_t^{(k)})$	ℓ	$\zeta_\ell^{(k)}$
0	1.76667	$2.4 \cdot 10^{-1}$	3	7.39	8.0	1.00	10	0.001
1	0.93147	$3.3 \cdot 10^{-2}$	5	1.90	9.6	0.96	10	0.023
2	0.90225	$2.1 \cdot 10^{-3}$						

Table 6.23: Example III: Iterations of TR-POD run 3.3.

size of the actual and predicted reduction during the TR-POD procedure provides an additional termination criterion. Generally, the TR-POD method does not lead to equally small gradient norms as obtained by the FE implementation. But the minimal values of the reduced cost functional are often smaller than the minimal value $\hat{J}(\bar{u}^h)$. The reader is asked to compare the respective values in the given above tables. We come back to TR-POD run 3.3. We have seen that the optimal control is found with one iteration less compared to the TR-FE or LSNCG-FE run. The reason is that the POD based inexact Newton steps differ from the FE based inexact Newton steps.

This is why we compare the second control iterate of several optimization runs. For illustration we add one TR-POD run. In **TR-POD run 3.4** we take the FE Hessian as Hessian of the quadratic model function (5.2). This means that only the gradient therein is replaced by its POD Galerkin approximation. For comparability to LSNCG-POD 3.1 we use $\ell_0 = 10$ POD basis functions for the first optimization step. In Figure 6.23 we visualize the second control iterate of LSNCG-POD run 3.1 (upper left), of TR-POD run 3.4 (upper right), of TR-POD run 3.1 (lower left) and of TR-POD run 3.3 (lower right). The control components in gray are those of the optimal FE control $\bar{u}^h$. Recall from Table 6.12 that the initial inexact Newton step in the LSNCG-FE run (being equal to the TR-FE run) is of length $\|d^{(0)}\|_{U_h} = 6.59$. In LSNCG-POD run 3.1 we have $\|d^{(0)}\|_{U_h} = 6.60$ (Table 6.17). Hence, the length of this first POD based inexact Newton step ($u_{bc} = \bar{u}^h$, $\ell = 10$) nearly equals the length of the first FE based inexact Newton step. And the resulting second iterates are merely visually distinguishable. This is why only the first control iterate $u^{(1)}$ of LSNCG-POD run 3.1 is shown in Figure 6.23 (upper left plot). For TR-POD run 3.4 we obtain $\|d_t^{(0)}\|_{U_h} = 6.59$ and the control iterate $u^{(1)}$ (lower right plot) is

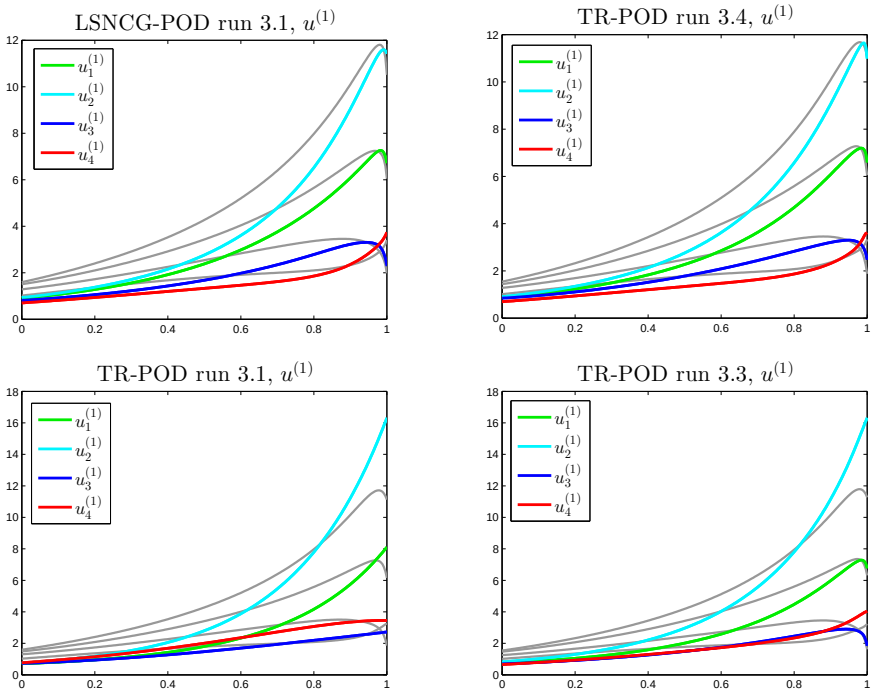


Figure 6.23: Example III: Second control iterate $u^{(1)}$ of LSNCG-POD run 3.1 (upper left), of TR-POD run 3.4 (upper right), of TR-POD run 3.1 (lower left) and of TR-POD run 3.3 (lower right).

visually nearly identical to that of the LSNCG-FE run, as for LSNCG-POD run 3.1. But in both TR-POD run 3.1 and 3.3 we obtain $\|d_t^{(0)}\|_{U_h} = 7.39$ (Tables 6.20, 6.23). Hence, the steps are larger than in the other considered runs but they do not result in better second control iterates, as the lower plots show. The gradient accuracy is guaranteed by the Carter condition. In LSNCG-POD run 3.1 the value $\zeta_\ell^{(0)}$ with $\ell = 10$ equals 0.038. In TR-POD run 3.1 we have $\zeta_\ell^{(0)} = 0.044$ with $\ell = 3$ and in TR-POD run 3.3 it holds $\zeta_\ell^{(0)} = 0.001$ with $\ell = 10$ (Table 6.23). Let us add that the use of the FE gradient in TR-POD run 3.3 yields the step length $\|d_t^{(0)}\|_{U_h} = 7.52$. We conclude that the differences in the iterates mainly arise from differences in the Hessian approximation. Recall that the ROMs of (LSE) and (AE_2) are built up from a POD basis computed from state and adjoint snapshots ((SE) and (AE_1)). The use of a POD basis that represents more the dynamics of the optimally controlled state and adjoint state instead of a POD basis computed from the current iterate results in a more accurate POD Galerkin approximation of the Hessian. The TR-POD method guarantees convergence by means of Cauchy decrease without requiring a certain accuracy of the model Hessian. But we are interested in performing inexact Newton steps. They have to satisfy the fraction of Cauchy decrease condition to be accepted as trial steps. For our strategy a certain accuracy of the reduced-order approximation of the Hessian is of course essential. To sum up, the TR-POD iterates move differently towards the optimal control compared to the FE based iterates mainly by reason of the Hessian approximation. Only by chance in TR-POD run 3.3 this different path of iterates results in less total iterations.

	CPU time	
	total	per iteration
Snapshot generation (4)	7.76	1.93
- state (4)	4.93	1.22
- adjoint state (4)	2.83	0.71
POD basis (3)	0.63	0.21
Carter condition (3)	3.12	0.34, 0.96, 1.82
CG algorithm (3)	1.50	0.38, 0.47, 0.65
Cauchy step (3)	0.36	0.12
Complete optimization	13.87	

Table 6.24: Example III: CPU times measured in seconds of TR-POD run 3.1.

Computational effort: We list the needed CPU time for the different parts of TR-POD run 3.1 in Table 6.24. The overall CPU time for the computation of $\bar{u}^{TR}$ is 13.87 seconds. This is about 1.8 times smaller than the time needed by the LSNCG-FE run and about 2.4 times smaller than the CPU time of the TR-FE run. But our main interest lies in the computational effort required by the different parts of the algorithm. The numbers in paranthesis in Table 6.24 name the total numbers of computations or evaluations. With given snapshots a POD basis computation is very cheap. It just takes 0.21 seconds. The evaluation of the reduced cost functional with given state snapshots and the computation of the FE gradient with given adjoint snapshots are so fast that they can merely be measured. For each testing of the Carter condition we have to solve the ROMs of (SE) and (AE_1) to build up the approximative POD gradient. As expected, it is not very efficient just to set $\ell = \ell+1$ as long as the condition is not satisfied. Recall that the POD basis rank ℓ must be increased from 3 ($k = 0$) to 5 ($k = 1$) to 10 ($k = 2$). The computation of all three inexact Newton steps by the CG algorithm takes only 1.50 seconds. On average, a Cauchy step is computed in 0.12 seconds. Hence, here it is not costly to compute the latter one in order to make use of the fraction of Cauchy decrease condition. The snapshot generation takes even more than half of the CPU time. From TR-POD run 3.2 we know that the last update of the POD basis is not needed when taking a few more POD basis functions for the last step. The FE state $y(u^{(3)})$ is required for the evaluation of the actual reduction $\mathsf{ared}^{(2)}(d_t^{(2)})$ and the FE adjoint state $p(u^{(3)})$ is needed to compute the FE gradient $\hat{J}'(u^{(3)})$. Hence, two full-order solutions of (SE) and (AE_1) could be avoided by realizing the modifications named in Section 5.2. In comparison with the OS-POD strategy, we could interpret the computation of $u^{(1)}$ as determination of a better initializing control for POD optimization. All in all it takes 3.08 seconds CPU time to compute $u^{(1)}$. But unlike the OS-POD strategy presented in [16] the POD optimization here is then based on an adaptive determination of the number of POD basis functions in the course of optimization (instead of using an a-posteriori error estimate to determine the required fixed number of POD basis functions, as the name says, a-posteriori).

6.2.2 Example IV

For this example we keep the problem setting from Example III except for the desired final state y_d. It is replaced such that negative curvature in the Hessian of the reduced

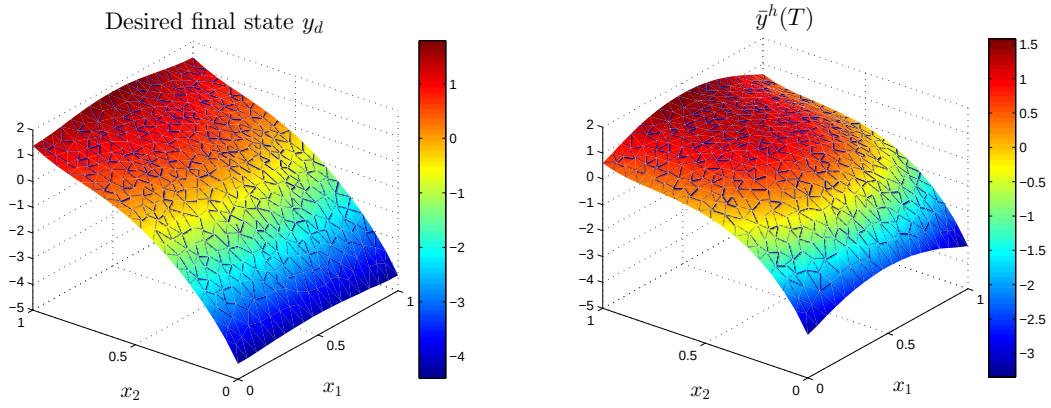


Figure 6.24: Example IV: Desired final state function y_d (left) and final optimal FE state $\bar{y}^h(T)$.

cost functional occurs. Thus, we could not simply apply the local Newton-CG procedure for optimization. The added globalization techniques, the Armijo line search strategy or the trust region method, become necessary.

The desired final state y_d is constructed as follows: The control u_d is chosen as

$$u_d(t) := \begin{pmatrix} -30 + 15 \cdot t \\ -10 + 10 \cdot t \\ -5 + 10 \cdot t \\ -10 + 10 \cdot t \end{pmatrix} \quad \text{for } t \in [0, T].$$

We denote by y_d^h the associated FE state. Then, the final state y_d is defined by $y_d := y_d^h(T)$. Hence, excluding control costs the desired final state could be reached (numerically). Notice that the second ('$x_1 = 1$') and the fourth ('$x_1 = 0$') component of u_d are equal. As a consequence, the function y_d is axially symmetric with respect to the vertical plane at $x_1 = 0.5$. It is visualized in the left plot in Figure 6.24.

For the LSNCG procedure we choose

$$\boxed{\tau_a := 10^{-4},\ \tau_r := 3 \cdot 10^{-3},\ \alpha := 10^{-1}.}$$

Step (1):
In Example III the constant α for the Armijo condition is given by 10^{-2}. Here, we increase it to $\alpha = 10^{-1}$. This means that a larger decrease in the values of the reduced cost functional is demanded. This has especially an effect on the iterations within which negative curvature is detected.

In the LSNCG-FE run negative curvature comes directly up in the first iteration. The FE optimization is summarized in Table 6.25. The first number of CG iterations is written in red since negative curvature was detected. Hence, the Hessian $\hat{J}''(0)$ is not positive definite. In fact, negative curvature occurs at the beginning of the second CG iteration ($j = 2$). Consequently, the output $d^{(0)}$ is a scalar multiple of the steepest descent direction $-\hat{J}'(0)$; see page 31. If we use $\alpha = 10^{-2}$ for the Armijo algorithm, the step

k	$\hat{J}(u^{(k)})$	$\|\hat{J}'(u^{(k)})\|_{U_h}$	η_k	#CG it	$\|d^{(k)}\|_{U_h}$	t_k
0	3.70810	$3.5 \cdot 10^{-1}$	0.0100	1	22.96	0.5
1	1.34616	$6.9 \cdot 10^{-2}$	0.0020	6	4.79	1
2	1.21561	$1.0 \cdot 10^{-2}$	0.0003	8	0.63	1
3	1.21268	$1.6 \cdot 10^{-4}$				

Table 6.25: Example IV: Iterations of the LSNCG-FE run.

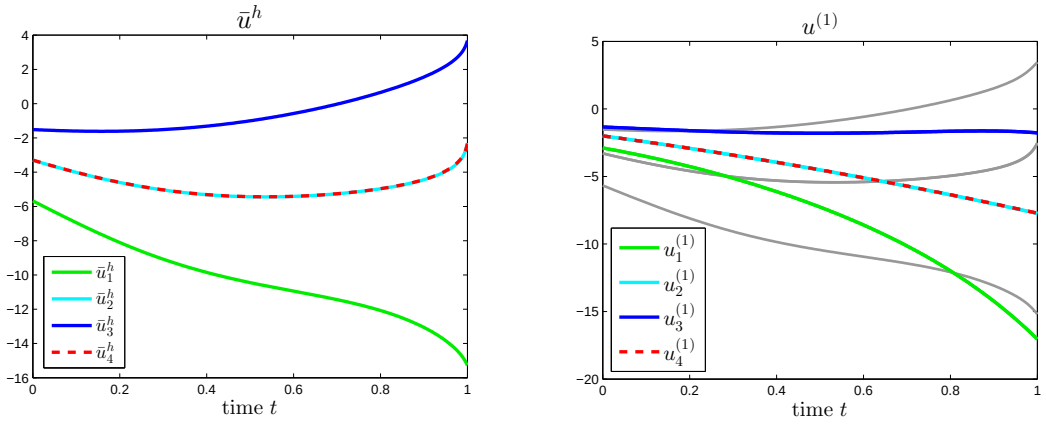


Figure 6.25: Example IV: optimal FEcontrol $\bar{u}^h$ (left) and second control iterate $u^{(1)}$ of TR-POD run 4.1 (right).

length $t_0 = 1$ is accepted throughout the iterations and in particular for the first step. This results in a second reduced cost functional value larger than 3 and the LSNCG-FE optimization needs two iterations more to find the optimal control compared to the listed run. Thus, the change of the constant α pays off. The CPU time for the LSNCG-FE run is about 32.06 seconds. Similar to Example III, changing η_a to 1 or 2 and/or η_b to 10^{-1} does not result in a different number of total iterations and the differences in number of CG iterations are not larger than 2.

Of course, the FE state y^h associated with the initial control $u^{(0)} = 0$ is the same as in Example III (Figure 6.14). Without controlling the value of the reduced cost functional is given by $\hat{J}(0) = J_1(y^h) = 3.70810$. The optimal value $\hat{J}(\bar{u}^h)$ is less than one third of the uncontrolled one. The distance of the final optimal FE state $\bar{y}^h(T)$ from the desired final state y_d is of order $J_1(\bar{y}^h) = 0.42398$. The final optimal FE state is presented in the right plot in Figure 6.24. The optimal FE control $\bar{u}^h$ is shown in the left plot in Figure 6.25. The components $\bar{u}_2^h$ and $\bar{u}_4^h$ are visually identical. This is caused by $u_{d,2} = u_{d,4}$. Likewise, symmetry is transferred to the final optimal FE state $\bar{y}^h(T)$.

Step (2):
For the LSNCG-POD strategy we utilize the controls $u_{bc} = 0$ and $u_{bc} = \bar{u}^h$ for the POD basis computation. We compare the H-norm and the V-norm implementation and use the two variants 'basis B_{SA}' and 'basis B_S'.

We start by choosing $u_{bc} = 0$ and 'basis B_{SA}'. The absolute and relative errors in the

	$X = H$			$X = V$		
ℓ	$e^u_{abs}(\ell)$	$e^u_{rel}(\ell)$	$e^y_{abs}(\ell)$	$e^u_{abs}(\ell)$	$e^u_{rel}(\ell)$	$e^y_{abs}(\ell)$
1	$7.5 \cdot 10^{+0}$	$5.9 \cdot 10^{-1}$	$5.9 \cdot 10^{-1}$	$7.5 \cdot 10^{+0}$	$5.9 \cdot 10^{-1}$	$5.9 \cdot 10^{-1}$
5	$1.4 \cdot 10^{+0}$	$1.1 \cdot 10^{-1}$	$1.9 \cdot 10^{-1}$	$1.4 \cdot 10^{+0}$	$1.1 \cdot 10^{-1}$	$2.0 \cdot 10^{-1}$
9	$7.9 \cdot 10^{-1}$	$6.3 \cdot 10^{-2}$	$1.3 \cdot 10^{-1}$	$1.2 \cdot 10^{+0}$	$9.3 \cdot 10^{-2}$	$1.7 \cdot 10^{-1}$
11	$4.5 \cdot 10^{-1}$	$3.6 \cdot 10^{-2}$	$1.1 \cdot 10^{-1}$	$7.1 \cdot 10^{-1}$	$5.6 \cdot 10^{-2}$	$1.3 \cdot 10^{-1}$
20	$1.6 \cdot 10^{-1}$	$1.3 \cdot 10^{-2}$	$6.6 \cdot 10^{-2}$	$1.5 \cdot 10^{-1}$	$1.2 \cdot 10^{-2}$	$6.3 \cdot 10^{-2}$
50	$1.3 \cdot 10^{-1}$	$1.0 \cdot 10^{-2}$	$5.3 \cdot 10^{-2}$	$1.3 \cdot 10^{-1}$	$1.0 \cdot 10^{-2}$	$5.3 \cdot 10^{-2}$
60	$\mathbf{8.6 \cdot 10^{-2}}$	$6.8 \cdot 10^{-3}$	$4.3 \cdot 10^{-2}$	$\mathbf{7.1 \cdot 10^{-2}}$	$5.6 \cdot 10^{-3}$	$3.8 \cdot 10^{-2}$

Table 6.26: Example IV, LSNCG-POD, 'basis B_{SA}', $u_{bc} = 0$: Absolute & relative errors in the control variable and absolute errors in the state variable for $X = H$ and $X = V$ for different ℓ .

control variable as well as the absolute errors in the state variable for selected numbers ℓ are named in Table 6.26. In contrast to Example II ($X = H$), the absolute error in the control variable drops beneath 10^{-1} when increasing the POD basis rank ℓ. But similarly, the absolute error in the state variable stays above 10^{-2}. The differences for the V-norm implementation compared to the H-norm implementation are little. For few POD basis functions the H-norm implementation yields sometimes a bit smaller errors ($\ell = 9, 11$). This is also the case for the other choices, $u_{bc} = \bar{u}^h$ with 'basis B_S' or 'basis B_{SA}', as it can be seen in the following.

We continue with $u_{bc} = \bar{u}^h$ while keeping 'basis B_{SA}'. The results are shown in Table 6.27. For $X = H$ the absolute error $e^u_{abs}(7)$ equals $9.2 \cdot 10^{-2}$ and $e^y_{abs}(7)$ is given by $2.3 \cdot 10^{-3}$. The respective errors for the V-norm implementation are slightly larger with $e^u_{abs}(7) = 1.1 \cdot 10^{-1}$. In both cases, $X = H$ and $X = V$, the number $\ell = 7$ of POD basis functions is the smallest number which yields a suboptimal control $\bar{u}^\ell$ for which $\Delta J(\bar{u}^\ell) < 0$ is satisfied. This means that the FE minimal value $\hat{J}(\bar{u}^h)$ is undermatched in both cases.

In a last step we only take state snapshots for the POD basis computation when utilizing $u_{bc} = \bar{u}^h$. The results are shown in Table 6.28. Compared to Example III the errors decrease more rapidly. For $\ell = 21$ we have $e^u_{abs}(\ell) = 7.1 \cdot 10^{-2}$ and $e^y_{abs}(\ell) = 2.5 \cdot 10^{-3}$. This let us hope that the Carter condition in the TR-POD approach is satisfiable when only utilizing state snapshots. But for $\ell = 60$ the ratio $\zeta^{(0)}_\ell$ still equals 0.363.

Decay of POD eigenvalues: The normalized eigenvalues computed from $u_{bc} = \bar{u}^h$ and 'basis B_{SA}' for $X = H$ and $X = V$ are visualized in the left plot in Figure 6.26. The curves in gray are obtained by 'basis B_S' and $u_{bc} = \bar{u}^h$. The right plot in this figure shows the eigenvalues resulting from $u_{bc} = 0$ and 'basis B_{SA}'. As it can be seen here, we observe the following **in all our numerical experiments**: The use of $X = V$, i.e. W = M + A, results in greater eigenvalues and a slower decay rate than the use of $X = H$, i.e. W = M. Independently of the choice of X the eigenvalues computed from $u_{bc} = 0$ decay more rapidly than those computed from $u_{bc} = \bar{u}^h$. The enrichment of the adjoint snapshots to the state snapshot ensemble results in a slower decay rate.

	$X = H$			$X = V$		
ℓ	$e^u_{abs}(\ell)$	$e^u_{rel}(\ell)$	$e^y_{abs}(\ell)$	$e^u_{abs}(\ell)$	$e^u_{rel}(\ell)$	$e^y_{abs}(\ell)$
1	$1.1 \cdot 10^{+1}$	$9.1 \cdot 10^{-1}$	$4.3 \cdot 10^{-1}$	$1.3 \cdot 10^{+1}$	$1.0 \cdot 10^{+0}$	$6.0 \cdot 10^{-1}$
3	$1.3 \cdot 10^{+0}$	$1.0 \cdot 10^{-1}$	$4.8 \cdot 10^{-2}$	$1.5 \cdot 10^{+0}$	$1.2 \cdot 10^{-1}$	$5.8 \cdot 10^{-2}$
5	$2.3 \cdot 10^{-1}$	$1.8 \cdot 10^{-2}$	$5.6 \cdot 10^{-3}$	$2.6 \cdot 10^{-1}$	$2.1 \cdot 10^{-2}$	$5.8 \cdot 10^{-3}$
6	$1.7 \cdot 10^{-1}$	$1.4 \cdot 10^{-2}$	$4.3 \cdot 10^{-3}$	$1.7 \cdot 10^{-1}$	$1.4 \cdot 10^{-2}$	$4.3 \cdot 10^{-3}$
7	$\mathbf{9.2 \cdot 10^{-2}}$	$7.3 \cdot 10^{-3}$	$2.3 \cdot 10^{-3}$	$1.1 \cdot 10^{-1}$	$8.7 \cdot 10^{-3}$	$2.9 \cdot 10^{-3}$
10	$\mathbf{1.9 \cdot 10^{-2}}$	$1.5 \cdot 10^{-3}$	$4.0 \cdot 10^{-4}$	$\mathbf{2.5 \cdot 10^{-2}}$	$2.0 \cdot 10^{-3}$	$6.7 \cdot 10^{-4}$
12	$\mathbf{1.4 \cdot 10^{-2}}$	$1.1 \cdot 10^{-3}$	$2.0 \cdot 10^{-4}$	$\mathbf{1.9 \cdot 10^{-2}}$	$1.5 \cdot 10^{-3}$	$4.9 \cdot 10^{-4}$
13	$\mathbf{7.5 \cdot 10^{-3}}$	$6.0 \cdot 10^{-4}$	$1.2 \cdot 10^{-4}$	$\mathbf{8.1 \cdot 10^{-3}}$	$6.5 \cdot 10^{-4}$	$4.9 \cdot 10^{-4}$
19	$\mathbf{2.8 \cdot 10^{-3}}$	$2.2 \cdot 10^{-4}$	$3.4 \cdot 10^{-5}$	$\mathbf{2.6 \cdot 10^{-3}}$	$2.0 \cdot 10^{-4}$	$3.6 \cdot 10^{-5}$
20	$\mathbf{9.4 \cdot 10^{-4}}$	$7.4 \cdot 10^{-5}$	$1.4 \cdot 10^{-5}$	$\mathbf{1.5 \cdot 10^{-3}}$	$1.2 \cdot 10^{-4}$	$2.6 \cdot 10^{-5}$
60	$\mathbf{9.1 \cdot 10^{-6}}$	$7.2 \cdot 10^{-7}$	$3.5 \cdot 10^{-7}$	$\mathbf{7.4 \cdot 10^{-6}}$	$5.9 \cdot 10^{-7}$	$4.0 \cdot 10^{-7}$

Table 6.27: Example IV, LSNCG-POD, 'basis B_{SA}', $u_{bc} = \bar{u}^h$: Absolute & relative errors in the control variable and absolute errors in the state variable for $X = H$ and $X = V$ for different ℓ.

	$X = H$			$X = V$		
ℓ	$e^u_{abs}(\ell)$	$e^u_{rel}(\ell)$	$e^y_{abs}(\ell)$	$e^u_{abs}(\ell)$	$e^u_{rel}(\ell)$	$e^y_{abs}(\ell)$
1	$1.1 \cdot 10^{+1}$	$9.1 \cdot 10^{-1}$	$4.3 \cdot 10^{-1}$	$1.3 \cdot 10^{+1}$	$1.0 \cdot 10^{+0}$	$6.0 \cdot 10^{-1}$
5	$1.2 \cdot 10^{+0}$	$9.7 \cdot 10^{-2}$	$6.6 \cdot 10^{-2}$	$1.4 \cdot 10^{+0}$	$1.1 \cdot 10^{-1}$	$7.6 \cdot 10^{-2}$
7	$7.9 \cdot 10^{-1}$	$6.3 \cdot 10^{-2}$	$2.4 \cdot 10^{-2}$	$8.5 \cdot 10^{-1}$	$6.8 \cdot 10^{-2}$	$2.7 \cdot 10^{-2}$
20	$1.4 \cdot 10^{-1}$	$1.1 \cdot 10^{-2}$	$4.7 \cdot 10^{-3}$	$2.0 \cdot 10^{-1}$	$1.6 \cdot 10^{-2}$	$6.0 \cdot 10^{-3}$
21	$\mathbf{7.1 \cdot 10^{-2}}$	$5.6 \cdot 10^{-3}$	$2.5 \cdot 10^{-3}$	$\mathbf{8.3 \cdot 10^{-2}}$	$6.6 \cdot 10^{-3}$	$2.6 \cdot 10^{-3}$
39	$\mathbf{8.4 \cdot 10^{-3}}$	$6.7 \cdot 10^{-4}$	$4.4 \cdot 10^{-4}$	$\mathbf{8.3 \cdot 10^{-3}}$	$6.6 \cdot 10^{-4}$	$4.1 \cdot 10^{-4}$
40	$\mathbf{6.0 \cdot 10^{-3}}$	$4.8 \cdot 10^{-4}$	$3.2 \cdot 10^{-4}$	$\mathbf{8.2 \cdot 10^{-3}}$	$6.5 \cdot 10^{-4}$	$4.0 \cdot 10^{-4}$
60	$\mathbf{3.1 \cdot 10^{-3}}$	$2.5 \cdot 10^{-4}$	$1.8 \cdot 10^{-4}$	$\mathbf{4.4 \cdot 10^{-3}}$	$3.5 \cdot 10^{-4}$	$2.7 \cdot 10^{-4}$

Table 6.28: Example IV, LSNCG-POD, 'basis B_S', $u_{bc} = \bar{u}^h$: Absolute & relative errors in the control variable and absolute errors in the state variable for $X = H$ and $X = V$ for different ℓ.

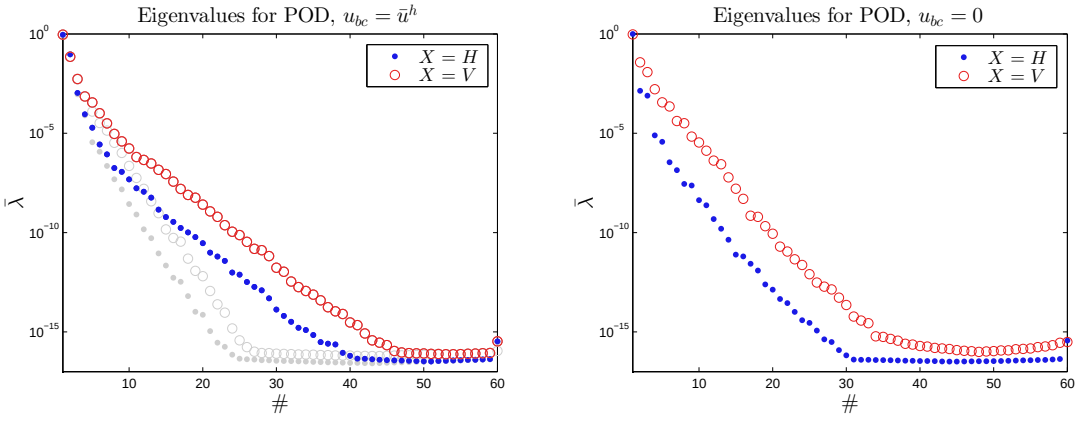


Figure 6.26: Example IV: Decay of normalized eigenvalues computed from $u_{bc} = \bar{u}^h$ (left) and from $u_{bc} = 0$ (right) using 'basis B_{SA}' for $X = H$ and $X = V$.

	CPU time	
LSNCG-FE optimization	32.06	
CG algorithm	23.04	(3.03, 8.63, 11.38)
Snapshot generation	1.89	
- state	1.18	
- adjoint state	0.71	
POD basis ($\ell = 15$)	0.22	
LSNCG-POD optimization	3.80	
CG algorithm	2.11	(0.30, 0.84, 0.97)

Table 6.29: Example IV: CPU times measured in seconds of LSNCG-POD run 4.1 compared to LSNCG-FE optimization (in brackets: CPU time needed by the CG algorithm in iteration $k = 0, 1, 2$).

Computational efficiency: We pick one LSNCG-POD run to illustrate the computational speedup which can be achieved. For **LSNCG-POD run 4.1** we choose $\ell = 7$, $u_{bc} = \bar{u}^h$, 'basis B_{SA}' and $X = H$. The reader is asked to look up the errors of the corresponding suboptimal control and state in Table 6.27. The computing times are summarized in Table 6.29. The overall CPU time, including snapshot generation, POD basis computation and LSNCG-POD optimization, is 5.69 seconds. Hence, the computation of the suboptimal control $\bar{u}^\ell$ is 5.6 times as fast as the computation of the FE optimal control $\bar{u}^h$. The speedup factor for the CG procedure, considered separately, is again impressive. On average it is given by almost 11. Note that the needed CPU time for the V-norm implementation is practically identical.

Step (3):
For the trust region strategy we define

$$\ell_{max} := 15, \quad \Delta^{(0)} = 11.5, \quad \eta_1 := 0.1, \ \eta_2 := 0.8,$$
$$\gamma_1 := 0.5, \ \gamma_2 := 1, \ \gamma_3 := 1.2.$$

k	$\hat{J}(u^{(k)})$	$\|\hat{J}'(u^{(k)})\|_{U_h}$	#CG it	$\|d_t^{(k)}\|_{U_h}$	$\Delta^{(k)}$	$\rho^{(k)}(d_t^{(k)})$	ℓ	$\zeta_\ell^{(k)}$
0	3.70810	$3.5 \cdot 10^{-1}$	(1)	23.0	23.0	0.06	3	0.035
–	–	–	–	11.5	11.5	0.78	–	–
1	1.33962	$6.8 \cdot 10^{-2}$	6	4.66	11.5	0.83	4	0.129
2	1.21799	$1.3 \cdot 10^{-2}$	8	0.84	13.8	1.05	6	0.130
3	1.21280	$1.8 \cdot 10^{-3}$	9	0.15	16.6	1.04	9	0.126
4	1.21267	$2.4 \cdot 10^{-4}$						

Table 6.30: Example IV: Iterations of TR-POD run 4.1.

The initial trust region radius is chosen by $\Delta^{(0)} = 11.5$ because the first LSNCG-FE step is of length $0.5 \cdot 22.96 \approx 11.5$ (Table 6.25). The value 22.96 is the U_h-norm of the FE based descent direction obtained from the CG algorithm. Remember that it is a scalar multiple of the negative gradient $-\hat{J}'(0)$.

TR-FE optimization: In the first iteration the Cauchy point is successfully taken as trial point. The Cauchy step lies on the boundary of the trust region so that it is nearly identical to the first step in the LSNCG-FE run. Consequently, this holds also for the following iterates. The required CPU time for TR-FE optimization is 38.38 seconds. Hence, it takes about 6 seconds more than the LSNCG-FE procedure. The ratios of actual and predicted reduction are given by $\rho^{(0)}(d_t^{(0)}) = 0.79$, $\rho^{(1)}(d_t^{(1)}) = 0.89$ and $\rho^{(2)}(d_t^{(2)}) = 1.04$. The associated step lengths equal 11.5, 4.78 and 0.61.

For the POD method we use $X = H$ and 'basis B_{SA}'. In **TR-POD run 4.1** we start the optimization with $\ell_0 = 3$ POD basis functions, as done in TR-POD run 3.1. At first, in order to test the trust region globalization of our algorithm we set $\Delta^{(0)} = 23$. The opimization run is summarized in Table 6.30. As intended, the initial trust region radius is too large and the first trial step (Cauchy step on the trust region boundary) is unsuccessful with $\rho^{(0)}(d_t^{(0)}) = 0.06$. Therefore, the trust region radius is divided in half ($\gamma_1 = 0.5$) and we obtain the desired radius $\Delta^{(0)} = 11.5$ with $\rho^{(0)}(d_t^{(0)}) = 0.78$. In case of such an unsuccessful step the POD reduced-order models are kept. When refering to TR-POD run 4.1 we assume that the initial radius was directly chosen by $\Delta^{(0)} = 11.5$, as fixed above. Recall that the number of CG iterations in the first line of Table 6.30 is set in brackets since the Cauchy step has to be taken as trial step. The ratios $\rho^{(k)}(d_t^{(k)})$ in iteration $k = 2$ and $k = 3$ are slightly larger than one. But we notice that TR-POD run 4.1 needs one iteration more than the TR-FE optimization process. The control iterate $u^{(2)}$ differs only $\|u^{(2)} - \bar{u}^{TR}\|_{U_h} = 0.84$ from the optimal TR-POD control $\bar{u}^{TR}$. The following shows that it might sometimes be better to choose some more POD basis functions than determined through the Carter condition. We realize two modifications that reduce the number of iterations to that of the TR-FE run:

(1) The Carter condition in iteration $k = 2$ of TR-POD run 4.1 is satisfied for $\ell = 6$ POD basis functions. We increase this number manually to $\ell = 9$ in **TR-POD run 4.2**. The last optimization step is presented in Table 6.31.

(2) Before testing the Carter condition in iteration k we determine the number $\tilde{\ell}_0$ of POD

k	$\hat{J}(u^{(k)})$	$\|\hat{J}'(u^{(k)})\|_{U_h}$	#CG it	$\|d_t^{(k)}\|_{U_h}$	$\Delta^{(k)}$	$\rho^{(k)}(d_t^{(k)})$	ℓ	$\zeta_\ell^{(k)}$
2	1.21799	$1.3 \cdot 10^{-2}$	8	0.84	13.8	1.05	9	0.021
3	1.21270	$5.1 \cdot 10^{-4}$						

Table 6.31: Example IV: Last optimization step of TR-POD run 4.2.

k	$\hat{J}(u^{(k)})$	$\|\hat{J}'(u^{(k)})\|_{U_h}$	#CG it	$\|d_t^{(k)}\|_{U_h}$	$\Delta^{(k)}$	$\rho^{(k)}(d_t^{(k)})$	ℓ	$\zeta_\ell^{(k)}$
1	1.33962	$6.8 \cdot 10^{-2}$	6	4.60	11.5	0.87	5	0.076
2	1.21689	$1.3 \cdot 10^{-2}$	7	0.74	13.8	1.08	7	0.094
3	1.21260	$1.0 \cdot 10^{-3}$						

Table 6.32: Example IV: Last optimization steps of TR-POD run 4.3.

basis functions as the minimal number which satisfies $1 - \mathcal{E}(\tilde{\ell}_0) < \|\hat{J}'(u^{(k)})\|_{U_h} \cdot 10^{-4}$. Then, we set $\ell_0 := \min\{\max\{\ell_0, \tilde{\ell}_0\}, \ell_{max}\}$. This can be viewed as attempt to estimate the required or rather recommended number of POD basis functions depending on the energy contained in the snapshot ensemble and on the current gradient norm. The resulting **TR-POD run 4.3** is summarized in Table 6.32. We have omitted the first iteration since it is equal to that of TR-POD run 4.1. The numbers $\tilde{\ell}_0$ are given by $3, 5$ and 7. For these numbers the Carter condition is directly satisfied. Recall that the number $\ell = 7$ is used in LSNCG-POD run 3.1, where the optimal POD basis was taken for optimization.

Computational effort: For completeness we just name the required overall CPU times for the TR-POD runs 4.2 and 4.3. The first run takes 11.80 seconds and the latter one needs 12.55 seconds. In TR-POD run 4.2 the Carter condition requires one redundant setup of a POD Galerkin approximation of the gradient. This yields a small but additional time factor. On average over the two runs, the computational effort is about 2.6 times smaller than that of the LSNCG-FE run and about 3.2 times smaller than that of the TR-FE procedure. About 60 per cent of the CPU time is needed for the snapshot generation, analogous to TR-POD run 3.1 from Example III. Below, we show that the POD basis computed from $u^{(1)}$ of TR-POD run 4.1 can be kept for terminating the optimization.

Decay of POD eigenvalues: We take a look at the change in the decay rate of the POD eigenvalues which arises from the first step in TR-POD run 4.1 (equal to that of the other two runs). In Figure 6.27 we visualize the normalized POD eigenvalues computed from $u_{bc} = u^{(0)} = 0$, from $u_{bc} = u^{(1)}$ and from $u_{bc} = \bar{u}^h$. Of course, the use of the uncontrolled snapshots in iteration $k = 0$ yields the most rapid decay. From the initial to the second iteration there is a visible improvement in the sense of getting a slower decay. And the eigenvalues computed from $u^{(1)}$ are close to those obtained from $u_{bc} = \bar{u}^h$.

Shape of POD basis functions: We visualize the first four POD basis functions computed from the three controls $u_{bc} = u^{(0)} = 0$, $u_{bc} = u^{(1)}$ and $u_{bc} = \bar{u}^h$. The control $u^{(1)}$ is visualized in the right plot in Figure 6.25. The deviation of this control iterate from the

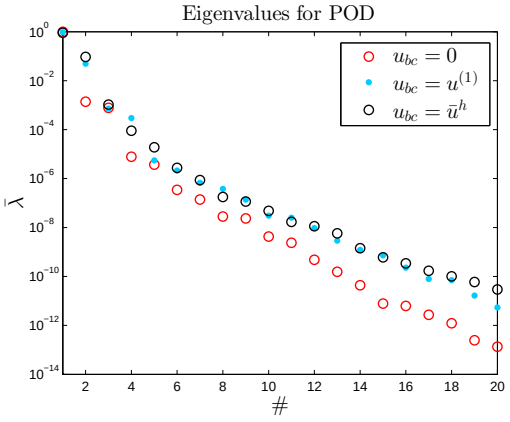


Figure 6.27: Example IV: Decay of normalized eigenvalues for the POD basis computed from $u_{bc} = 0$, from $u_{bc} = u^{(1)}$ in TR-POD run 4.1 and from $u_{bc} = \bar{u}^h$ ($X = H$, 'basis B_{SA}').

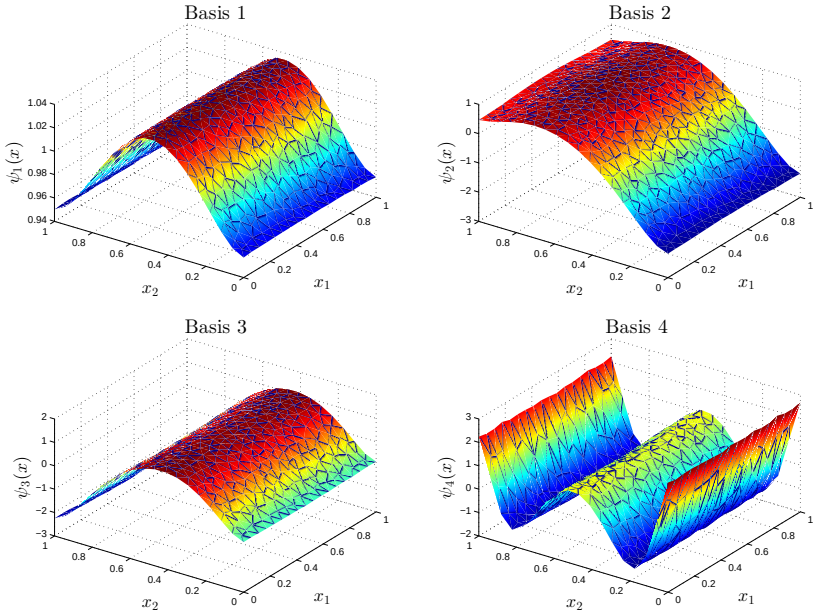


Figure 6.28: Example IV: First four POD basis functions computed from $u_{bc} = 0$ with $X = H$ and 'basis B_{SA}'.

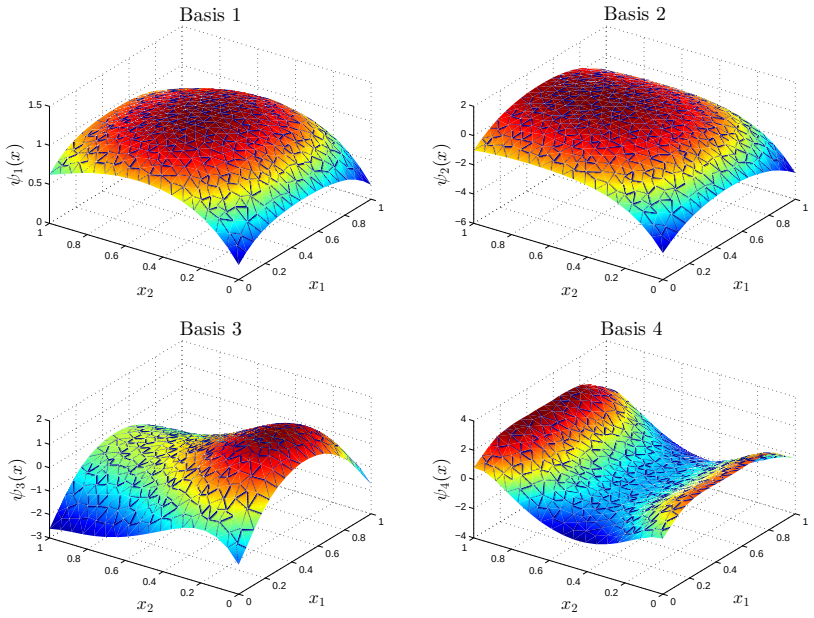


Figure 6.29: Example IV: First four POD basis functions computed from $u_{bc} = u^{(1)}$ in TR-POD run 4.1.

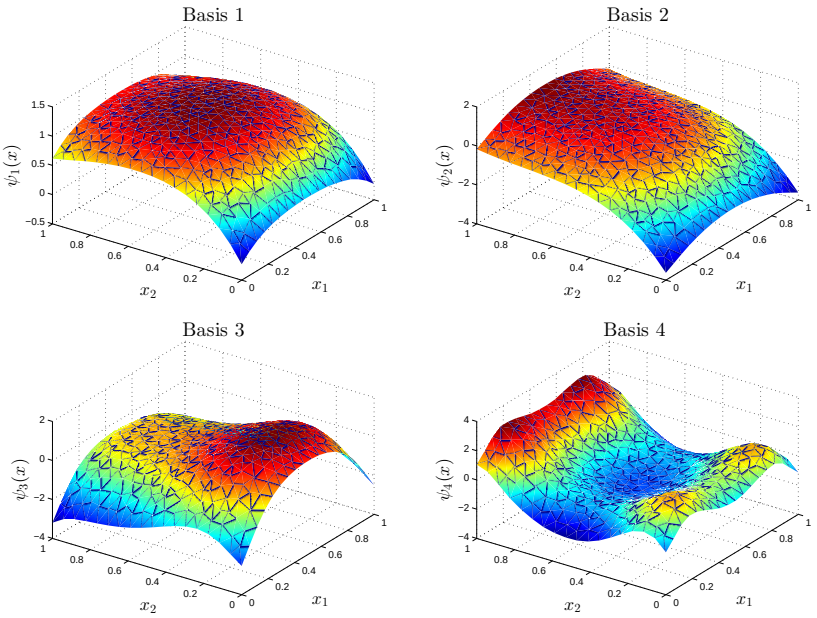


Figure 6.30: Example IV: First four POD basis functions computed from $u_{bc} = \bar{u}^h$ with $X = H$ and 'basis B_{SA}'.

k	$\hat{J}(u^{(k)})$	$\|\hat{J}'(u^{(k)})\|_{U_h}$	#CG it	$\|d_t^{(k)}\|_{U_h}$	$\Delta^{(k)}$	$\rho^{(k)}(d_t^{(k)})$	ℓ	$\zeta_\ell^{(k)}$
2	1.21799	$1.3 \cdot 10^{-2}$	8	0.82	13.8	1.18	10	0.082
3	1.21269	$9.1 \cdot 10^{-4}$						

Table 6.33: Example IV: Last optimization step of TR-POD run 4.4.

optimal FE control $\bar{u}^h$ is given by $\varepsilon^u_{abs}(u^{(1)}) = 4.14$ and by $\varepsilon^u_{rel}(u^{(1)}) = 0.33$. Figure 6.28 shows the basis functions corresponding to $u_{bc} = 0$. It is interesting to compare these basis functions to those from Example III (Figure 6.20). Note that the uncontrolled state snapshots are the same in both examples. As a result the first POD basis function here is optically identical to that in Figure 6.20 and the fourth basis functions are also very similar. The second basis function here is related to the uncontrolled adjoint state. In Example III the third POD basis function is the first which represents the dynamics of the uncontrolled adjoint state. In Example III (TR-POD run 3.1), $\ell = 3$ basis functions are needed to fulfill the Carter condition. Here, the Carter condition is already satisfied for $\ell = 2$ with $\zeta_2^{(0)} = 0.157$. However, we chose $\ell_0 = 3$ to obtain a better second control iterate $u^{(1)}$ with the aim of keeping the corresponding POD basis for the last optimization step. The first four POD basis functions computed from $u^{(1)}$ are shown in Figure 6.29. We observe that a siginificant change in the shape of the first four POD basis functions results from this first TR-POD step. The latter basis functions are very similar to the optimal ones. The first four optimal POD basis functions, computed from $u_{bc} = \bar{u}^h$, can be seen in Figure 6.30. The differences are a little larger compared to Example III. We do not visualize the first four POD basis functions corresponding to $u_{bc} = u^{(2)}$ of one of the above TR-POD runs. They are optically indistinguishable from the first four optimal POD basis functions.

The next TR-POD run, **TR-POD run 4.4**, shows that the POD basis obtained from $u_{bc} = u^{(1)}$ is sufficiently good to continue and terminate the POD optimization. We modify TR-POD run 4.1 by keeping the POD basis computed from $u^{(1)}$. In the subsequent itera-tion ($k = 2$) we manually set $\ell = 10$. The Carter condition is already satisfied for $\ell = 7$. But this choice results in an additional iteration (with $\ell = 13$ for the step $d_t^{(2)}$, $\|\hat{J}'(u^{(2)})\|_{U_h} = 2.1 \cdot 10^{-3}$). The last optimization step of TR-POD run 4.4 is summarized in Table 6.34. If the POD basis is analogously kept in TR-POD run 4.3, we also have to take $\ell = 10$ POD basis functions in iteration $k = 2$ in order to find the optimal TR-POD control with the following step. If the initial number ℓ_0 in TR-POD run 4.4 is replaced by $\ell_0 = 2$, which yields $\ell = 4$ in iteration $k = 1$, we finally have to take $\ell = 11$ POD basis functions so that the optimization terminates with the fourth control iterate $u^{(3)}$.

In TR-POD run 3.2 the second control iterate is obtained by a POD based inexact Newton step. In TR-POD run 4.4 it results from a POD based scaled gradient step (Cauchy step) which is computed with only $\ell = 3$ POD basis functions. And this latter step, as well as that in TR-POD run 3.2, yields a satisfying initializing control $u^{(1)}$ (if interpreted as that) for POD optimization based on a Newton-CG procedure. This is very interesting in comparison with the application of OS-POD in a gradient step to improve the initializing control; see [16].

k	$\hat{J}(u^{(k)})$	$\|\hat{J}'(u^{(k)})\|_{U_h}$	$\|d_t^{(k)}\|_{U_h}$	$\Delta^{(k)}$	$\rho^{(k)}(d_t^{(k)})$	ℓ	$\zeta_\ell^{(k)}$
0	3.70810	$3.5 \cdot 10^{-1}$	11.5	11.5	0.78	3	0.035
1	1.33962	$6.8 \cdot 10^{-2}$	3.51	11.5	0.85	4	0.129
2	1.23452	$2.6 \cdot 10^{-2}$	1.42	13.8	1.02	10	0.175
3	1.21646	$1.1 \cdot 10^{-2}$	0.50	16.6	0.88	10	0.084
4	1.21397	$6.5 \cdot 10^{-3}$	0.23	19.9	1.21	10	0.156
5	1.21314	$3.5 \cdot 10^{-3}$	0.15	23.9	1.11	12	0.119
6	1.21285	$2.5 \cdot 10^{-3}$	0.08	28.6	1.27	12	0.174
7	1.21272	$1.4 \cdot 10^{-3}$	0.06	34.3	0.79	16	0.161
8	1.21269	$1.0 \cdot 10^{-3}$					

Table 6.34: Example IV: Iterations of TR-POD run 4.5.

Comparison of inexact Newton steps to simple Cauchy steps: So far, we did not compare the use of inexact Newton steps to the use of gradient steps. We investigate one optimization run, **TR-POD run 4.5**, where we take the Cauchy points throughout the iterations as trial points with $\ell_0 = 3$ and $\ell_{max} = 20$. The first step is the same as that in TR-POD run 4.4. The POD basis computed from $u^{(1)}$ is kept for further optimization steps. In iteration $k = 2$ we set $\ell = 10$. The run is summarized in Table 6.34. Starting from the second control iterate $u^{(1)}$, the use of inexact Newton steps in TR-POD run 4.4 results in less than one third total iterations compared to the use of Cauchy steps. If we let it to the Carter condition to determine the number of POD basis functions, we obtain the numbers $\ell = 3, 4, 6, 7, 10, 12, 12, 16$ with the same number of total iterations as in the listed run. If the POD basis is updated in each iteration of the considered optimization process it takes as well $k = 8$ iterations with POD basis ranks $\ell = 3, 4, 5, 7, 7, 9, 9, 9$ (the final number is equal to that of TR-POD run 4.1). Let us mention that the TR-FE procedure using the Cauchy points needs $k = 8$ iterations, too. Here, the Carter condition controls the number of POD basis functions in a very satisfactory manner. Only for the use of inexact Newton steps, when the Hessian approximation is more important, it can be better to use some more basis functions than determined through the Carter condition. In TR-POD run 4.4 the final POD basis rank is $\ell = 10$. But in TR-POD run 4.5 this final number is given by $\ell = 16$. Seemingly, we benefit from the faster local convergence of the inexact Newton method compared to the gradient algorithm in an additional aspect. Here, it does not only result in less total iterations but the better convergence also allows to run the optimization with fewer POD basis functions when a basis update is not performed in the last iterations. The main reason seems to lie in the fact that the optimal control is found from a next to last iterate which has a larger gradient norm. Notice that $\|\hat{J}'(u^{(2)})\|_{U_h}$ equals $1.3 \cdot 10^{-2}$ in TR-POD run 4.4, whereas $\|\hat{J}'(u^{(k)})\|_{U_h} < 5 \cdot 10^{-3}$ holds for $k \geq 5$ in TR-POD run 4.5.

6.2.3 Example V

Due to the very satisfactory results obtained in Example III and IV for the nonlinearity $N(y) = y^3$ we move on to investigate the nonlinear function $N(y) := -0.5y^3$. Except for this modification, the example coincides with Example III.

The first-order derivative $N'(y) = -1.5y^2$ is not monotone increasing as required in assumption (**A2**). This assumption is needed to show solvability of the involved parabolic equations. We briefly discuss their solvability for nonlinearities of type $N(y) = -\alpha y^3$, where $\alpha \in (0, 1]$. We begin with the state equation. It can occur that the solution to (SE) exists only on some finite time interval $[0, T^*]$ since its L^∞-norm becomes unbounded. This is referred to as blow-up. The time $T^* > 0$ depends on the given data, i.e. the PDE's right-hand sides including the initial condition. These results can be found in [37, Chapter II, Section 15]. We assume that the solution exists on the whole time interval $[0, T]$. Then, solvability of the equations (LSE), (AE_1) and (AE_2) can be shown by considering these as abstract evolution problems. For a detailed presentation of this approach we refer the reader to [11, pp. 512-520]. Here, we only sketch the principle ideas. Recall the symmetric and positive definite bilinear form a from Section 2.4. The variational formulations of the named equations involve the bilinear form $\tilde{a}(t, \cdot, \cdot) : V \times V \to \mathbb{R}$, $t \in (0, T)$ a.e., defined by

$$\tilde{a}(t, \varphi_1, \varphi_2) := a(\varphi_1, \varphi_2) - 3\alpha \int_\Omega y(t, \boldsymbol{x})^2 \varphi_1(\boldsymbol{x}) \varphi_2(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \quad \text{for } \varphi_1, \varphi_2 \in V.$$

The needed solvability is ensured if the above bilinear form satisfies the two inequalities

$$\begin{aligned} |\tilde{a}(t, \varphi_1, \varphi_2)| &\leq c_1 \|\varphi_1\|_V \|\varphi_2\|_V, \\ |\tilde{a}(t, \varphi, \varphi)| &\geq c_2 \|\varphi\|_V^2 - c_3 \|\varphi\|_H^2, \end{aligned}$$

for all $\varphi, \varphi_1, \varphi_2 \in V$ and $t \in (0, T)$ a.e., where c_1, $c_2 > 0$ and $c_3 \geq 0$ are constants which do not depend on time t. These inequalities can be shown by making use of $y \in L^\infty(Q)$.

Remark 6.1. Numerically, we tried to solve (SE) with nonlinearity $N(y) = -\alpha y^3$ for the constants $\alpha = 0.1, 0.2, \ldots, 1$. Our solver has no problems to handle the values $\alpha \leq 0.6$. But starting from $\alpha = 0.7$, the solution becomes critical because oscillations in the state occur at later times steps in the interval $[0, 1]$. The interesting thing is that the larger the constant α is chosen the earlier this happens. Depending on the fineness of the time grid, it occurs that the Newton method can no longer reach the stopping criterion in a later time step. Seemingly, the solutions actually blow up, i.e. cease to exist, at respective times.

We turn to the numerical solution of (**P**), via ($\hat{\mathbf{P}}$), with the above defined nonlinear function $N(y) = -0.5y^3$. The parameters for the LSNCG method are defined by

$$\boxed{\tau_a := 10^{-4},\ \tau_r := 8 \cdot 10^{-4},\ \alpha := 10^{-2}}$$

Step (1):
The LSNCG-FE run is summarized in Table 6.35. It takes $k = 5$ iterations to compute the optimal FE control $\bar{u}^h$. The required CPU time is 79.40 seconds. In contrast to

k	$\hat{J}(u^{(k)})$	$\|\hat{J}'(u^{(k)})\|_{U_h}$	η_k	#CG it	$\|d^{(k)}\|_{U_h}$	t_k
0	1.77870	$1.2 \cdot 10^{+0}$	0.01000	6	2.65	1
1	0.62253	$4.1 \cdot 10^{-1}$	0.00333	6	1.96	1
2	0.30243	$1.2 \cdot 10^{-1}$	0.00010	8	1.61	1
3	0.23745	$2.5 \cdot 10^{-2}$	0.00020	12	0.50	1
4	0.23378	$1.5 \cdot 10^{-3}$	0.00001	15	0.03	1
5	0.23377	$5.5 \cdot 10^{-6}$				

Table 6.35: Example V: Iterations of the LSNCG-FE run.

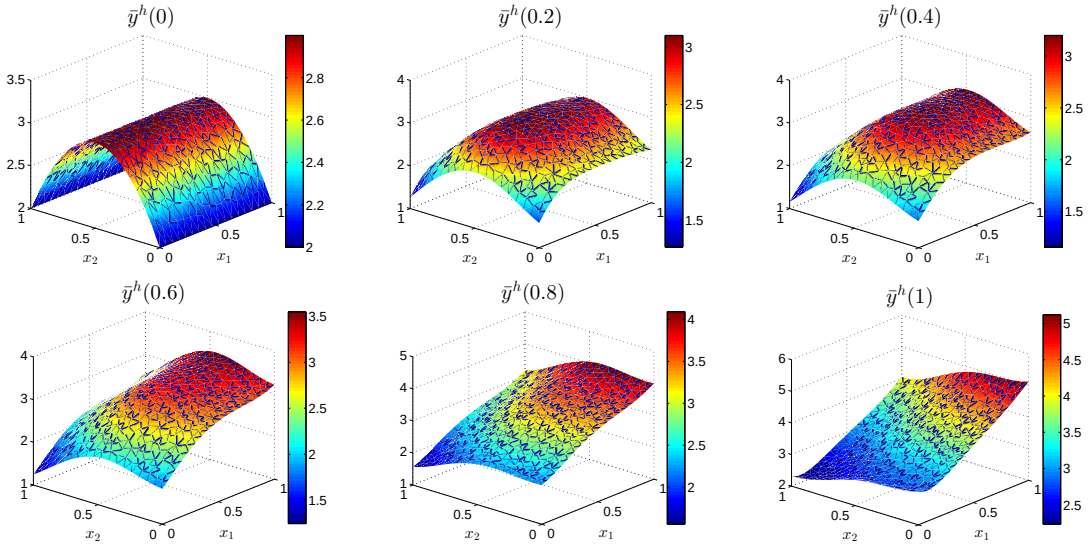


Figure 6.31: Example V: Optimal FE state $\bar{y}^h(t)$ at six different time steps $t = t_j$.

both previous examples, the norm of the initial gradient is here larger than 1. The initial reduced cost functional value is given by $\hat{J}(0) = 1.77870$. The LSNCG-FE optimization yields a minimal value of $\hat{J}(\bar{u}^h) = 0.23377$, where $J_1(\bar{y}^h)$ is given by 0.1300. Negative curvature is not encountered within the CG algorithm and the Armijo strategy accepts the step length 1 in all iterations (the same for $\alpha = 10^{-1}$). When we replace η_a by 1 or 2 and/or η_b by 10^{-1}, the number of iterations remains the same. Only the numbers of CG iterations differ a little.

The optimal FE state $\bar{y}^h$ is presented in Figure 6.31. The final optimal FE state $\bar{y}^h(T)$ is shown in the lower right plot in this figure. With only $\mathsf{k} = 4$ boundary segments we can not achieve a 'smooth kink' in this function, as in Example III. Recall that the desired final state can be seen in the right plot in Figure 6.11. We observe that the computed final state function is less curved than the one obtained for the nonlinearity $N(y) = y^3$ in Example III (lower right plot in Figure 6.15). In fact, it resembles more the final optimal FE state computed from the linear heat equation (see the left plot in Figure 6.13). The optimal FE adjoint state $\bar{p}^h$ is visualized in Figure 6.32. The state values lie between 1.6 and 5.1 whereas the values of the adjoint state are only in the range of -0.1 to 0.2. The optimal FE control $\bar{u}^h$ is shown in the left plot in Figure 6.33.

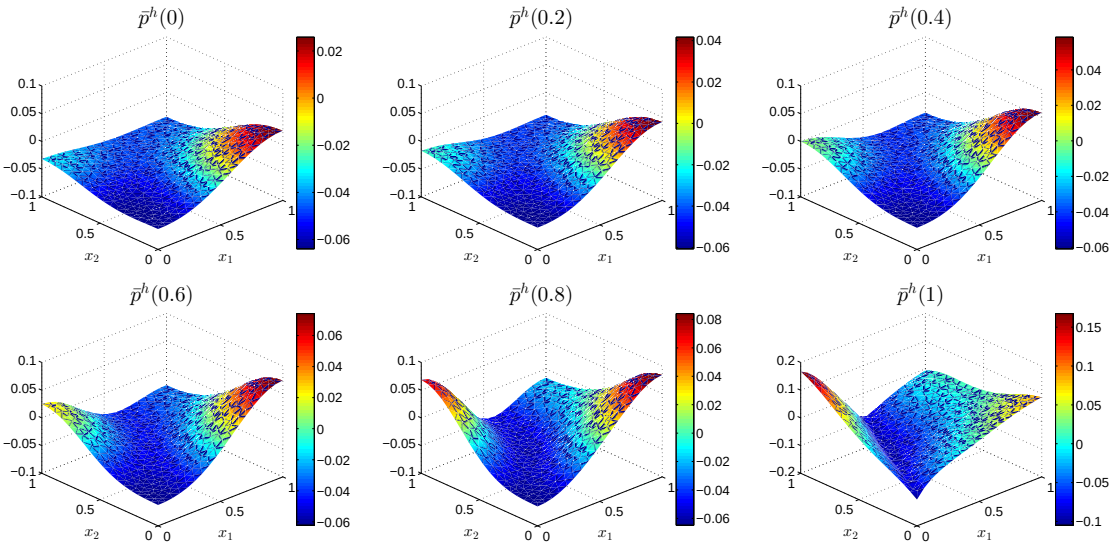


Figure 6.32: Example V: Optimal FE adjoint state $\bar{p}^h(t)$ at six different time steps $t = t_j$.

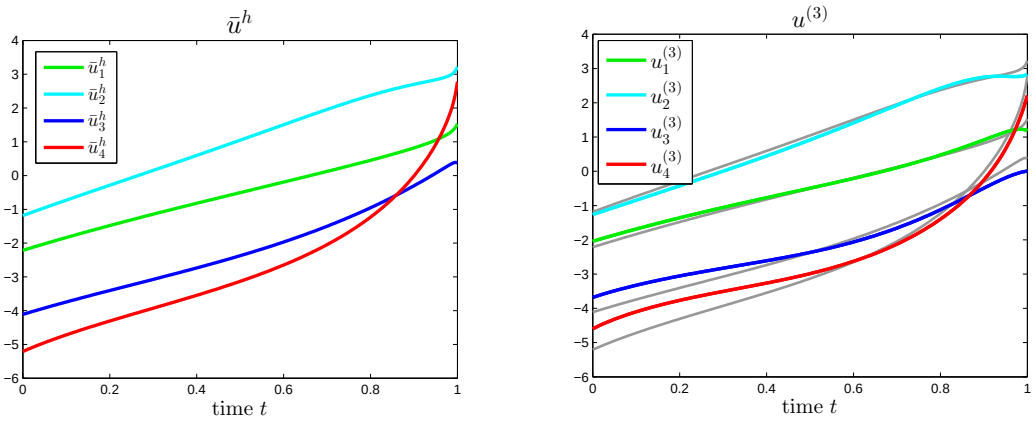


Figure 6.33: Example V: Optimal FE control $\bar{u}^h$ (left) and control iterate $u^{(3)}$ of TR-POD run 5.1 (right).

h_{max} (N_x)	Δt	$\varepsilon^u_{abs}(u^{ref})$	$\varepsilon^u_{rel}(u^{ref})$	$\varepsilon^y_{abs}(y^{ref})$	$\varepsilon^y_{rel}(y^{ref})$	$\Delta J(u^{ref})$
0.06 (498)	0.004	$6.9 \cdot 10^{-2}$	$1.5 \cdot 10^{-2}$	$3.6 \cdot 10^{-3}$	$1.2 \cdot 10^{-3}$	$1.3 \cdot 10^{-5}$

Table 6.36: Example V: Comparison of optimal LSNCG-FE solution $(\bar{u}^h, \bar{y}^h)$ with reference optimal FE solution (u^{ref}, y^{ref}).

We briefly compare the optimal LSNCG-FE state and control to the reference optimal FE state and control. The latter are computed by a gradient algorithm. The results are shown in Table 6.36. The relative values are very close to those from Example III and the (unrounded) absolute difference of the two states is also smaller than $h_{max}^2 \leq \Delta t$. The difference of the minimal reduced cost functional values is given by $1.3 \cdot 10^{-5}$. Let us already state here that in our computations in step (2) a suboptimal control $\bar{u}^\ell$ differs at most by $e_{abs}^u(\ell) = 7.5 \cdot 10^{-2}$ from the optimal FE control $\bar{u}^h$, if the difference $\Delta J(\bar{u}^\ell)$ is negative.

Step (2):
We turn to the LSNCG-POD approach. The differences between the H- and the V-norm implementation are only small. Hence, we again restrict ourselves to the choice of $X = H$ for comparability. For the POD basis computation we utilize the controls $u_{bc} = 0$, $u_{bc} = \bar{u}^h$ and the control iterate $u_{bc} = u^{(3)}$ of TR-POD run 5.1. The latter control can be seen in the right plot in Figure 6.33. We first disregard that this control is computed within a TR-POD procedure.

We begin by presenting the results for the two combinations $u_{bc} = \bar{u}^h$ with 'basis B_S' and $u_{bc} = 0$ with 'basis B_{SA}'. The absolute and relative errors in the control variable and the absolute errors in the state variable for different numbers ℓ are listed in Table 6.37. For larger POD basis ranks ℓ the errors in the control variable which result from $u_{bc} = \bar{u}^h$ with 'basis B_S' are smaller than those obtained from $u_{bc} = 0$ with 'basis B_{SA}'. Nonetheless, we need at least $\ell = 42$ POD basis functions in order to obtain good POD suboptimal controls. The errors in the state variable are throughout smaller for $u_{bc} = \bar{u}^h$ with 'basis B_S'. Here we again observe that for equally good or even worse suboptimal controls $\bar{u}^\ell$ the associated suboptimal states $\bar{y}^\ell$ are closer to the optimal FE state $\bar{y}^h$ when utilizing $u_{bc} = \bar{u}^h$. Let us just mention that $\varepsilon_{abs}^y(y(\bar{u}^\ell))$ equals $3.3 \cdot 10^{-3} > 2.8 \cdot 10^{-3} = e_{abs}^y(\ell)$ in case of $u_{bc} = \bar{u}^h$ and 'basis B_S' for $\ell = 42$, whereas the use of $u_{bc} = 0$ with 'basis B_{SA}' yields $\varepsilon_{abs}^y(y(\bar{u}^\ell)) = 1.2 \cdot 10^{-3} < 2.8 \cdot 10^{-2} = e_{abs}^y(\ell)$ for $\ell = 64$.

In Table 6.38 we compare the errors for the controls $u_{bc} = u^{(3)}$ (of TR-POD run 5.1) and $u_{bc} = \bar{u}^h$ when using 'basis B_{SA}'. The difference between these two controls is just of order $\varepsilon_{abs}^u(u^{(3)}) = 4.6 \cdot 10^{-1}$ or $\varepsilon_{rel}^u(u^{(3)}) = 1.0 \cdot 10^{-1}$, respectively. In Figure 6.33 it can be seen how little their components differ over time. But despite their similarity, the errors resulting from the optimal FE control $\bar{u}^h$ decrease significantly faster. For $u_{bc} = \bar{u}^h$ and $\ell = 11$ the suboptimal control $\bar{u}^\ell$ satisfies $e_{abs}^u(\ell) = 4.5 \cdot 10^{-2}$. In case of $u_{bc} = u^{(3)}$ the respective value is still of order $3.1 \cdot 10^{-1}$ and we have to take $\ell = 34$ POD basis functions so that $e_{abs}^u(\ell)$ is equally small.

Decay of POD eigenvalues: In Figure 6.34 we compare the decay of the normalized POD eigenvalues computed from the three above controls using 'basis B_{SA}'. As always, the eigenvalues associated with the uncontrolled snapshots, i.e. $u_{bc} = 0$, decay most rapidly. But we can see that the normalized eigenvalues computed from $u_{bc} = u^{(3)}$ are very close to those obtained from $u_{bc} = \bar{u}^h$. In fact, the latter ones do not decay clearly slower. This can be interpreted such that the corresponding POD basis functions model nearly equally much system energy (proportional to total energy).

Shape of POD basis functions: For comparison we first show the first four POD basis

	$u_{bc} = \bar{u}^h$, 'basis B_S'			$u_{bc} = 0$, 'basis B_{SA}'		
ℓ	$e^u_{abs}(\ell)$	$e^u_{rel}(\ell)$	$e^y_{abs}(\ell)$	$e^u_{abs}(\ell)$	$e^u_{rel}(\ell)$	$e^y_{abs}(\ell)$
1	$3.9 \cdot 10^{+0}$	$8.6 \cdot 10^{-1}$	$2.0 \cdot 10^{-1}$	$3.5 \cdot 10^{+0}$	$7.7 \cdot 10^{-1}$	$4.3 \cdot 10^{-1}$
5	$1.7 \cdot 10^{+0}$	$3.7 \cdot 10^{-1}$	$9.4 \cdot 10^{-2}$	$5.6 \cdot 10^{-1}$	$1.2 \cdot 10^{-1}$	$1.1 \cdot 10^{-1}$
15	$9.2 \cdot 10^{-1}$	$2.0 \cdot 10^{-2}$	$4.9 \cdot 10^{-2}$	$3.9 \cdot 10^{-1}$	$8.5 \cdot 10^{-2}$	$8.1 \cdot 10^{-2}$
21	$5.4 \cdot 10^{-1}$	$1.2 \cdot 10^{-2}$	$3.3 \cdot 10^{-2}$	$3.7 \cdot 10^{-1}$	$8.1 \cdot 10^{-2}$	$7.9 \cdot 10^{-2}$
41	$1.0 \cdot 10^{-1}$	$2.2 \cdot 10^{-2}$	$6.0 \cdot 10^{-3}$	$3.5 \cdot 10^{-1}$	$7.7 \cdot 10^{-2}$	$6.4 \cdot 10^{-2}$
42	$\mathbf{6.2 \cdot 10^{-2}}$	$1.4 \cdot 10^{-2}$	$2.8 \cdot 10^{-3}$	$3.5 \cdot 10^{-1}$	$7.7 \cdot 10^{-2}$	$6.3 \cdot 10^{-2}$
54	$\mathbf{3.1 \cdot 10^{-2}}$	$6.8 \cdot 10^{-3}$	$1.9 \cdot 10^{-3}$	$9.0 \cdot 10^{-2}$	$2.0 \cdot 10^{-2}$	$3.3 \cdot 10^{-2}$
55	$\mathbf{3.1 \cdot 10^{-2}}$	$6.8 \cdot 10^{-3}$	$1.9 \cdot 10^{-3}$	$\mathbf{7.4 \cdot 10^{-2}}$	$1.6 \cdot 10^{-2}$	$3.0 \cdot 10^{-2}$
64	$\mathbf{3.0 \cdot 10^{-2}}$	$6.6 \cdot 10^{-2}$	$1.8 \cdot 10^{-3}$	$\mathbf{4.4 \cdot 10^{-2}}$	$9.6 \cdot 10^{-3}$	$2.8 \cdot 10^{-2}$

Table 6.37: Example V, LSNCG-POD, $X = H$: Absolute & relative errors in the control variable and absolute errors in the state variable for $u_{bc} = \bar{u}^h$ with 'basis B_S' and for $u_{bc} = 0$ with 'basis B_{SA}' for different ℓ.

	$u_{bc} = u^{(3)}$			$u_{bc} = \bar{u}^h$		
ℓ	$e^u_{abs}(\ell)$	$e^u_{rel}(\ell)$	$e^y_{abs}(\ell)$	$e^u_{abs}(\ell)$	$e^u_{rel}(\ell)$	$e^y_{abs}(\ell)$
1	$3.9 \cdot 10^{+0}$	$8.5 \cdot 10^{-1}$	$2.1 \cdot 10^{-1}$	$3.9 \cdot 10^{+0}$	$8.6 \cdot 10^{-1}$	$2.0 \cdot 10^{-1}$
3	$2.1 \cdot 10^{+0}$	$4.5 \cdot 10^{-1}$	$1.3 \cdot 10^{-1}$	$2.2 \cdot 10^{+0}$	$4.7 \cdot 10^{-1}$	$1.2 \cdot 10^{-1}$
5	$8.0 \cdot 10^{-1}$	$1.7 \cdot 10^{-1}$	$4.1 \cdot 10^{-2}$	$3.5 \cdot 10^{-1}$	$7.6 \cdot 10^{-2}$	$1.3 \cdot 10^{-2}$
10	$3.6 \cdot 10^{-1}$	$7.9 \cdot 10^{-2}$	$1.4 \cdot 10^{-2}$	$9.6 \cdot 10^{-2}$	$2.1 \cdot 10^{-2}$	$3.3 \cdot 10^{-3}$
11	$3.1 \cdot 10^{-1}$	$6.7 \cdot 10^{-2}$	$1.2 \cdot 10^{-2}$	$\mathbf{4.5 \cdot 10^{-2}}$	$9.8 \cdot 10^{-3}$	$1.1 \cdot 10^{-3}$
15	$2.0 \cdot 10^{-1}$	$4.4 \cdot 10^{-2}$	$1.4 \cdot 10^{-2}$	$\mathbf{1.3 \cdot 10^{-2}}$	$2.8 \cdot 10^{-3}$	$3.1 \cdot 10^{-4}$
16	$2.0 \cdot 10^{-1}$	$4.4 \cdot 10^{-2}$	$1.4 \cdot 10^{-2}$	$\mathbf{7.0 \cdot 10^{-3}}$	$1.5 \cdot 10^{-3}$	$1.6 \cdot 10^{-4}$
20	$1.3 \cdot 10^{-1}$	$3.0 \cdot 10^{-2}$	$1.0 \cdot 10^{-2}$	$\mathbf{5.5 \cdot 10^{-4}}$	$1.2 \cdot 10^{-4}$	$1.4 \cdot 10^{-5}$
30	$8.2 \cdot 10^{-2}$	$1.8 \cdot 10^{-2}$	$6.0 \cdot 10^{-3}$	$\mathbf{2.2 \cdot 10^{-5}}$	$4.9 \cdot 10^{-6}$	$2.1 \cdot 10^{-6}$
34	$\mathbf{4.3 \cdot 10^{-2}}$	$9.5 \cdot 10^{-3}$	$3.3 \cdot 10^{-3}$	$\mathbf{1.6 \cdot 10^{-5}}$	$3.5 \cdot 10^{-6}$	$1.6 \cdot 10^{-6}$
40	$\mathbf{2.8 \cdot 10^{-2}}$	$6.1 \cdot 10^{-3}$	$2.3 \cdot 10^{-3}$	$\mathbf{1.4 \cdot 10^{-5}}$	$3.1 \cdot 10^{-6}$	$1.5 \cdot 10^{-6}$
64	$\mathbf{3.1 \cdot 10^{-3}}$	$6.8 \cdot 10^{-4}$	$3.6 \cdot 10^{-4}$	$\mathbf{7.1 \cdot 10^{-7}}$	$1.6 \cdot 10^{-7}$	$7.3 \cdot 10^{-8}$

Table 6.38: Example V, LSNCG-POD, $X = H$, 'basis B_{SA}': Absolute & relative errors in the control variable and absolute errors in the state variable for $u_{bc} = u^{(3)}$ of TR-POD run 5.1 and $u_{bc} = \bar{u}^h$ for different ℓ.

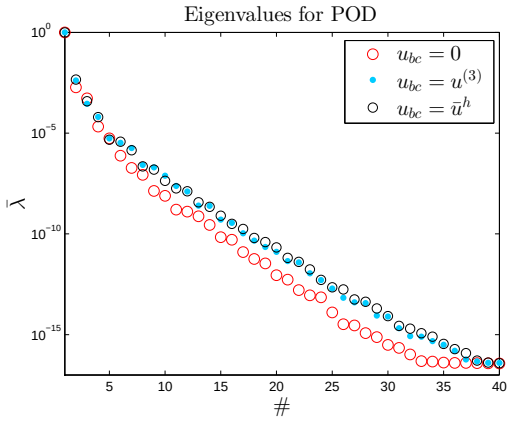


Figure 6.34: Example V: Decay of normalized eigenvalues for the POD basis computed from $u_{bc} = 0$, from $u_{bc} = u^{(3)}$ of TR-POD run 5.1 and from $u_{bc} = \bar{u}^h$ with $X = H$ and 'basis B_{SA}'.

functions computed from $u_{bc} = 0$ in Figure 6.35. Here, the first and the third basis function represent the dynamics of the uncontrolled state whereas the second and the fourth basis function are attributed to the dynamics of the uncontrolled adjoint state. The first four POD basis functions corresponding to the control iterate $u^{(3)}$ are visualized in Figure 6.36. These basis functions are completely different compared to the above ones. But they are very similar to the optimal POD basis functions. These can be seen in Figure 6.37. We conclude that the worse LSNCG-POD results for $u_{bc} = u^{(3)}$ compared to $u_{bc} = \bar{u}^h$, see Table 6.38, can not arise from these little differences in the first four POD basis functions. Notice that the first four POD basis functions are the four most important ones. The larger the number ℓ the smaller is the POD eigenvalue λ_ℓ and the more local is the support of the corresponding POD basis function. We compare the shape of the subsequent POD basis functions. Already in the fifth basis functions we observe larger differences, as it can be seen in Figure 6.38. For $\ell \geq 5$ the POD basis functions remain similar in basic shape but the occuring differences, similar to those in Figure 6.38, are apparently the reason for the worse POD Galerkin approximations when utilizing $u_{bc} = u^{(3)}$. To conclude, little deviations of the control u_{bc} from the optimal FE control $\bar{u}^h$ can deteriorate the POD basis critically for the given example.

We tabulate one LSNCG-POD run. In **LSNCG-POD run 5.1** we utilize the control $u_{bc} = \bar{u}^h$ with 'basis B_{SA}' and $\ell = 11$ POD basis functions. It is summarized in Table 6.39. The reader is asked to compare this table to Table 6.35. Notice that the last step in both the LSNCG-FE run and in LSNCG-POD run 5.1 is very small. The step is performed because it provides a huge last reduction in the gradient norm. The performance of LSNCG-POD run 5.1 is very satisfactory although the differences to the LSNCG-FE run are a bit larger compared to Example III (see Tables 6.12 and 6.17).

Computational efficiency: The computational performance of LSNCG-POD run 5.1 is shown in Table 6.40. The overall CPU time to compute the suboptimal control $\bar{u}^\ell$ is 10.70 seconds. This is about 7.4 times smaller than the CPU time required by the LSNCG-FE optimization. Recall that the achieved speedup factors for the selected LSNCG-POD

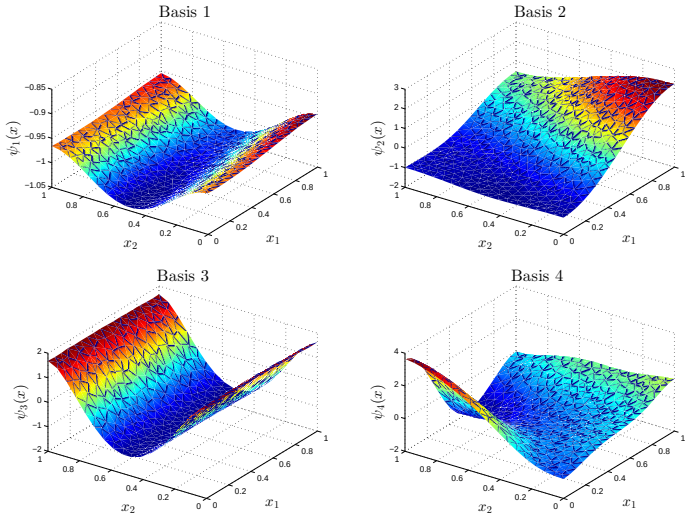


Figure 6.35: Example V: First four POD basis functions computed from $u_{bc} = 0$ with $X = H$ and 'basis B_{SA}'.

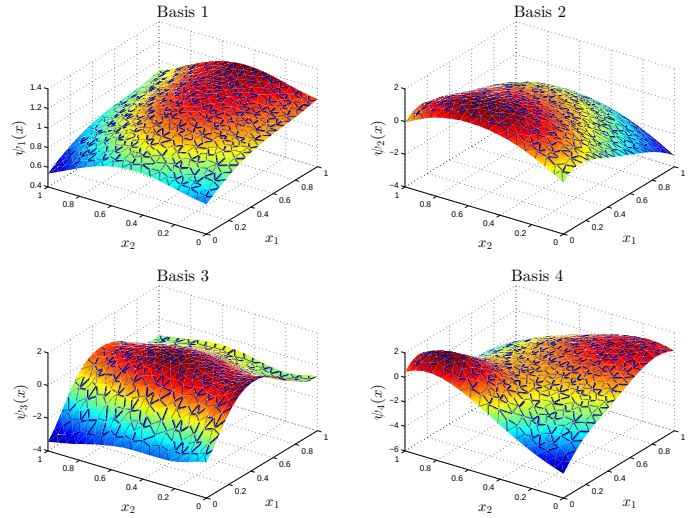


Figure 6.36: Example V: First four POD basis functions computed from $u_{bc} = u^{(3)}$ of TR-POD run 5.1 with $X = H$ and 'basis B_{SA}'.

k	$\hat{J}^\ell(u^{(k)})$	$\|g_\ell^{(k)}\|_{U_h}$	η_k	#CG it	$\|d^{(k)}\|_{U_h}$	t_k
0	1.69643	$1.2 \cdot 10^{+0}$	0.01000	5	2.80	1
1	0.59111	$3.9 \cdot 10^{-1}$	0.00328	5	1.97	1
2	0.29357	$1.1 \cdot 10^{-1}$	0.00094	9	1.60	1
3	0.23647	$2.2 \cdot 10^{-2}$	0.00018	10	0.43	1
4	0.23377	$1.1 \cdot 10^{-3}$	0.00001	14	0.02	1
5	0.23377	$5.4 \cdot 10^{-6}$				

Table 6.39: Example V: Iterations of LSNCG-POD run 5.1.

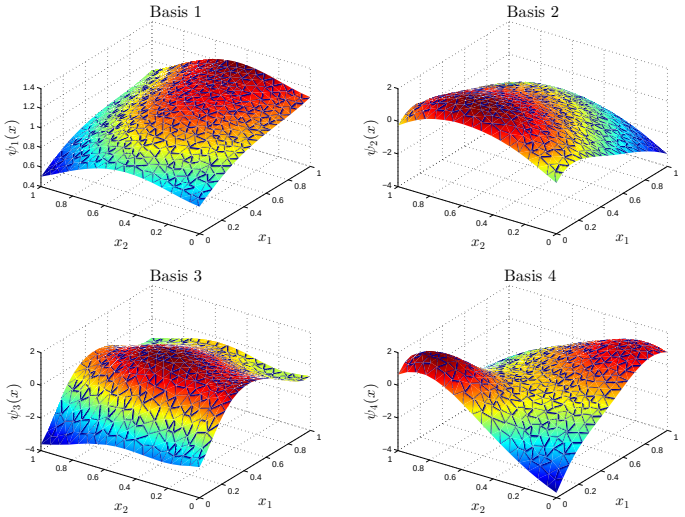


Figure 6.37: Example V: First four POD basis functions computed from $u_{bc} = \bar{u}^h$ with $X = H$ and 'basis B_{SA}'.

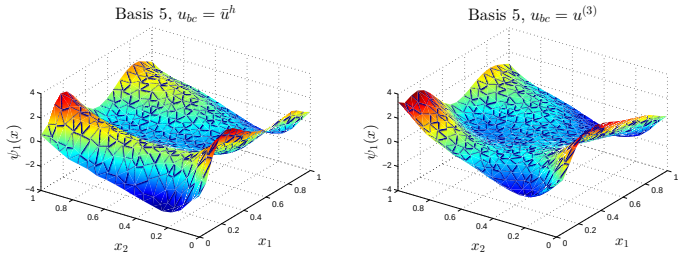


Figure 6.38: Example V: Fifth POD basis function computed from $u_{bc} = \bar{u}^h$ (left) and from $u^{(3)}$ of TR-POD run 5.1 (right) with $X = H$ and 'basis B_{SA}'.

	CPU time	
LSNCG-FE optimization	79.40	
CG algorithm	67.32	(8.67, 8.36, 11.61, 17.43, 21.25)
Snapshot generation	1.94	
- state	1.20	
- adjoint state	0.74	
POD basis ($\ell = 30$)	0.23	
LSNCG-POD optimization	8.53	
CG algorithm	6.34	(0.80, 0.82, 1.28, 1.47, 1.97)

Table 6.40: Example V: CPU times measured in seconds of LSNCG-POD run 5.1 compared to LSNCG-FE optimization (in brackets: CPU time needed by the CG algorithm per iteration).

runs in Example III and IV are of order 4.7 and 5.6. Hence, here we obtain the largest reduction in computational costs. We observe that the speedup factor becomes larger if the FE optimization takes longer. Of course, the setup of the POD basis which includes the snapshot generation takes then, relatively seen, less time.

Step (3):
In step (2) the LSNCG-POD approach requires an utmost good control u_{bc} in order to obtain satisfactory results. The system dynamics caused by the function $N(y) = -0.5y^3$ appear to be more complicated or complex as for the nonlinearity $N(y) = y^3$. This makes it very interesting to investigate the TR-POD approach. For the trust region strategy we define

$$\ell_{max} := 40, \quad \Delta^{(0)} = 3.1, \quad \eta_1 := 0.1,\ \eta_2 := 0.8,$$
$$\gamma_1 := 0.25,\ \gamma_2 := 0.9,\ \gamma_3 := 1.2.$$

TR-FE optimization: Analogous to Example III, a globalization strategy for the FE based Newton-CG method is not needed. Hence, the TR-FE iterates are just those of the LSNCG-FE run. The ratios of actual and predicted reduction are given by $\rho^{(0)}(d_t^{(0)}) = 1.24$, $\rho^{(1)}(d_t^{(1)}) = 1.22$, $\rho^{(2)}(d_t^{(2)}) = 1.13$, $\rho^{(3)}(d_t^{(3)}) = 1.00$ and $\rho^{(4)}(d_t^{(4)}) = 0.39$. The latter value indicates that the corresponding step is only of very small size. The TR-FE optimization takes 103.40 seconds CPU time. Recall that the LSNCG-FE run needs 79.40 seconds. Here, the line search strategy is clearly superior to the trust region globalization in terms of computational effort.

We use $X = H$ and 'basis B_{SA}' for the POD basis computation. The first iterations of **TR-POD run 5.1** are given in Table 6.41. For this run we pursued the strategy of TR-POD run 4.3: Before testing the Carter condition we compute the number $\tilde{\ell}_0$ of POD basis functions as the minimal number which fulfills $1 - \mathcal{E}(\tilde{\ell}_0) < \|\hat{J}'(u^{(k)})\|_{U_h} \cdot 10^{-6}$. Then, the number ℓ_0 is chosen as $\ell_0 := \min\{\max\{\tilde{\ell}_0, \ell_0\}, \ell_{max}\}$. The idea behind is to better take some more POD basis functions than the number determined by the Carter condition in order to obtain an accurate POD Galerkin approximation of the Hessian. As we can see, the values $\hat{J}(u^{(k)})$, $k = 1, 2, 3$, of the reduced cost functional and the corresponding gradient norms are close to those of the LSNCG-FE run (equal to the TR-FE run). The latter ones can be looked up in Table 6.35. Up to the control iterate $u^{(3)}$ the TR-POD strategy works very well. The first POD based inexact Newton steps are computed with only $\ell = 5, 8$ and 9 POD basis functions. Let us add that if the POD basis corresponding to the initial iterate is kept for the second step, we obtain $\hat{J}(u^{(2)}) = 0.31021$ and $\|\hat{J}'(u^{(2)})\|_{U_h} = 1.3 \cdot 10^{-1}$, computed with only $\ell = 6$ POD basis functions. Here, we could omit the update of the POD basis.

Remember that the control iterate $u^{(3)}$ of TR-POD run 5.1 is shown in the right plot in Figure 6.33 and that it is utilized as control u_{bc} for the LSNCG-POD procedure in step (2). But for small POD basis ranks ℓ this control iterate does not yield good POD suboptimal controls. And here we observe that the inexact Newton step computed in iteration $k = 3$ provides only a small reduction in both the reduced cost functional value and the gradient norm, with $\rho^{(3)}(d_t^{(3)}) = 0.26$. The subsequent non-listed iterations of TR-POD run 5.1 can be summarized as follows: In iteration $k = 4$ the POD based inexact Newton step

k	$\hat{J}(u^{(k)})$	$\|\hat{J}'(u^{(k)})\|_{U_h}$	#CG it	$\|d_t^{(k)}\|_{U_h}$	$\Delta^{(k)}$	$\rho^{(k)}(d_t^{(k)})$	ℓ	$\zeta_\ell^{(k)}$
0	1.77870	$1.2 \cdot 10^{+0}$	5	3.08	3.1	1.20	5	0.004
1	0.63676	$4.2 \cdot 10^{-1}$	7	2.40	3.7	1.19	8	0.005
2	0.30921	$1.3 \cdot 10^{-1}$	10	1.80	4.5	1.03	9	0.008
3	0.23755	$2.9 \cdot 10^{-2}$	10	0.56	5.4	0.26	11	0.010
4	0.23640	$1.9 \cdot 10^{-2}$						

Table 6.41: Example V: First iterations of TR-POD run 5.1.

($\ell = 12$) does not yield a successful step, the trust region radius is decreased (even twice) and the Cauchy step must be taken as trial step. In the next iteration a POD based inexact Newton step is computed. But then, the trust region radius is again decreased and a Cauchy step is performed. Including the unsuccessful iterations the TR-POD optimization needs 10 iterations. This takes 39.76 seconds CPU time. At least, this is still about half the CPU time of the LSNCG-FE run. The final number of POD basis functions is $\ell = 13$. If we do not modify the number ℓ_0 in each iteration, i.e. if we let it just to the Carter condition to determine the number of used POD basis functions, the basic observation is the same but it takes even 15 iterations (with final number $\ell = 12$) and already in iteration $k = 3$ the trust region radius is decreased and a Cauchy step carried out.

The problem in TR-POD run 5.1 is the accuracy of the Hessian approximation which is not monitored during optimization. If we increase the number ℓ of POD basis functions in iteration $k = 4$ manually, first for $\ell = 33$ the resulting POD based inexact Newton step is successfully taken as trial step. In this case the POD basis is computed from the current iterate $u^{(4)}$ with $\varepsilon^u_{abs}(u^{(4)}) = 3.0 \cdot 10^{-1}$. But obviously, the thereby obtained POD Galerkin approximation of the Hessian is quite poor. If we keep the POD basis from $u^{(3)}$, $\ell = 36$ POD basis functions are needed to satisfy the Carter condition in iteration $k = 4$. The resulting inexact Newton step provides a successful iteration. Notice that $\|u^{(3)} - u^{(4)}\|_{U_h} = \|d_t^{(3)}\|_{U_h}$ is only of order 0.56.

The above observations constitute the worst-case scenario. The approximation quality of the model Hessian, which is not controlled within the presented TR-POD strategy, becomes critical when moving towards the optimal control. As a consequence, our TR-POD algorithm is sometimes forced to perform POD based Cauchy steps instead of the desired POD based inexact Newton steps. Nevertheless, since convergence is guaranteed, the optimal control is found and the POD basis is altered in direction of the optimal basis during optimization. But once we are quite close to the optimal control, say $\varepsilon^u_{abs}(u^{(k)}) < 1$, the use of a POD basis computed from the current iterate does not always yield a satisfactory POD Galerkin approximation of the Hessian. The existing differences of the respective POD basis functions from the optimal ones (see Figure 6.38) sometimes result in the need of a huge POD basis rank in order to obtain a sufficient Hessian accuracy for the use of inexact Newton steps. To sum up, the TR-POD procedure works very well in the sense of performing inexact Newton steps until the iterates enter a certain neighbourhood of the optimal control. But if we want to terminate the optimization with this POD based second-order Newton-CG strategy, the Hessian approximation has

to be monitored. Even though, this might sometimes require a huge number of POD basis functions. If the POD basis rank is kept low, Cauchy steps are now and then performed instead of inexact Newton steps. To conclude, in the absolute worst case we can imagine, our TR-POD procedure might result in a POD based first-order gradient method, if the Hessian accuracy is not considered.

7 Conclusion

In a basic step the considered boundary optimal control problem is formulated as a reduced problem by including the semilinear heat equation in the formulation of the cost functional. The computation of first- and second-order derivatives is presented in detail. These enable to solve the nonlinear reduced problem by a globalized inexact Newton method. The inexact Newton steps are computed iteratively with a CG algorithm.

In a first approach, an Armijo backtracking strategy is chosen for globalization of the Newton-CG procedure. For discretizing the spatial variable a classical FE Galerkin technique is applied. The presented strategy works extremely efficiently for solving the optimal control problem in terms of required iterations. Likewise, the CG iterations converge very rapidly. But within our algorithm several PDEs have to be solved repeatedly. In discrete form this requires the numerous solution of large-dimensional (originating from the FE subspace) nonlinear and linear systems of equations.

Hence, we apply a reduced-order approach based on POD to reduce the computational complexitiy. In our numerical experiments we first discuss the application of a POD Galerkin method for the solution of a semilinear heat equation. Since the resulting semi-discrete scheme is nonlinear we investigate the use of two different versions of empirical interpolation. This leads to a ROM which is completely independent of the underlying FE dimension. Our results demonstrate efficiency and accuracy of the derived ROM. But a noteworthy speedup in computational time by adding empirical interpolation to the POD approach is only achieved if the underlying FE dimension is 'sufficiently large'.

The main part of this work deals with the application of POD for the numerical solution of the optimal control problem. The above described optimization procedure is applied by using a POD Galerkin method. Here, a control u_{bc} to set up the POD basis is chosen at the beginning and the ROMs are fixed during optimization. The same POD basis is used for the involved PDEs. It is crucial to add snapshots from the adjoint state to the state snapshot ensemble in order to obtain satisfactory results. If the POD basis rank is kept low, the POD based (globalized) Newton-CG strategy is highly efficient for the solution of the given optimal control problem. But if the system dynamics resulting from the control u_{bc} differ from those associated with the optimal FE control, few POD basis functions do not suffice to obtain good POD suboptimal controls. The accuracy of the ROMs depends essentially on the choice of the control u_{bc}. There is no guarantee that the reduced-order globalized Newton-CG strategy converges to a stationary point of the reduced cost functional.

To overcome this problem we embed the reduced-order Newton-CG procedure in a trust region framework. We improve the POD basis successively in the course of optimization by utilizing the updated control values. Our TR-POD algorithm belongs to the class of trust region methods that use a quadratic model function with inexact gradient information. Both the gradient and the Hessian of the model function are computed from the respective POD Galerkin approximations of the reduced cost functional. Whenever the POD based inexact Newton step does not exist or whenever it does not satisfy the fraction of Cauchy decrease condition, the POD based Cauchy step is taken as trial step. To guarantee the

named convergence the number of POD basis functions is controlled such that gradient accuracy in the sense of Carter is satisfied.

Our numerical results show a rapid adaptation of the POD basis in direction of the optimal basis. In two presented examples for the nonlinearity $N(y) = y^3$ we observe the following: The first TR-POD step, which is a POD based inexact Newton step or simply a POD based Cauchy step, leads to a POD basis which is as close to the optimal basis such that the POD optimization process can be terminated in a satisfactory manner by keeping the corresponding ROMs. The presented TR-POD algorithm works well and our results motivate the investigation of error estimates, amongst others to replace the Carter condition, for improving efficiency in terms of computational effort.

An open question is the handling of the Hessian accuracy, which is not monitored within the presented TR-POD strategy. Our numerical examples reveal this weakness because the number ℓ of POD basis functions determined by the Carter condition does not always yield a sufficient Hessian accuracy in order to obtain a satisfactory POD based inexact Newton step. In this case the algorithm might be forced to perform a POD based Cauchy step. But in doing this, convergence is guaranteed and the POD basis is altered in direction of the optimal basis.

8 Deutsche Zusammenfassung

Gegenstand der vorliegenden Diplomarbeit ist die numerische Lösung eines Optimalsteuerungsproblems mit semilinearer Wärmeleitungsgleichung. Die Steuerung wirkt als Wärmequelle in einem vorgegebenen Zeitintervall $[0, T]$, $T > 0$, auf den Rand des Ortsgebietes und ist punktweise beschränkt. Zu Beginn ist eine gewisse Temperaturverteilung im Gebiet gegeben. Es wird eine optimale Steuerung gesucht, sodass die daraus resultierende Temperaturverteilung zur Endzeit eine gewünschte Temperaturverteilung bestmöglich approximiert. Gleichzeitig sollen die Kosten für die Steuerung minimal gehalten werden. Dies wird durch ein zu minimierendes quadratisches Zielfunktional beschrieben, welches von Zustand und Steuerung abhängt.

Zu jeder Steuerung gehört genau ein Zustand. Dieser ist gegeben als eindeutige Lösung der semilinearen Wärmeleitungsgleichung. Deshalb lässt sich die Zustandsvariable als Funktion der Steuervariablen ausdrücken. Auf diese Weise können wir den Zustand im Zielfunktional eliminieren und erhalten ein reduziertes Zielfunktional, welches nur noch von der Steuerung abhängt. Dies ermöglicht uns, das ursprüngliche Problem als ein auf die Steuerung reduziertes nichtlineares Optimierungsproblem zu schreiben.

Das reduzierte Problem wird mit einem inexakten Newton-Verfahren gelöst. Dieses ist bekannt für seine schnelle lokale Konvergenz. Die Newton-Schritte werden hierbei mit der Methode der konjugierten Gradienten berechnet. Dabei muss die Hesse-Matrix nicht explizit bekannt sein. Es genügt eine Bildungsvorschrift für ihre Anwendung auf Steuerfunktionen. Diese wird, ebenso wie der Gradient des reduzierten Zielfunktionals, detailliert hergeleitet.

In einem ersten Schritt nutzen wir eine Armijo-Schrittweitenstrategie zur Globalisierung des verwendeten inexakten Newton-Verfahrens. Die räumliche Diskretisierung wird dann zunächst mit einer klassischen Finite-Elemente-Galerkin-Methode realisiert. Der so erhaltene Algorithmus ist äußerst effizient was die Anzahl der benötigen inexakten Newton-Schritte und die Anzahl der CG-Iterationen betrifft. Allerdings müssen im Laufe der Optimierung mehrere partielle Differentialgleichungen wiederholt gelöst werden. In diskreter Form verlangt dies das zahlreiche Lösen groß-dimensionaler linearer und nichtlinearer Gleichungssysteme.

Um den daraus resultierenden numerischen Aufwand zu verringern, wenden wir uns einem Verfahren der Modellreduktion zu, und zwar der Proper Orthogonal Decomposition (POD) Methode. Der beschriebene Algorithmus wird dann unter Verwendung einer POD-Galerkin-Methode angewandt. Es muss anfangs eine Steuerung zur POD-Basisberechnung gewählt werden. Die Modelle reduzierter Ordnung werden dann mittels der berechneten POD-Basis aufgestellt und bleiben während des Optimierungsprozesses unverändert. Ist die Anzahl der verwendeten POD-Basisfunktionen klein, so kann die Rechenzeit mit diesem Ansatz im Vergleich zur Finite-Elemente-Methode deutlich verkürzt werden. Allerdings hängt die Qualität der Modelle reduzierter Ordnung von der zugrunde liegenden POD-Basis ab. Bei „schlechter" Wahl der zur POD-Basisberechnung genutzten Steuerung reichen wenige POD-Basisfunktionen nicht aus um gute suboptimale, mittels POD berechnete, Steuerungen zu erhalten. Bei willkürlicher Wahl der zur

POD-Basisberechnung genutzten Steuerung ist nicht garantiert, dass das auf den POD-Modellen basierende globalisierte inexakte Newton-Verfahren zu einem stationären Punkt des reduzierten Zielfunktionals konvergiert.

Um diese Problematik zu überwinden, verfolgen wir einen weiteren Ansatz. In diesem wird die POD-Basis im Laufe der Optimierung angepasst und dadurch verbessert. Dies realisieren wir, indem wir das POD basierte inexakte Newton-Verfahren in eine Trust-Region-Strategie einbetten. Damit ist die genannte Konvergenz gesichert. Der Fokus dieser Arbeit liegt darauf, den so erhaltenen Algorithmus auf Funktionalität und Effizienz bezüglich der Anpassung der POD-Basis zu testen. Seine zeitliche Effizienz ist zunächst nebensächlich. Unsere numerischen Ergebnisse zeigen, dass sich die POD-Basis im Laufe des Optimierungsprozesses schnell und signifikant der „optimalen" Basis annähert.

Bibliography

[1] R.A. Adams and J.F. Fournier. *Sobolev Spaces.* Pure Appl. Math. 65, Academic Press, New York, 1975.

[2] E. Arian, M. Fahl and E.W. Sachs. *Trust-region proper orthogonal decomposition for flow control.* Technical Report 2000-25, ICASE, 2000.

[3] M. Barrault, Y. Maday, N.C. Nguyen and A.T. Patera. *An 'empirical interpolation' method: Application to efficient reduced-basis discretization of partial differential equations.* C. R. Acad. Sci. Paris, Ser. I, 339:667–672, 2004.

[4] F. Bengzon and M.G. Larson. *The Finite Element Method: Theory, Implementation, and Applications.* Springer-Verlag, Berlin Heidelberg, 2013.

[5] D. Braess. *Finite Elemente. Theorie, schnelle Löser und Anwendungen in der Elastizitätstheorie.* 5. ed., Springer-Verlag, Berlin, , 2013.

[6] R.G. Carter. *On the global convergence of trust region algorithms using inexact gradient information.* SIAM J. on Numerical Analysis 28(1):251–265, 1991.

[7] S. Chaturantabut and D.C. Sorensen. *Nonlinear model reduction via discrete empirical interpolation.* SIAM J. Sci. Comput., 32:2737–2764, 2010.

[8] L.C. Evans. *Partial Differential Equations.* 2nd ed., Math. Society, Providence, Rhode Island, 2010.

[9] A.R. Conn, N.I.M. Gould and P.L. Toint. *Trust-Region Methods.* SIAM, Philadelphia, 2000.

[10] W. Dahmen and A. Reusken. *Numerik für Ingenieure und Naturwissenschaftler.* 2nd ed., Springer-Verlag, Berlin, 2008.

[11] R. Dautray and J.-L. Lions. *Mathematical Analysis and Numerical Methods for Science and Technology. Volume 5: Evolution Problems I.* Springer-Verlag, Berlin, 2000.

[12] G. Dziuk. *Theorie und Numerik partieller Differentialgleichungen.* Walter de Gruyter, Berlin, 2010.

[13] M. Fahl. *Trust-Region Methods for Flow Control Based on Reduced Order Modelling.* PhD thesis, University of Trier, 2000.

[14] C. Geiger and C. Kanzow. *Numerische Verfahren zur Lösung unrestringierter Optimierungsaufgaben.* Springer-Verlag, Berlin, 1999.

[15] M.A. Grepl, Y. Maday, N.C. Nguyen and A.T. Patera. *Efficient reduced-basis treatment of nonaffine and nonlinear partial differential equations.* ESAIM:Math. Model. Numer. Anal., 41:575-605, 2007.

[16] E. Grimm. *Optimality system POD and a-posteriori error analysis for linear-quadratic optimal control problems.* Master's thesis, University of Konstanz, 2013.

[17] M. Gubisch and S. Volkwein. *Proper orthogonal decomposition for linear-quadratic optimal control.* Submitted, 2013. http://nbn-resolving.de/urn:nbn:de:bsz:352-250378.

[18] B. Haasdonk and M. Ohlberger. *Reduced basis method for finite volume approximations of parametrized evolution equations.* M2AN Math. Model. Numer. Anal., 42(2):277-302, 2008.

[19] M. Heikenschloss, M. Ulbrich and S. Ulbrich. *Global Convergence of Trust-region Interior-point Algorithms for Infinite-dimensional Nonconvex Minimization Subject to Pointwise Bounds.* SIAM J. Control Optim., 37(3):731–764, 1999.

[20] M. Hinze and K. Kunisch. *Second order methods for optimal control of time-dependent fluid flow.* SIAM J. Control Optim. 40:925-946, 2001.

[21] M. Hinze, R. Pinnau, M. Ulbrich and S. Ulbrich. *Optimization with PDE Constraints.* Springer-Verlag, Berlin, 2009.

[22] M. Hinze and S. Volkwein. *Proper orthogonal decomposition surrogate models for nonlinear dynamical systems: error estimates and suboptimal control.* In Reduction of Large-Scale Systems, P. Benner, V. Mehrmann and D.C. Sorensen (eds.), Lecture Notes in Computational Science and Engineering, 45:261-306, 2005.

[23] P. Holmes, J.L. Lumley, G. Berkooz and C.W. Rowley *Turbulence, Coherent Structures, Dynamical Systems and Symmetry.* 2nd ed., Cambridge Monogr. Mech., Cambridge University Press, New York, 2012.

[24] E. Kammann, F. Tröltzsch and S. Volkwein. *A-posteriori error estimation for semi-linear parabolic optimal control problems with application to model reduction by POD.* ESAIM. Mathematical Modelling and Numerical Analysis, 47(2):555–581, 2013.

[25] C.T. Kelley. *Iterative Methods for linear and nonlinear equations.* Frontiers in Applied Mathematics, SIAM, Philadelphia, PA, 1995.

[26] C.T. Kelley. *Iterative Methods for Optimization.* Frontiers in Applied Mathematics, SIAM, Philadelphia, PA, 1999.

[27] P. Knabner and L. Angermann. *Numerik partieller Differentialgleichungen. Eine anwendungsorientierte Einführung.* Springer-Verlag, Berlin Heidelberg, 2000.

[28] N. Köckler and H.R. Schwarz. *Numerische Mathematik.* 7th ed., Vieweg+Teubner, Wiesbaden, 2009.

[29] K. Kunisch and S. Volkwein. *Galerkin proper orthogonal decomposition methods for parabolic problems.* Num. Math., 90:117-148, 2001.

[30] K. Kunisch and S. Volkwein. *Galerkin proper orthogonal decomposition for a general equation in fluid dynamics.* SIAM J. Numer. Anal., 40:492-515, 2002.

[31] K. Kunisch and S. Volkwein. *Proper orthogonal decomposition for optimality systems.* ESAIM: Mathematical Modelling and Numerical Analysis, 42:1-23, 2008.

[32] S. Kurcyusz and J. Zowe. *Regularity and stability for the mathematical programming problem in Banach spaces.* Appl. Math. Optimization, 5:49–62, 1979.

[33] O. Lass. *Efficient POD reduced-order modeling for parametrized nonlinear PDE systems.* Konstanzer Schriften in Mathematik, 130, 2012. `http://nbn-resolving.de/urn:nbn:de:bsz:352-208711`

[34] O. Lass and S. Volkwein. *POD Galerkin Schemes for nonlinear elliptic-parabolic systems.* SIAM J. ScI. Comput. 35:A1271–A1298, 2013.

[35] J.J. Moré. *Recent Developments in Algorithms and Software for Trust Region Methods.* In Mathematical Programming - The State of the Art, A. Bachem, M. Grötschel and B. Korte (eds.), Springer-Verlag, Berlin, 258-287, 1983.

[36] I. Neitzel and B. Vexler. *A priori error estimates for space-time finite element discretization of semilinear parabolic optimal control problems.* Num. Math. 120(2):345-386, 2012.

[37] P. Quittner and P. Souplet. *Superlinear Parabolic Problems: Blow-up, Global Existence and Steady States.* Birkhäuser-Velag, Basel, 2007.

[38] A. Patera and G. Rozza. *Reduced Basis Approximation and A-Posteriori Error Estimation for Parametrized Partial Differential Equations.* MIT Pappalardo Graduate Monographs in Mechanical Engineering, 2006.

[39] C. Pozrikidis. *Introduction to Finite and Spectral Element Methods Using MATLAB.* 2nd ed. Taylor & Francis/CRC, Boca Raton, 2014.

[40] J. Schropp. *Numerik Partieller Differentialgleichungen II.* Lecture notes, University of Konstanz, Summer Term 2012. http://www.math.uni-konstanz.de/schropp/numerik4/skript.pdf.

[41] M. Schu. *Adaptive Trust-Region POD Methods and Their Applications in Finance.* Ph.D thesis, University of Trier, 2012.

[42] A. Studinger. *Numerical Analysis of POD A-Posteriori Error Estimates for Linear-Quadratic Optimal Control Problems.* Diploma thesis, University of Konstanz, 2011.

[43] A. Studinger and S. Volkwein. *Numerical analysis of POD a-posteriori error estimation for optimal control.* In Control and Optimization with PDE Constraints, K. Kunisch, K. Bredies, C. Clason and G. von Winckel (eds.), International Series of Numerical Mathematics, 164, 2013.

[44] P. Toint. *Global convergence of a class of trust region methods for nonconvex minimization in Hilbert spaces.* IMA Journal of Numerical Analysis, 8(2): 231–252, 1988.

[45] F. Tröltzsch. *Optimal Control of Partial Differential Equations. Theory, Methods and Applications.* American Mathematical Society, Providence, 112, 2010.

[46] F. Tröltzsch and S. Volkwein. *POD a-posteriori error estimates for linear-quadratic optimal control problems.* Computational Optimization and Applications, 44:83-115, 2009.

[47] M. Ulbrich and S. Ulbrich. *Nichtlineare Optimierung.* Springer-Verlag, Berlin, 2012.

[48] S. Volkwein. *Proper Orthogonal Decomposition: Theory and Reduced-Order Modelling.* Lecture Notes, University of Konstanz, Summer Term 2012. http://www.math.uni-konstanz.de/numerik/personen/volkwein/teaching/POD-Book.pdf

[49] J. Nocedal and S.J. Wright. *Numerical Optimization.* 2nd ed., Springer, New York, 2006.

[50] E. Zeidler. *Nonlinear Functional Analysis and its Applications II/B: Nonlinear Monotone Operators.* Springer, New York, 1990.

[51] K. Zhou, J. Doyle and K. Glover. *Robust and Optimal Control.* Prentice Hall, Upper Saddle River, NJ, 1996.

Acknowledgments

MANY THANKS TO ...

... my supervisor, Prof. Dr. Stefan Volkwein, for motivating lectures that aroused my interest in the topics Optimization and POD and for giving me the opportunity to write this thesis. It certainly brought me forward in my personal developement. I am grateful for constructive feedback and informative answers to my questions. Besides, many thanks for the invitations to very interesting talks.

... Prof. Dr. Ekkehard Sachs for accepting to be my second reviewer and his interest in my work.

... Stefan Trenz for interesting discussions and helpful suggestions. I enjoyed to get an insight in distributed nonlinear optimal control problems and some corresponding numerical observations.

... Carmen-Constanze Gräßle for the exchange we had during our diploma theses. In particular, I am thankful for useful comparisons of my numerical results to those of (POD based) inexact SQP methods.

... Carmen-Constanze Gräßle, Laura Lippmann and Christiane Mezger for carefully proofreading my thesis. The comments and suggestions I received were very valuable to improve my work.

... my parents Karola and Walter, my brother Tobias and my sisters Stefanie, Marina and Lorena. Just a huge thanks!