Adaptive Lossy Trajectory Compression for Optimal Control of Parabolic PDEs

Dissertation

**zur Erlangung des Grades eines
Doktors der Naturwissenschaften**

**am Fachbereich Mathematik und Informatik
der Freien Universität Berlin**

vorgelegt von

Sebastian Götschel

Berlin, Mai 2014

Erstgutachter:
Prof. Dr. Dr. h.c. Peter Deuflhard
Fachbereich Mathematik und Informatik
Freie Universität Berlin
Arnimallee 2-6
14195 Berlin

Zweitgutachter:
Prof. Dr. Matthias Heinkenschloss
Department of Computational and Applied Mathematics
Rice University
6100 Main St. – MS 134
Houston, TX 77005-1892
USA

Datum der Disputation: 20. Januar 2015

Contents

Notation

General Function Spaces and Norms

$C^k(\Omega)$	space of k-times continuously differentiable functions on $\Omega \subset \mathbb{R}^d$
$L^p(\Omega)$	Lebesgue space
$W^{k,p}(\Omega)$	Sobolev space
$H^k(\Omega)$	$W^{k,2}(\Omega)$
$L^p(a,b;V)$	linear space of vector-valued functions $y : [a,b] \to V$ with $\int_a^b \|y(t)\|_V^p \ dt < \infty$
$C^k(a,b;V)$	space of k-times continuously differentiable functions $y : [a,b] \to V$
$\|\cdot\|_V$, $(\cdot,\cdot)_V$	norm and scalar product on V
$V^\star$, $\langle\cdot,\cdot\rangle_{V^\star,V}$	dual space of V and dual pairing

Optimal Control Problems

Y	space of state variables y
U	space of control variables u
Z	space of adjoint variables λ
$J(y,u), j(u)$	objective functional and reduced objective functional
$c(y,u) = 0$	state equations

Lossy Compression

$Q_\delta(y), Q_\delta^\dagger(i)$	quantization of y and de-quantization of i with quantization tolerance δ
$\hat{\cdot}$	de-compressed quantities, e.g. $\hat{y} = Q_\delta^\dagger(Q_\delta(y))$
$\varepsilon_\cdot$	quantization error, e.g. $\varepsilon_y = \hat{y} - y$
$\tilde{\cdot}$	inexact quantity due to compression of an input
$e_\cdot$	error due to inexact input, e.g. $\tilde{\lambda} = \lambda + e_\lambda$

1. Introduction

Optimal control problems governed by nonlinear, time-dependent PDEs on three-dimensional spatial domains are an important tool in many fields, ranging from engineering applications to medicine. They occur, for example, in non-destructive testing of materials by active thermography, where the aim is to reconstruct the geometry of the inaccessible rear side from temperature measurements on the front surface. This can be achieved by minimizing the difference between computed and measured temperatures, subject to the heat equation. For medical applications, we exemplarily mention the problem of cardiac defibrillation. Cardiac arrhythmia, like ventricular fibrillation, are treated by applying an electrical shock. There, one optimization goal is the design of shocks with amplitudes as small as possible, which still are sufficient to extinguish fibrillation.

In typical applications, the control is not acting in the whole domain (*distributed control*), but only on the boundary of the domain (thermography), or is spatially localized on a part of the domain and only varying in time (defibrillation). For the solution of such optimization problems, methods working on the reduced objective functional are often employed to avoid a full spatio-temporal discretization of the problem. There, the *state variable* (in the mentioned examples temperature or transmembrane voltage, respectively) is interpreted as a function of the control, which can be computed for any given admissible control by some method (*black-box approach*). This leads to an unconstrained optimization problem, which can be solved by standard gradient-based or higher-order methods. The evaluation of the reduced gradient requires one solve of the state equation forward in time, and one backward solve of the adjoint equation. The state enters into the adjoint equation, requiring the storage of a full 4D data set. If Newton-CG methods are used, two additional trajectories have to be stored.

To get numerical results that are accurate enough, in many cases very fine discretizations in time and space are necessary, leading to a significant amount of data to be stored and transmitted to mass storage. Here, not only the mere storage size is important, with the ever-growing speed of CPUs, storage access time is more and more becoming a bottleneck for large-scale simulation and optimization. To be able to tackle real-world applications, compression methods are required in order to reduce the amount of data.

Two types of methods have been proposed to compress such trajectories: *lossless* and *lossy* compression algorithms. *Checkpointing* is a lossless compression method, originally developed for computation of gradients via the reverse mode of automatic differentiation by Volin and Ostrovskii [135], and Griewank [45]. The state is stored only at selected timesteps, from where on parts of the trajectory are re-computed as needed. This incurs a significant computational overhead, even for optimal placement of checkpoints, cf. Griewank and Walther [46]. Application of checkpointing techniques to solve parabolic optimal control problems can be found, e.g., in the works of Hinze and Sternberg [56, 117], and Becker, Meidner and Vexler [5].

Lossy methods typically consist of three ingredients. Depending on the actual application, a *change of basis* is used to de-correlate the data; in our setting, this often is performed using predictors based on previously encoded data, see, e.g., Lindstrom and Isenburg [82]. Also, wavelet transforms are frequently used in this step, e.g. by Lounsbery, DeRose and Warren [84] for geometry compression, or recently by Rettenmeier [101] for computational fluid dynamics simulations. The accuracy of the data is reduced by *quantization*, before finally the resulting values are *entropy coded*, assigning fewer bits to more frequently occurring coefficients. Theory of quantization has a long history, early works by Bennet [7] and Shannon [112] date back to 1948; a detailed overview was published by Gray and Neuhoff [42]. For entropy coding, arithmetic encoders, introduced by Rissanen and Langdon [103] in the late 1970s, and variants thereof, are widely used.

Checkpointing is tailored to adjoint gradient computation, but computationally expensive. In contrast to this, lossy compression methods have not been designed for adjoint gradient computation, and an analysis of the influence of the quantization error on the gradient, and thus the optimization progress, is not available. Furthermore, these methods, to a large extent, rely on structured grids, or do not exploit the hierarchical structure of (adaptively) refined finite element meshes.

In this thesis, we develop and analyze lossy compression techniques tailored to PDE-constrained optimization. Besides a small computational overhead, we aim at methods that can be easily implemented and included in existing finite element software. A special focus is on the adaptive control of the quantization error through the course of optimization.

Outline. In Chapter 2, we specify the problem setting and summarize important results for the theory of PDE-constrained optimization. We give first and second order optimality conditions. Focusing on methods working on the reduced objective functional, we derive representations for the reduced gradient and reduced Hessian.

The chapter closes with a discussion of the discretization in space and time of the arising parabolic PDEs.

Compression of scientific data—typically double precision floating point values—is a wide area of research. Chapter 3 gives an overview over existing techniques and algorithms. We discuss approaches for general floating point compression as well as methods specialized for optimal control.

In Chapter 4, we develop and analyze a computationally inexpensive lossy compression method, adapted to the specific needs of PDE-constrained optimization. We provide algorithms easily usable on unstructured, adaptively refined grids in two and three space dimensions, and derive a-priori estimates for the achievable compression factors.

Due to the inexact reconstruction, and thus inexact data for the adjoint equation, the error induced in the reduced gradient, and reduced Hessian, has to be controlled, to not impede convergence of the optimization. In Chapter 5, we analyze accuracy requirements of different optimization methods, and derive (computable) error estimates for the influence of lossy trajectory storage. These tools are used to adaptively control the accuracy of the compressed data.

We present a variety of numerical results in Chapter 6. Examples range from a simple boundary control problem for the linear heat equation to the optimal control problem of cardiac defibrillation, where the dynamics are described by a reaction-diffusion system.

In the previous chapters, the quantization error is controlled pointwise. In Chapter 7, we substitute the hierarchical basis transform of the basic algorithm by a wavelet-based transform and suitable quantization of the wavelet coefficients. This allows to control the quantization error in norms other than L^∞. We discuss the construction of finite element wavelets, and give numerical results.

2. Optimal Control of Parabolic PDEs

Both application examples, thermography and cardiac defibrillation, lead to optimization problems governed by parabolic PDEs. In this chapter we fix the problem setting and briefly summarize the theory for optimal control problems. In Section 2.2 we state necessary and sufficient optimality conditions, and introduce adjoint gradient computation. Finally, in Section 2.3 the discretization of the parabolic PDEs is discussed. There, the multilevel nature of the spatial discretization is especially important, as this will be the foundation for the compression method in Chapter 4.

2.1. Problem Setting

We consider the abstract optimal control problem

$$\min_{y \in Y, u \in U} J(y,u) \text{ subject to } c(y,u) = 0, \tag{2.1}$$

with $c : Y \times U \to Z^\star$ a semi-linear parabolic PDE on Banach spaces Y, Z and Hilbert space U. More precisely, we deal with semi-linear systems of m reaction-diffusion equations

$$\begin{aligned} \frac{\partial y}{\partial t} - D\nabla \cdot (\sigma \nabla y) &= f(y) + g_\Omega(u) && \text{in } \Omega \times (0,T) \\ B\partial_\nu y + Cy &= g_\Gamma(u) && \text{on } \partial\Omega \times (0,T) \\ y(\cdot,0) &= y_0 && \text{in } \Omega \end{aligned} \tag{RDS}$$

with $y : \Omega \times (0,T) \to \mathbb{R}^m$, and $D \in \mathbb{R}^{m \times m}$ a diagonal matrix with at least one non-zero element. $\nu = \nu(x)$ denotes the outward unit normal at some $x \in \partial\Omega$ in the boundary condition. Often the distributed control is only supported on parts of the space-time cylinder. A typical example is a time-dependent control which is spatially constant on a control domain $\Omega_c \subset \Omega$, $g_\Omega(u) = \chi_{\Omega_c}(x)u(t)$.

We assume that the spatial domain $\Omega \subset \mathbb{R}^d$ has at least a Lipschitz-continuous boundary Γ; for application of comparison theorems (Appendix A), higher regularity is required (C^2, or at least satisfying the interior sphere property, see, e.g., [34, 10]). Further, we assume that the PDEs possess an at least locally unique solution $y(u)$

for every control $u \in U$. For error estimation in Chapter 5 we require the functions g_Ω, g_Γ to be monotonously increasing in the control u.

Throughout this thesis, we assume that $J : Y \times U \to \mathbb{R}$ is given by

$$J(y,u) = J_1(y) + J_2(u). \tag{2.2}$$

Further, we assume that J and c are sufficiently smooth. Partial derivatives, e.g. of the operator c, are denoted by

$$c_y(y,u) : Y \to Z^\star, \quad c_{yy}(y,u) : Y \times Y \to Z^\star, \quad c_u(y,u) : U \to Z^\star, \text{ etc.},$$

with the corresponding adjoints

$$c_y(y,u)^\star : Z \to Y^\star, \quad c_{yy}(y,u)^\star : Z \to Y^\star \times Y^\star, \quad c_u(y,u)^\star : Z \to U^\star, \text{ etc.}$$

For a classical solution y of the parabolic equation (RDS) we would need at least $Y = C^{2,1}(\Omega \times (0,T))$, i.e. existence and continuity of all appearing derivatives. For optimal control problems, where typically the control belongs to some L^2-space, as well as for the numerical treatment, this is too restrictive. Instead we will use the weak formulation. Let given Hilbert spaces V, H form a Gelfand triple

$$V \hookrightarrow H \hookrightarrow V^\star$$

where the embeddings are continuous and dense. Further define

$$W(0,T) = \{v \in L^2(0,T;V) | v_t \in L^2(0,T;V^\star)\},$$

where $L^2(0,T;V)$ denotes the space of Bochner integrable mappings $f : (0,T) \to V$ and y_t the distributional derivative (see, e.g., [2, 144]). With a mapping $a : V \times V \to \mathbb{R}$, linear in the second argument, which incorporates the spatial differential operator and boundary conditions, we arrive at the weak formulation

$$\begin{aligned} \langle y_t(t), v(t)\rangle_{V^\star,V} + a(y(t),v(t)) - \langle f(y(t)), v(t)\rangle_{V^\star,V} \\ + (y(0) - y_0, v(0))_H = 0 \quad \forall v \in W(0,T). \end{aligned} \tag{2.3}$$

We refer to the literature, e.g. [127, 144], for a thorough discussion, and give two examples for such optimal control problems to conclude this section.

Example 2.1.1. As a simple example we consider boundary control of the linear heat equation. Let $U = L^2((0,T) \times \partial\Omega), V = H^1(\Omega)$,

$$Y = \{y \in L^2(0,T;V) \mid y_t \in L^2(0,T;V^\star)\}$$

and $Z^\star = L^2(0,T;V^\star) \times L^2(\Omega)$. The weak formulation

$$\int_0^T (y_t, \varphi)_{V^\star,V}\ dt + \int_0^T \int_\Omega \nabla y \cdot \nabla \varphi - f\varphi\ dx\ dt + \int_0^T \int_\Gamma (y-u)\varphi\ ds\ dt + \int_\Omega (y(0) - y_0)\varphi(0)\ dx = 0 \qquad \forall \varphi \in Y.$$

defines the bounded linear operator $c : Y \times U \to Z^\star$. This weak formulation of the PDEs corresponds to the initial-boundary value problem

$$\begin{aligned} y_t - \Delta y &= f && \text{in } \Omega \times (0,T) \\ \partial_\nu y + y &= u && \text{on } \Gamma \times (0,T) \\ y(\cdot,0) &= y_0 && \text{in } \Omega. \end{aligned}$$

As an exemplary objective functional we choose

$$J(y,u) = \frac{1}{2}\int_0^T \int_\Omega (y - y_d)^2\ dx\ dt + \frac{\alpha}{2}\int_0^T \int_{\partial\Omega} u^2\ ds\ dt,$$

i.e. the goal is to minimize the deviation from some desired state y_d, with additional penalization of the control effort.

Example 2.1.2. Optimal control of cardiac defibrillation. A simplified approximation for the electrical activity of the heart muscle is given by the monodomain model, consisting of a reaction-diffusion equation for the transmembrane voltage, coupled to ODEs defining the evolution of gating variables related to ion transport:

$$\begin{aligned} v_t &= \nabla \cdot \sigma \nabla v - I_{\text{ion}}(v,w) + I_{\text{e}} && \text{in } \Omega \times (0,T) \\ w_t &= G(v,w) && \text{in } \Omega \times (0,T), \end{aligned}$$

together with homogeneous Neumann boundary conditions and suitable initial conditions. The weak formulation is given by

$$\int_\Omega \left(v_t + I_{\text{ion}}(v,w) - I_{\text{e}}\right)\varphi + \sigma \nabla v \cdot \nabla \varphi\ dx\ dt = 0 \quad \forall \varphi \in H^1(\Omega) \text{ and a.a. } t \in (0,T)$$

$$\int_\Omega \left(w_t - G(v,w)\right)\psi = 0 \quad \forall \psi \in L^2(\Omega) \text{ and a.a. } t \in (0,T).$$

Weak solutions v, w satisfying this system belong to the spaces

$$\begin{aligned} &v \in C^0(0,T;L^2(\Omega)) \cap L^2(0,T;H^1(\Omega) \cap L^p(\Omega \times (0,T)),\ 2 \le p \le 6 \\ &w \in C^0(0,T;L^2(\Omega)), \end{aligned}$$

see, e.g., [74, 76]. Here, the time-dependent external current stimulus $I_\mathrm{e} \in L^2(0,T)$ acts as the control. The objective functional

$$J(v, I_\mathrm{e}) = \frac{1}{2}\int_0^T \int_{\Omega_\mathrm{obs}} v^2 \, dx \, dt + \frac{\alpha}{2}\int_0^T I_\mathrm{e}^2 \, dt$$

aims at dampening out the electrical excitation. We refer to Chapter 6 for more details and numerical results.

Next, we derive optimality conditions for the abstract optimal control problem given by equation (2.1).

2.2. Optimality Conditions

This section is based on the textbooks [55, 127, 8]. A more general discussion of optimality conditions can be found, e.g., in [89].

To derive optimality conditions, we assume that the state equation $c(y,u) = 0$ possesses a unique solution $y = y(u) \in Y$ for each control $u \in U$. We additionally assume that $c_y(y,u) : Y \to Z^\star$ is continuously invertible. Then, by the implicit function theorem (see, e.g., [143, Section 4.7]), the control-to-state mapping is continuously differentiable, and the derivative $y'(u)$ is given by the solution of

$$c_y(y,u)y'(u) + c_u(y,u) = 0. \tag{2.4}$$

By inserting $y(u)$ into the optimal control problem (2.1) we arrive at the *reduced* problem

$$\min_{u\in U} j(u) := J(y(u), u). \tag{2.5}$$

In this unconstrained setting, the following simple first order necessary optimality condition holds.

Theorem 2.2.1. *Let the above assumptions hold. If $u^\star \in U$ is a local solution of the reduced problem (2.5) it satisfies $j'(u^\star) = 0$. If additionally the reduced functional j is convex, this condition is also sufficient.*

Proof. See [55, Thm. 1.48]. □

Remark 2.2.2. If we allow control constraints, i.e. demand $u \in U_{\text{ad}}$ with $U_{\text{ad}} \subset U$ non-empty, convex and closed, the optimality condition changes to the *variational inequality* for the local minimizer $u^\star \in U_{\text{ad}}$

$$\langle j'(u^\star), u - u^\star \rangle_{U^\star,U} \geq 0 \quad \forall u \in U_{\text{ad}}. \tag{2.6}$$

As this generalization has no influence on the storage demands we restrict ourselves to the unconstrained case.

Further we recall the following second order conditions, which can be found, e.g., in [8, Thm. 2.12].

Theorem 2.2.3. *Let the reduced functional j be two times continuously Fréchet differentiable. If $u^\star \in U$ is a local solution of the reduced problem* (2.5) *it satisfies*

$$j''(u^\star)(\delta u, \delta u) \geq 0 \quad \forall \delta u \in U.$$

Theorem 2.2.4. *Let the reduced functional j be two times continuously Fréchet differentiable. Assume that $u^\star \in U$ satisfies $j'(u^\star) = 0$. Moreover, assume that there exists a real constant $\gamma > 0$ such that*

$$j''(u^\star)(\delta u, \delta u) \geq \gamma \|\delta u\|_U^2 \quad \forall \delta u \in U.$$

Then $u^\star$ is a local solution of the reduced problem (2.5).

Remark 2.2.5. The optimality conditions were presented in the most simple setting. Sufficient differentiability is often not clear, and a Hilbert space setting may be not given in applications. More elaborate techniques to derive optimality conditions are discussed e.g. in [65].

To formally derive a representation for the reduced gradient, we define the *Lagrange* functional $\mathcal{L} : Y \times U \times Z \to \mathbb{R}$,

$$\mathcal{L}(y, u, \lambda) = J(y, u) + \langle \lambda, c(y, u) \rangle_{Z,Z^\star}. \tag{2.7}$$

Clearly, inserting $y = y(u)$ into (2.7), we get $j(u) = \mathcal{L}(y(u), u, \lambda)$ for arbitrary $\lambda \in Z$. Differentiation in direction $\delta u \in U$ yields

$$\langle j'(u), \delta u \rangle_{U^\star,U} = \langle \mathcal{L}_y(y(u), u, \lambda), y'(u)\delta u \rangle_{Y^\star,Y} + \langle \mathcal{L}_u(y(u), u, \lambda), \delta u \rangle_{U^\star,U}.$$

Choosing $\lambda = \lambda(u)$ such that

$$\mathcal{L}_y(y(u), u, \lambda(u)) = J_y(y(u), u) + c_y(y(u), u)^\star \lambda(u) = 0 \tag{2.8}$$

yields

$$j'(u) = \mathcal{L}_u(y(u), u, \lambda(u)) = J_u(y(u), u) + c_u(y(u), u)^\star \lambda(u). \tag{2.9}$$

Equation (2.8) is the *adjoint equation*.

Remark 2.2.6. In the Hilbert space setting, the reduced gradient $\nabla j \in U$ is defined as the Riesz representative of the reduced derivative $j'(u) \in U^\star$, i.e. via

$$\big(\delta u, \nabla j(u)\big)_U = j'(u)\delta u \ \forall u \in U.$$

For better readability we use the notation $j'(u)$ for both, reduced gradient and derivative, as the meaning is usually clear from the context. The only exception is the analysis of the steepest-descent method in Section 5.2.

In the setting of parabolic optimal control problems, the adjoint equation (2.8) is backward in time. Due to the occurrence of $-J_y(y(u), u)$ as a source term, and—in the nonlinear case—the dependence of $c_y(y(u), u)$ on the state solution $y(u)$, adjoint gradient computation consists of three steps (see also Figure 2.1):

1. solve $c(y, u) = 0$ for $y \in Y$ and *store* the solution trajectory
2. solve $c_y(y, u)^\star \lambda = -J_y(y, u)$ for $\lambda \in Z$
3. set $j'(u) = J_u(y, u) + c_u(y, u)^\star \lambda$.

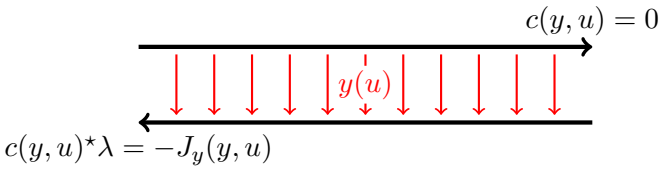

Figure 2.1.: Adjoint gradient computation

Example 2.2.7. We continue Example 2.1.1. Using the above formalism, the reduced gradient is given as

$$j'(u) = \alpha u + \lambda_{|\partial\Omega}$$

with the adjoint variable λ solving

$$-\lambda_t - \Delta\lambda = y - y_d \text{ on } \Omega \times (0,T), \quad \partial_\nu \lambda + \lambda = 0 \text{ on } \partial\Omega \times (0,T), \quad \lambda(\cdot, T) = 0 \text{ in } \Omega.$$

Proceeding analogously, we can derive a representation for the reduced Hessian $j''(u)$, see, e.g., [54]:

$$j''(u) = T(u)^\star \begin{pmatrix} \mathcal{L}_{yy}(y,u,\lambda) & \mathcal{L}_{yu}(y,u,\lambda) \\ \mathcal{L}_{uy}(y,u,\lambda) & \mathcal{L}_{uu}(y,u,\lambda) \end{pmatrix} T(u), \tag{2.10}$$

with

$$T(u) = \begin{pmatrix} -c_y(y,u)^{-1} c_u(y,u) \\ \mathrm{Id}_U \end{pmatrix},$$

where $\mathrm{Id}_U : U \to U$ denotes the identity operator.

Since for practical applications, the construction of the complete Hessian matrix is infeasible, the representation (2.10) is used to compute Hessian-vector products, e.g. during a Newton-CG method. This is discussed in more detail in Section 5.1.2.

2.3. Discretization

In this work we follow the *first optimize, then discretize* approach. To implement optimization methods based on the optimality conditions of the previous section, we need to discretize the arising parabolic PDEs in time and space. This is done using the *method of time layers* (also known as *Rothe's method*), so we discretize *time first*. The resulting sequence of elliptic partial differential equations are discretized in space by finite elements. This discretization order is especially suited for full adaptivity in time and space.

We only give a short summary of the techniques, mainly to fix the notation used throughout this thesis. For more details we refer to the textbooks [9, 32] on finite elements, as well as [29], where a special focus is on adaptivity.

Discretization in time and space is exemplarily described for the state equation; other arising PDEs, like the adjoint equation, are discretized similarly, possibly with different spatial and temporal meshes, to account for different dynamics.

Time Discretization

We consider discretization of the parabolic state equation by a time stepping scheme on a (not necessarily uniform) temporal grid $0 = t_0 < \cdots < t_F = T$. For the numerical experiments, we choose the linearly implicit Euler method. In abstract form our model problem (RDS) can be generalized to

$$By_t = F(y), \tag{2.11}$$

where we assume that the operator B is independent of y. For our examples this always holds true; for the generalization of non-constant B we refer to [85]. In the abstract PDE (2.11), we left aside the dependence of F on the control u, as for the moment we are only interested in the PDE discretization. We assume here that

u can be evaluated at arbitrary coordinates (x, t), e.g. by interpolation of a fixed discretization of the control.

One step of length τ of the linearly implicit Euler method is realized by

$$\begin{aligned} (B - \tau F_y(y^0)\delta y_k &= \tau F(y_k) \\ y_{k+1} &= y_k + \delta y_k, \end{aligned} \tag{2.12}$$

where y^0 denotes the linearization point.

To increase the order, this linearly implicit Euler method is combined with τ-extrapolation, as realized, e.g., by the code LIMEX [25, 27]. The step length $\tau = t_{k+1} - t_k$ is subdivided into successively smaller timesteps $\tau_j = \tau/n_j, j = 1, \ldots, j_{\max}$, where the harmonic sequence $\{n_j\} = \{1, 2, \ldots\}$ is used. Equation (2.12) with step size τ_j is used to compute approximations $Y_{j,1}$ to y_{k+1}. The extrapolation tableau defining the higher order approximations $Y_{j,k}$ is given by

$$Y_{j,i} = Y_{j,i-1} + \frac{Y_{j,i-1} - Y_{j-1,i-1}}{n_j/n_{j-i+1} - 1}, \quad j = 1, \ldots, j_{\max},\ i = 1, \ldots, j.$$

Instead of choosing a fixed number of extrapolation stages (determined above by $j_{\max}$), an acceptance test can be realized by taking $\|Y_{j,j} - Y_{j,j-1}\|$ as an error estimate, and accepting $Y_{j,j}$ as an approximation to y_{k+1} when a prescribed tolerance is met.

For more details, like adaptive order and step size control, we refer to [22, 99, 31] as well as the textbooks [24, 29].

Space Discretization

At each timestep, the arising elliptic sub-problems are discretized with linear finite elements on a hierarchical mesh. For ease of exposition we abstain from discussing higher order finite elements. Further, we assume in the following that $\Omega \subset \mathbb{R}^d,\ d = 2, 3$ is polygonally bounded.

We consider a nested family $\mathcal{T}_0 \subset \cdots \subset \mathcal{T}_l$ of triangulations, constructed from an initial triangulation $\mathcal{T}_0$ with

$$\overline{\Omega} = \bigcup_{c \in \mathcal{T}_0} .$$

To be more precise, $\mathcal{T}_j$ is generated by j levels of refinement, either uniform, or adaptively via a-posteriori error estimators. We refer to j as the *level* of the triangulation $\mathcal{T}_j$, and accordingly to $\mathcal{T}_j$ as the level-j-grid. The triangulation is supposed to be conforming, i.e. for two distinct cells $c_1, c_2 \in \mathcal{T}_j$, the intersection $c_1 \cap c_2$ is either

empty, a vertex, an edge, or a complete face. If Ω was not polygonally bounded, we would need to allow curved faces for cells on the boundary. Let S_j be the space of piecewise linear finite elements over the triangulation $\mathcal{T}_j$,

$$S_j = \{y \in C^0(\overline{\Omega}) \mid y \text{ is a linear polynomial on each } T \in \mathcal{T}_j\}. \tag{2.13}$$

The nested triangulations give rise to nested finite element spaces

$$S_0 \subset S_1 \subset \cdots \subset S_l \subset V,$$

with a suitable function space V (depending on the PDE to be solved). The set of nodes on level j is denoted by $\mathcal{N}_j$, in the following we will sometimes write $k \in \mathcal{N}_j$ instead of $x_k \in \mathcal{N}_j$. With the nodal basis of S_l,

$$\Phi = \{\varphi_i \mid i = 0, \dots |\mathcal{N}_l| - 1\}, \quad \varphi_i(x_k) = \delta_{i,k} \text{ for } x_k \in \mathcal{N}_l, \tag{2.14}$$

the PDE solution $y(x,t)$ at a fixed timestep t is represented as

$$y(x,t) = \sum_{k \in \mathcal{N}_l} y_k(t)\varphi_k(x). \tag{2.15}$$

As this section is mainly used to fix notation, we end the discussion here, and refer to the literature for topics like adaptive mesh refinement, e.g. [29] and the references therein. Some results on approximation theory are stated in Section 4.2, where they are required for the derivation of a-priori estimates for the compression factor.

3. Compression in Scientific Computing and Optimal Control

As we have seen in the previous chapter, for adjoint gradient computation the state trajectory over the whole time interval $[0, T]$ is required, together with the adaptively refined spatial grids. Also, for post-processing algorithms, visualization, or archiving of results, the efficient storage of simulation results is important. For accurate results, often very fine discretizations are needed, leading to large amounts of data to be stored. In this chapter, we discuss various techniques to reduce the memory requirements, both bandwidth and size.

A primary criterion to judge the quality of compression methods is the *compression factor*, which is defined as the ratio between uncompressed and compressed storage size. Typically—but not in all cases—a reduction of memory size leads also to a similar reduction of the required memory bandwidth, as the amount of data transferred to and from the memory is reduced. Of course, when using lossy compression, where parts of the original information are discarded, the compression factor has to be discussed in relation with the induced error.

For the storage of scientific data, there exists a vast amount of literature. Of the various different approaches, we discuss a selection in the following. Section 3.1 is devoted to general-purpose compression methods for floating-point data. In computer graphics and visualization, (lossy) compression is a common tool; we briefly present some important methods in Section 3.2, before we come to methods specialized for optimal control problems. There, checkpointing methods are frequently used. We summarize the basic approach and recent work in Section 3.3. Additionally, in Section 3.4, we comment on two methods for solving optimal control problems with memory reduction as a side-effect, Model Reduction and Multiple Shooting.

Most parts of this survey are published in [38]. Some parts of the analysis of checkpointing are located in [141].

3.1. General Floating Point Compression

In this section, we discuss approaches for general-purpose floating point-compression, both lossless and lossy. While the list is by no means complete, the selected algorithms—from a multitude of available methods—give an overview over the past and ongoing research.

3.1.1. Lossless Methods

For lossless methods, the sole criterion for comparison of different approaches is the compression factor. The comparison depends on the test data sets used, which differ in the literature. Nevertheless, the reported compression factors are good indicators for the quality and applicability of the algorithms to our problem at hand.

FPC. In [11], Burtscher and Ratanaworabhan present the lossless, single-pass, linear-time compression algorithm FPC. It aims at compressing floating-point data with unknown internal structure, with maximizing throughput, i.e. compression speed, as the main objective. Sequences of double-precision floating-point values are processed by predicting a value, determining the prediction error by an `XOR` operation, and compressing the result.

As predictors, "finite context method predictors" (fcm, [107]) and "differential finite context method predictors" (dfcm, [36]) are used. They count the occurrences of a value following a certain pattern (*context*) of preceding values, such that prediction is essentially a hash-table look-up to determine which value followed the last time a given sequence of values occurred. If the predicted value is close to the true value, the `XOR` operation produces many leading zeros. The number of leading zeros is encoded in a 3-bit value, which is stored together with a single bit specifying the chosen predictor and the remaining non-zero bytes of the prediction error. The reported compression factors range between 1.02 and up to 15.05 (for one special test data set), the geometric mean compression factor is 1.2–1.9 depending on the size of the look-up table for the predictors.

fpzip. While FPC uses no information about the structure of the data, the algorithm fpzip by Lindstrom and Isenburg, based on [82], traverses the data in some coherent order, and uses the Lorenzo predictor [60] to estimate values based on a subset of the already encoded data. Row-by-row traversal of the data works especially well for data on structured, cartesian grids. The predicted and true value is mapped from floating-point to an integer representation. While fpzip is foremost

a lossless compression algorithm, it can be run in a lossy mode. Then, during the mapping stage, the least significant bits are discarded, reducing the precision to 48, 32 or 16 bits/value, without control of the quantization error. The integer residual is stored using range coding [87], a variant of arithmetic coding. Lossless compression factors of 1.4–2.7 for a double precision test data set are reported in [82], with a average factor of approximately 1.6.

3.1.2. Lossy Methods

As expected, lossless methods can not reduce the amount of data significantly, due to many quasi-random least significant bits. To achieve good compression ratios, we have to resort to lossy compression techniques. Typically, the accuracy is reduced by *quantization* of the true values, or of predicted values, which essentially is rounding. Here, control of the quantization error is of crucial importance.

Comparison criteria for lossy methods are the compression factor in relation with the induced error. The different test data sets given in the literature, together with the different error norms used to report the quantization errors, make it difficult to give a quantitative comparison of the algorithms described below.

Adaptive coarsening/sub-sampling. This method by Shafaat, Baden, and coworkers [111, 129] is closely related to adaptive mesh refinement. Starting from simulation results on some fine, uniform mesh, the mesh is tentatively coarsened. After reconstructing the solution, grid points are removed where the data is represented on the coarser mesh with sufficient accuracy. This procedure is carried out recursively on the new coarser meshes, until no further coarsening is possible without violating the error bound. The result is an adaptive mesh representing the data up to a specified accuracy. As no quantization is used, compression is solely achieved by adaptivity. If the simulations are carried out using standard adaptive mesh refinement during the solution process, data reduction is only possible, if the necessary accuracy for solution and post-processing differ, like for adjoint gradient computation. In [129] the reported compression factors range between 7.44 (3D data) and 25.1 (2D data) for a pointwise relative ℓ^∞-error of 10^{-3}.

Graph Decomposition. In a recent work, Iverson, Kamath and Karypis [66] propose a compression algorithm based on the decomposition of the computational grid into so-called ε-bounded sets. The method works on structured and unstructured meshes, which are modeled via a graph. The nodes of the graph are the grid vertices for which values are computed. These vertices are partitioned into non-overlapping

sets V_i, such that each set contains only vertices with values differing at most by a specified ε. In each set V_i, the values are replaced by the mean value of the set, such that the point-wise absolute error is bounded by ε. If there is local smoothness in the data, this substitution increases the redundancy of the data, which is afterwards compressed using standard lossless compression methods. For a testset consisting of data on structured and unstructured grids with between 486 015 and 100 663 296 vertices, they report average compression ratios between 20 and 50 for pointwise relative ℓ^∞-errors of orders 10^{-2} to 10^{-3}.

ISABELA. Lakshminarasimhan et al. [78, 79] propose a method for "In situ Sort-And-B-spline Error-bounded Lossy Abatement" (ISABELA), specifically designed for spatio-temporal scientific data that is inherently noisy and random-like. In the spatial domain, data is sorted from an irregular signal to a smooth monotonous curve. Then a B-spline is fitted to the sorted data, the difference between data and fitted curve is quantized and stored, together with the information necessary to invert the sorting process. Their experience suggests that the ordering of the sorted data is similar between adjacent timesteps such that delta-encoding can be used to compress the ordering information. The accuracy of the reconstructed data is reported by two quantities, the normalized root mean squared error (NRMSE), and Pearson's correlation coefficient ρ defined by

$$\mathrm{NRMSE} = \frac{\left(\sum_i (D_i - \hat{D}_i)^2\right)^{\frac{1}{2}}}{\max(D) - \min(D)}, \qquad \rho = \frac{\mathrm{cov}(D, \hat{D})}{\sigma(D)\sigma(\hat{D})},$$

where D denotes the original data, $\hat{D}$ the de-compressed data, and σ the standard deviation. In [79] they report compression factors between 3.8 and 5.6 for error bounds $\rho > 0.99$ and $\mathrm{NRMSE} < 0.01$, and five different data sets.

FEMZIP. FEMZIP [125, 124] is a commercial tool for the compression of finite element solutions created by certain finite element programs. After a quantization step with prescribed relative or absolute tolerance, a prediction step follows. In space, a hierarchical approximation of the finite element functions is performed, using coarsening of the computational grid by algebraic multigrid techniques [125]. In time, prediction based on rigid body movements is used. As a final step, the approximation residual is compressed using arithmetic encoding. Compression factors of up to 13.3 are reported [124], but without quantitative specification of the accuracy.

Application-specific methods. To conclude this section, we exemplarily mention two methods developed for specific applications. In [110], Schröder-Pander et al. use

generalized multiresolution analysis to develop algorithms for compressing and analyzing cell averages used in finite volume methods for the solution of systems of hyperbolic conservation laws. They include a lossy step by discarding coefficients smaller than some prescribed tolerance. For two test functions they achieve compression factors of 3–7 for relative L^∞-errors of approximately 10^{-4}, and factors 13–53 for relative L^∞-errors of approximately 10^{-2} (see Section 6.1 for the test functions and comparison with the approach developed in this thesis).

Another data compression technique for computational fluid dynamics was recently developed by Rettenmeier [101]. There, after a quantization step, two different de-correlation methods are specified and compared. The first method uses the connectivity of the mesh to predict coefficients, where the traversal order is based on spanning trees of the connectivity graph. The second de-correlation technique is based on an integer wavelet transform, utilizing algebraic multigrid methods to create a multiresolution setting from a single-resolution mesh. They report compression factors ranging between 4 and 30 for a testset containing data from several large-scale simulations, and a fixed quantization tolerance 10^{-3}. In some special cases factors of up to 40 were achieved.

3.2. Computer Graphics and Visualization

For the compression of general, possibly time-varying, data on unstructured grids as needed in computer graphics and visualization, the combination of prediction and lossy encoding of the prediction errors is a common approach, see, e.g., [61, 81, 134]. As there is a huge amount of work in this area, here we shortly discuss a selection of methods only, and refer to the survey [3] by Alliez and Gotsman for a more detailed overview.

3.2.1. Mesh Compression

Triangular meshes consist of two types of information: connectivity, i.e. the triangle-vertex incidence graph, and geometry, i.e. the coordinates of the vertices. For *connectivity encoding*, lossless compression methods are of main interest, to be able to reconstruct the topology of the meshes exactly. Edgebreaker by Rossignac [105] and its variant optimized Edgebreaker by Szymczak [123] are prominent algorithms, making use of mesh regularity to create a compressed representation of the connectivity, based on a certain traversal of the mesh. Other well-known, successful methods are by Touma and Gotsman [126], or the more recent early-split coder by Isenburg and Snoeyink [63].

For *geometry encoding* wavelet transforms are usually used, either based on mesh coarsening (e.g. Isenburg and Snoeyink [62]) or subdivision (e.g. Lounsbery, DeRose, and Warren [84]). Here, typically lossy encoding is used, with the wavelet coefficients being uniformly quantized to achieve a fixed bit-rate. Exemplarily we mention the well-known "progressive geometry compression" method by Khodakovsky, Schröder, and Sweldens (PGC, [73]), which is based on a wavelet transform, combined with zerotree coding [113].

Improvements can be achieved by exploiting correlation between connectivity and (already encoded) geometry information, as is done e.g. by the code Angle-Analyzer of Lee, Alliez, and Desbrun [80] for quadrangular meshes, and FreeLence by Kälberer et al. [68] for triangular meshes. For a root mean square error of order 10^{-4} to 10^{-5} with respect to the bounding box diameter, average compression factors of 21 for a test-set of irregular triangle meshes are reported for FreeLence. By exploiting the mesh hierarchy and using context-based entropy coding, von Tycowicz et al. report an average compression factor of 29 for a test-set consisting of adaptively refined hierarchical meshes [136].

3.2.2. Compression of Time-Varying Data

For the compression of animated meshes, elaborate algorithms exist, e.g. FAMC [86, 116]. FAMC is based on motion-compensation for a sequence of frames with identical mesh connectivity. In our setting, the grids for the PDE solution are typically generated adaptively, with a varying connectivity at each timestep. Additionally, as in most PDE solutions there are no distinct moving objects, the benefit of motion compensation for optimal control of PDEs is questionable.

The most popular lossy compression approach to time-varying data on equidistant, cartesian grids can be found in the MPEG video compression standard [92]. Videos consist of a series of single frames showing spatial and temporal correlations. The spatial correlations are reduced by the discrete cosine transform applied to blocks of typically 8×8 or 16×16 pixels. The resulting coefficients are then quantized in a way to maintain a certain optical quality of the video. For example, the human visual system is more sensitive to low spatial frequencies than high spatial frequencies, allowing for a coarser quantization of high frequency components. Quantization is done by dividing the coefficients by predefined factors and a rounding step. Motion prediction is performed to construct a frame from previous (and possibly later) frames, as mostly only small changes occur from one frame to the next. Typically, the encoding process, in particular the motion compensation, is rather time-consuming. There is a vast amount of literature on video compression, we refer to [92, 102, 121] and the references therein.

3.3. Checkpointing

So-called checkpointing methods are a tool for the computation of the reduced gradient using the adjoint equation, first introduced by Volin and Ostrovskii [135], and Griewank [45]. Instead of keeping track of the whole forward trajectory, only the solution at some intermediate timesteps is stored. During the integration of the adjoint equation, the required states are re-computed, starting from the snapshots. Typically, for the analysis of checkpointing methods it is assumed that each checkpoint has the same size. This means that only fixed grids are considered for discretization in space.

3.3.1. Fixed Timesteps

During the forward simulation, the algorithm has to decide when to create a checkpoint. In the simplest setting, the temporal mesh is fixed as well as the spatial grid, and the checkpoint distribution can be computed beforehand (*offline* checkpointing). In the following we denote by c the total number of checkpoints, and by n_t the total number of timesteps of the time discretization.

One obvious strategy would be a to place checkpoints uniformly over the time interval, a technique also known as *windowing* [5]. Recursive application of this strategy to each group of timesteps between two checkpoints results in a *multilevel checkpointing* strategy [5, 47]. Both techniques do not yield optimal distributions, i.e., distributions leading to a minimal amount of re-computations. *Binomial checkpointing* [45, 46] is based on the fact that the maximal number of timesteps $\beta(c, r)$ that can be reversed fulfills

$$\beta(c, r) = \binom{c + r}{c},$$

when c checkpoints and at most r re-computations of each timestep are allowed. Via dynamic programming one can evaluate the minimal extra number of forward steps $t(n_t, c)$ necessary to compute the adjoint using c checkpoints as

$$t(n_t, c) = rn_t - \beta(c + 1, r - 1),$$

where r is the unique integer satisfying $\beta(c, r-1) < n_t \leq \beta(c+1, r-1)$, see [46, 47]. An implementation called revolve by Griewank and Walther [46] is available. The number r, the so-called repetition number, is the maximum number of times a single forward timestep is computed, and thus an upper bound of the cost of checkpointing relative to the cost of a forward solve. For a given number of checkpoints, fixing r determines the maximum number of timesteps that can be reversed using binomial checkpointing.

The achieved compression factor for storage space is given by n_t/c. In Figure 3.1, the resulting increase in runtime for a range of compression factors is shown, for a fixed number of timesteps ($n_t = 100$) and varying number of checkpoints.

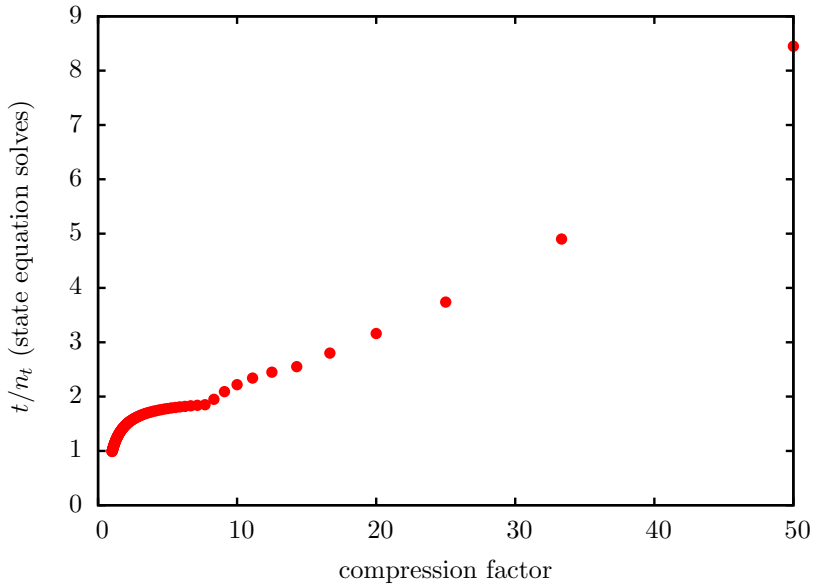

Figure 3.1.: Relative work vs. compression factor for checkpointing, $n_t = 100$ timesteps.

One appealing feature of checkpointing is the slow growth of the relative work for an increasing number of timesteps. For $r \gg c = \text{const}$, $r \approx n_t^{1/c}$ [47]. This is very satisfactory for reversing a large number of timesteps, like in algorithmic differentiation, where every single arithmetic operation has to be reversed. In the context of optimal control of time-dependent PDEs, however, the number of forward timesteps is often rather small in comparison, such that the excellent limit behavior of checkpointing is not that crucial.

Due to multiple read- and write-accesses of checkpoints during the re-computations for the adjoint equation, the reduction in memory bandwidth requirements is significantly smaller. An evaluation of the number of times a snapshot is written or read can be found in [119]. There Stumm and Walther analyze a multistage approach, where some checkpoints are kept in RAM, others written to a hard disk drive or tape.

In Figure 3.2, the computed write counts are shown for $n_t = 1000$ timesteps, and

$c = 50, \ldots, 100$ checkpoints, leading to compression factors between 10 and 20.

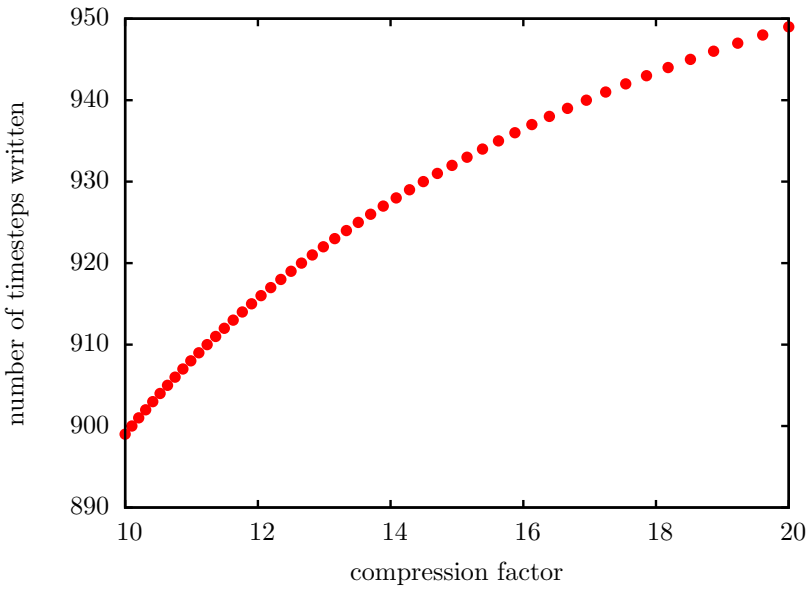

Figure 3.2.: Actual write accesses for checkpointing, $n_t = 1000$ timesteps.

It is apparent that the amount of data transferred to the storage device, and hence the memory bandwidth, is barely reduced. Evaluating the write counts for instance for $n_t = 1000$ timesteps, and $c = 50$ checkpoints, i.e. compression factor 20, shows that only about 5% reduction of memory bandwidth is achieved for these parameters. In this example we get $r = 2$, and the computational overhead amounts to 1948 additional forward steps. However, there are settings for which a reduction in memory bandwidth actually is achieved, e.g. for $r = 2$ and $n_t \leq 2c + 1$. For such a setting, the store-everything approach, i.e. writing all timesteps of the forward solution to disk, turns out to be more expensive in terms of computation time than checkpointing, despite the need for re-computations, see [119]. Moreover, frequently accessed checkpoints can possibly be kept in RAM, decreasing the runtime. Clearly, memory bandwidth has a significant impact on the computational efficiency of the algorithm.

Here, we assumed that each timestep has the same computational cost; in case of non-uniform timestep cost, optimal checkpoint distributions can be evaluated in $\mathcal{O}(cn_t^2)$ if the timestep costs are known a-priori [138], or generated using heuristics [117].

3.3.2. Adaptive Timesteps

If the number of timesteps is not known beforehand, the optimal checkpoint distribution can not be computed. Thus, in practical applications the user has to resort to *online* placement of checkpoints during the state integration

An extension of the revolve algorithm, named a-revolve, is proposed by Hinze and Sternberg [56, 117], and applied to optimal control of the Navier-Stokes equations. There, a heuristic strategy to overwrite the contents of a previously recorded checkpoint is developed, based on estimates of the computational cost for the reversal of the current and the updated snapshot distribution. While the resulting scheme is not proven to be optimal, numerical experiments indicate that the generated checkpoint distribution is close to the corresponding offline one.

Other work on online checkpointing was started by Heuveline and Walther [53], with extensions and theoretical foundations by Stumm and Walther [120]. The approach presented there is proven to be optimal in terms of re-computations for repetition number $r = 2$ and $n_t \leq \beta(c, 2)$. For $r = 3$ and $\beta(c, 2) < n_t \leq \beta(c, 3)$ optimal checkpoint distributions can not be computed, but for a wide range of timesteps n_t, the resulting algorithm is close to optimal. The method works by continuously overwriting certain previously set checkpoints, until the end of the state integration. For re-computations during the adjoint integration, intermediate snapshots are stored using optimal offline checkpointing.

A different strategy for choosing which checkpoints to replace is devised by Wang, Moin, and Iaccarino [139]. Although their algorithm, called *dynamic checkpointing*, works for an arbitrary number of timesteps n_t, the resulting distribution has just an optimal repetition number r, but is not optimal in terms of the total number of re-computations.

For all three methods the reduction in memory bandwidth is drastically smaller than the reduction in storage space. In fact, due to the frequent overwriting of snapshots, it is questionable if a reduction of bandwidth requirements can be achieved at all.

3.3.3. Discussion

Checkpointing is a compression method, which originally was developed for computation of gradients via the reverse mode of automatic differentiation, where a large number of arithmetic operations has to be reversed. In that context, two features are particularly important: checkpointing is lossless, and the additional computational work grows slowly for an increasing number of operations. For optimization with time-dependent differential equations as constraints, the second property is not

as important, as the number of timesteps is typically small compared to the number of arithmetic operations in automatic differentiation. The additional work, for typical settings two up to four additional solves of the state equation, carries more weight. Also, in terms of data transferred, only a small reduction of bandwidth can be achieved, in particular with online checkpointing.

When using second order optimization methods, like Newton-CG, the state trajectory is needed in each CG iteration to evaluate Hessian-times-vector products, leading to higher computational work, as typically checkpoints are overwritten during adjoint integration, and thus their original information is lost for the subsequent CG iterations and has to be re-computed as well.

For non-uniform timestep cost which is not known a-priori, checkpoint distributions have to be chosen heuristically. With adaptive mesh refinement, also the sizes of the snapshots are unknown a priori. For this case, no optimal checkpoint distributions are known, not even heuristics.

3.4. Other Techniques

In this section we briefly discuss two techniques for the solution of optimal control problems, with memory reduction as a side effect.

3.4.1. Model Reduction

Model reduction techniques focus mainly on the reduction of computational complexity via approximation of large-scale problems by smaller ones. First developed for handling parameter-dependent differential equations, in the last years this algorithm class is applied to optimal control and inverse problems as well. One popular method for the construction of reduced models is proper orthogonal decomposition (POD). There, a basis is computed from the solution of the state equation at a number of given timesteps by principal component analysis. If the involved eigenvalues decay quickly, comparatively few basis vectors are sufficient for a good approximation of the solution. A detailed analysis of POD methods for parabolic PDEs can be found in [75] by Kunisch and Volkwein, see Hinze and Volkwein [57] for the use of POD in optimal control. In terms of memory requirements, only the snapshots of the solution of the large-scale problem need to be stored.

Due to the reduced-order model, only sub-optimal controls can be computed. To judge the quality of the approximate solution, a-posteriori error estimators were developed. In [128], such an estimator is derived by Tröltzsch and Volkwein for

the linear-quadratic case, and extended to semilinear equations in [70]. For the evaluation of the error estimate, state and adjoint solutions of the full problem are needed, posing the same requirements for storage space as the original large-scale problem. A different technique is suggested by Jörres, Vossen, and Herty [67]: they use the full model to compute the gradient and only use reduced models to find a suitable steplength for the control update. Again, no reduction in memory size is achieved. While both methods reduce memory bandwidth, a combination with lossy trajectory compression for evaluation of error estimators or gradient computation appears attractive.

3.4.2. Multiple Shooting

Multiple shooting is a well established method for the solution of ODE boundary value problems. The time interval $[0, T]$ is decomposed in a number of sub-intervals, with auxiliary variables for the interfaces ensuring continuity of the solution. The resulting cyclic, nonlinear system of equations is typically solved using Newton's method. Details and a short overview of the history of shooting methods can be found in the textbook [24], for instance. In the last years, this principle was applied to solve optimal control problems governed by time-dependent partial differential equations by Heinkenschloss [49], Comas [17], as well as Hesse and Kanschat [51, 52], for instance. The decomposition of the time domain leads to optimization problems on the sub-intervals, where locally state and adjoint are implicit functions of the control and auxiliary variables. Sequential solution of the local problems leads to a storage reduction, as only the trajectory on the respective sub-interval is needed. The coupling of the sub-problems via the auxiliary variables ("matching conditions") avoids the disadvantage of moving horizon techniques, where only sub-optimal controls can be computed (see, e.g., [64]). Combination with adaptive mesh refinement is discussed in [51, 52, 13], where a dual weighted residual (DWR) method [6] is used for error estimation.

Although the resulting algorithms are easily parallelizable due to the splitting in local sub-problems, significant storage reduction is only achieved in sequential computations, or if the number of sub-intervals is considerably larger than the number of CPUs. Each CPU then has to provide storage only for the currently processed local problem, plus additional storage for the auxiliary variables. Again, a combination with lossy trajectory storage is an attractive possibility.

4. Lossy Compression with Pointwise Error Control

The review of compression methods in Section 3.1 indicates that lossless methods are not suited to significantly reduce memory requirements. Moreover, for simulation and optimal control we are interested in the actual physical problem, or, mathematically, in the infinite-dimensional function space solution. The latter, however, can not be achieved, as discretization, quadrature and iterative solution of linear equation systems have to be used. In view of the errors incurred by these numerical techniques, it seems evident to use lossy compression techniques to store the already inexact solutions, and derive means to control the additional error.

In this chapter, we develop and analyze a computationally inexpensive lossy compression method, adapted to the specific needs of PDE-constrained optimization. We aim at algorithms easily usable on unstructured, adaptively refined finite element meshes in two and three space dimensions, making use of the multilevel nature of such grids.

For the compression algorithm, the principle of *transform coding* (Figure 4.1) is used. It contains three main ingredients: a *transform*, e.g. realized by a *predictor*, *quantization* of the transformed coefficients, and *entropy coding*. While the transform—essentially a change of basis—reduces correlations in the data and is invertible, the quantization step reduces the accuracy of the data. For this operation, the inverse can only be approximated, leading to errors in the reconstructed values. By the lossless entropy coding step, symbols for the transformed and quantized coefficients are created and a bitstream is written to the storage medium.

For our specific problem, these steps are adapted as follows. First, a prediction step is used to construct an approximation to the finite element solution of the state equation at the current timestep. As we require the predictor to be cheap in terms of computational complexity, an inexact predictor is used. Spatial correlations are exploited using prolongation in mesh hierarchies, while temporal correlations are exploited by taking values from the next timestep into account. As the adjoint

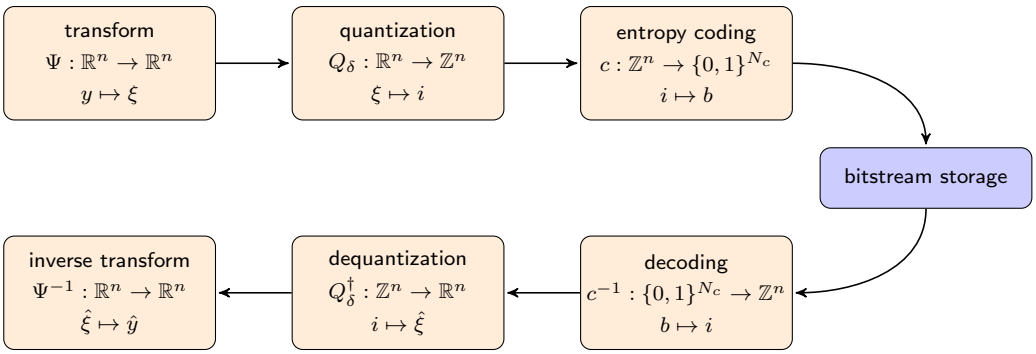

Figure 4.1.: Principle of transform coding

equation is integrated backwards in time, these values are available. Uniform quantization of the prediction error and entropy coding of the quantized values reduces the storage requirement at the price of a loss of information.

The remainder of this chapter is organized as follows. In Section 4.1, we make use of the hierarchical spatial grids to compress a finite element solution at a fixed timestep. Aiming at a quantization error of the same magnitude as the discretization error, in Section 4.2 we derive a-priori estimates for the achievable compression factors. For adjoint gradient computation, the state solution is only accessed backward in time, which we exploit for temporal prediction in Section 4.3. For basic encoding and decoding, prototype algorithms in a compact pseudo-code form can be found in Section 4.4.

Many of these results are published in [141].

4.1. Multilevel Compression in Hierarchical Meshes

In this section, we exploit spatial correlations at a fixed time point and turn to temporal correlations afterwards in Section 4.3. We discuss in detail each step of the transform coding procedure.

Quantization. The essential step of lossy compression is quantization.

Definition 4.1.1. For a given error bound $\delta > 0$, let $Q_\delta : \mathbb{R} \to \mathbb{Z}$ with

$$Q_\delta(y) := \left\lfloor \frac{y + \delta}{2\delta} \right\rfloor$$

denote the *quantization* and $Q_\delta^\dagger : \mathbb{Z} \to \mathbb{R}$ the *reconstruction* defined by

$$Q_\delta^\dagger(i) := 2\delta i.$$

Then the actual pointwise *quantization error* $|y - Q_\delta^\dagger(Q_\delta(y))|$ is bounded by δ.

Prediction. Values y_k of coarse level nodes $x_k \in \mathcal{N}_0$ are quantized directly to $i_k = Q_\delta(y_k)$ to be stored, yielding a reconstructed value $\hat{y}_k := Q_\delta^\dagger(i_k)$. For new nodes $k \in \mathcal{N}_j \backslash \mathcal{N}_{j-1}$ on level $j > 0$, we make use of the grid hierarchy and quantize and store only the deviation of y_k from a prediction $P_k(\hat{y}_m : m \in \mathcal{N}_{j-1})$ obtained from reconstructed values $\hat{y}_m$ of lower level nodes.

The most simple prediction P_k is linear interpolation between grid levels, which is the usual multigrid prolongation operator. For a node $k \in \mathcal{N}_j \backslash \mathcal{N}_{j-1}$ subdividing the edge between nodes m_1 and $m_2 \in \mathcal{N}_{j-1}$, we define

$$P_k(\hat{y}_{m_1}, \hat{y}_{m_2}) := \frac{1}{2}(\hat{y}_{m_1} + \hat{y}_{m_2}).$$

This prediction step changes the basis for the finite element coefficient vector from the standard nodal basis to the hierarchical basis. This observation is used in Section 4.2 to derive a-priori estimates for the achievable compression factors. Figure 4.2 illustrates the effect of the transformation: instead of storing nodal coefficients $y_k, k \in \mathcal{N}_l$ of the finest grid level, only the coarse grid values $y_k, k \in \mathcal{N}_0$ together with the hierarchical basis-coefficients ξ_k have to be stored.

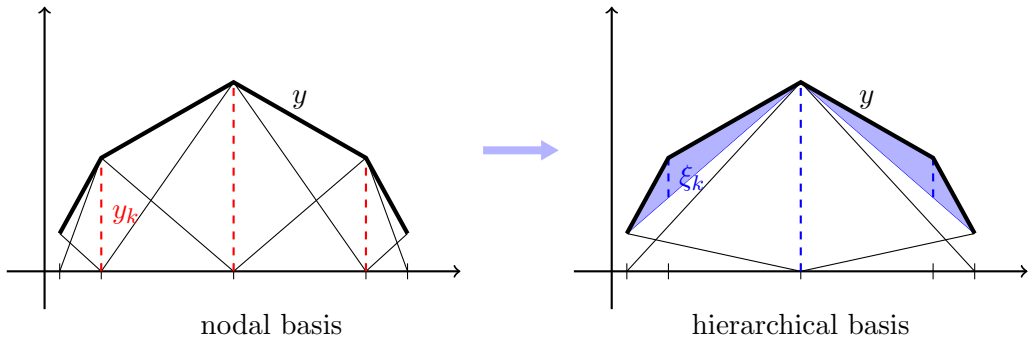

Figure 4.2.: Representation in nodal and hierarchical basis

Remark 4.1.2. Here we use a deliberately simple predictor working with the inherent hierarchical structure of the mesh. The prolongation can be realized using cheap matrix-vector multiplication, where the sparse prolongation matrices P_k have

already been computed during the mesh refinement and can be re-used. Of course more elaborate predictors, like higher order prolongation or wavelet transforms [84] could be used as well.

We thus obtain quantized values $i_k = Q_\delta(y_k - P_k(\hat{y}_{m_1}, \hat{y}_{m_2}))$ to be stored, and reconstructed values $\hat{y}_k = P_k(\hat{y}_{m_1}, \hat{y}_{m_2}) + Q_\delta^\dagger(i_k)$. The effect of quantizing the prediction error is that the quantized values i_k are clustered around 0, which allows to perform a highly effective entropy coding.

Note that predicting the value of y_k from the reconstructed values $\hat{y}_{m_1}$ and $\hat{y}_{m_2}$ instead of y_{m_1} and y_{m_2} allows the decompression routine to obtain the same predictions, as only information available to the decompression routine is used. A similar technique is used in MPEG video compression, where the reconstructed values (i.e. inverse quantization, inverse discrete cosine transform, and, in the MPEG-standard H.264/AVC, application of in-loop deblocking filters) are used for motion prediction in the encoder as well as in the decoder [121].

Remark 4.1.3. We have presented a predict-then-quantize approach. The order of prediction and quantization could be reversed, as is often done in geometry compression, leading to a quantize-than-predict scheme: After quantization of all nodal values, for a vertex k dividing the edge (m_1, m_2) the predictor then yields

$$\hat{y}_k = \frac{1}{2}\left((Q_\delta^\dagger(Q_\delta(y_{m_1})) + Q_\delta^\dagger(Q_\delta(y_{m_2}))\right),$$

and $Q_\delta(\hat{y}_k) - Q(y_k)$ is stored on disk. This leads to comparable results for compression rates and runtimes in most cases. For linear y, however, in the worst case $\hat{y}_k = y_k \pm \delta$, and generally $Q_\delta(y_k \pm \delta) \neq Q_\delta(y_k)$. When quantizing the residuals as proposed here, $Q_\delta(y_k \pm \delta - y_k) = Q_\delta(\pm\delta) = 0$, leading to significantly better compression rates. Numerical tests show that this is also true for mildly nonlinear functions, like $f_4(x)$ from Section 6.1. For more oscillatory functions, the quantize-then-predict approach performs a little better, see Table 6.1 in Section 6.1.

Remark 4.1.4. Choosing the quantization error bound δ individually for each node makes it possible to perform spatially nonuniform quantization and to bound the quantization error in weighted L^∞-norms. E.g., the quantization error bound could be determined from sensitivities

$$\frac{\partial \left\| e_{j'} \right\|^2}{\partial \delta},$$

where $e_{j'}$ denotes the gradient error. If these quantities are computable by both encoder and decoder, the interval bounds for each node need not be stored.

Entropy coding. The quantized coefficients $i_k, k = 0, \ldots, |\mathcal{N}_l| - 1$ are written to disk using range encoding [87]. According to their different frequency of occurrence, the coefficients are encoded with variable-sized symbols, assigning fewer bits to more frequent coefficients. As the frequency distribution has a peak at 0, this increases the compression factor. Since this peak becomes smaller for higher grid levels, entropy coding is performed for all levels individually, using the corresponding frequency distribution.

The frequency distributions can be computed and stored before encoding, or continuously updated during encoding and decoding. While storing the frequency distributions introduces a minor overhead in storage space, numerical experience shows that it may increase the performance of the range coder and thus the overall compression factor more than updating, at least for moderate problem sizes.

Reconstruction. With adaptive mesh refinement and time stepping, interpolation of the reconstructed state values in space and time will be needed for the adjoint solution. When storing the state values, the mesh needs to be stored additionally, see [69] for an efficient method. During the adjoint computation, at a given timestep first the corresponding state time needs to be found, $t_{\text{state}} = T - t_{\text{adjoint}}$. Since the time grids will in general be different, $y(t_{\text{state}})$ has to be approximated from the nearest state timesteps for which the solution was stored, e.g. by constant or linear interpolation. Secondly, the state mesh needs to be restored, such that prediction, de-quantization and correction is performed at the correct nodes. In space, the reconstructed finite element solution of the forward equation can be evaluated by interpolation.

4.2. A-Priori Estimates

As seen in Section 4.1, the trade-off between compression and quantization error depends on the range of prediction errors on each level. For ease of presentation, we look at a simple model problem on a 2D domain. As in this section we are concerned with spatial prediction only, we leave out the time dependence, assume $y \in W^{2,\infty}(\Omega)$, and use linear interpolation for prolongation. The semi-norm $|\cdot|_{2,\infty,\Omega}$ is given by

$$|y|_{2,\infty,\Omega} = \max_{|\alpha|=2} \|\partial^\alpha y\|_{L^\infty(\Omega)} ,$$

with a multi-index α.

Let $\Omega \subset \mathbb{R}^2$ be a polygonal domain, and $\mathcal{T}_0, \ldots, \mathcal{T}_l$ a nested family of triangulations of Ω as before, with $\mathcal{T}_j$ generated from $\mathcal{T}_0$ by j *uniform* refinement steps,

i.e. every triangle on level $j-1$ is subdivided into four congruent triangles in the jth refinement step. The maximum diameter of a triangle on level j is given by $h_j = \max_{T\in\mathcal{T}_j} \operatorname{diam}(T)$.

From standard finite element theory, e.g. [32], an estimate for the interpolation error is known:

Lemma 4.2.1. *Let $\mathcal{T}_j$ be a shape-regular family of triangulations of a polyhedral domain Ω, and denote by $I_j := I_{h_j}$ the interpolation operator with linear polynomials. Then for $y \in W^{2,\infty}(\Omega)$,*

$$\|y - I_j y\|_{L^\infty(\Omega)} \leq c h_j^2 |y|_{2,\infty,\Omega}. \tag{4.1}$$

For uniform refinement, $h_j = h_0 2^{-j}$ with given constant initial mesh-width h_0. For an a priori estimate of the error-to-compression ratio of the lossy compression algorithm, we are interested in the prediction error on level $j > 0$,

$$\|I_j y - I_{j-1} y\|_{L^\infty(\Omega)},$$

as the range of the prediction error determines the number of bits needed to keep a given error bound.

Lemma 4.2.2. *With the same assumptions as in Lemma 4.2.1, it holds*

$$\|(I_j - I_{j-1})y\|_{L^\infty(\Omega)} \leq 4 c h_0^2 \frac{1}{2^{2j}} |y|_{2,\infty,\Omega}, \tag{4.2}$$

with c independent of j.

Proof. With a generic constant c independent of h_j,

$$\begin{aligned}\|(I_j - I_{j-1})y\|_{L^\infty(\Omega)} &\leq \|y - I_{j-1} y\|_{L^\infty(\Omega)} = c h_{j-1}^2 |y|_{2,\infty,\Omega} \\ &\leq 4 c h_j^2 |y|_{2,\infty,\Omega} = 4 c h_0^2 \frac{1}{2^{2j}} |y|_{2,\infty,\Omega}.\end{aligned}$$

□

Let S_l be the space of piecewise linear finite elements over the family of triangulations defined above, and consider the hierarchical basis splitting $S_l = V_0 \oplus \cdots \oplus V_l$, with $V_j = \operatorname{span}\{\psi_{j,k} : k \in \mathcal{N}_j \backslash \mathcal{N}_{j-1}\}$. Here, $\mathcal{N}_{-1} = \emptyset$. With the notation introduced in Section 2.3 the hierarchical basis $\psi_{j,k}$ is given as follows:

$$\begin{aligned}\psi_{0,k}(x_i) &= \varphi_k(x_i), \quad x_i \in \mathcal{N}_0 \\ \psi_{j,k}(x_i) &= \varphi_k(x_i), \quad x_i \in \mathcal{N}_j \backslash \mathcal{N}_{j-1},\end{aligned}$$

see [142]. Hence, $y_h \in S_l$ can be written as

$$y_h = \sum_{j=0}^{l} \sum_{k \in \mathcal{N}_j \setminus \mathcal{N}_{j-1}} a_{j,k} \psi_{j,k}. \tag{4.3}$$

This decomposition yields the coefficients

$$a_{0,k} = (I_0 y)(x_k), \qquad x_k \in \mathcal{N}_0 \tag{4.4}$$
$$a_{j,k} = (I_j y - I_{j-1} y)(x_k), \quad x_k \in \mathcal{N}_j \setminus \mathcal{N}_{j-1}. \tag{4.5}$$

With Lemma 4.2.2 we can estimate the ℓ^∞-norm of the coefficients of the hierarchical basis representation on a given level $j > 0$,

$$\|(a_{j,k})_k\|_{\ell^\infty} \leq c 2^{-2(j-1)} |y|_{2,\infty,\Omega}. \tag{4.6}$$

Quantization is chosen such that a given error bound δ is maintained, yielding an interval length 2δ, and thus at most

$$\frac{2\left(4ch_j^2 |y|_{2,\infty,\Omega} + \delta\right)}{2\delta} = \frac{4ch_j^2 |y|_{2,\infty,\Omega}}{\delta} + 1 \tag{4.7}$$

different quantized values on a given level j. The additive factor 2δ in the numerator on the left is due to the inexact storage of the nodes on level $j-1$, which will in the worst case differ from the original nodes by this value, see Figure 4.3 for a sketch of the situation.

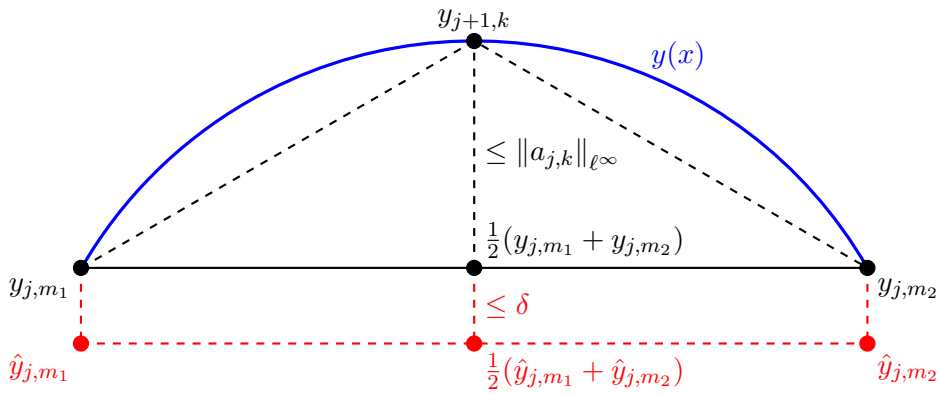

Figure 4.3.: Prediction and quantization error.

For reaching a given discretization accuracy, l refinements are needed. Allowing a quantization error of the same magnitude as the interpolation error yields

$$\delta = \|y - I_l y\|_{L^\infty(\Omega)} \leq c h_l^2 |y|_{2,\infty,\Omega}. \tag{4.8}$$

Thus, the number of quantized values on level j can be estimated as

$$\frac{4ch_j^2|y|_{2,\infty,\Omega}}{ch_l^2|y|_{2,\infty,\Omega}} + 1 = 2^{2(l-j+1)} + 1. \tag{4.9}$$

Each value can be stored using

$$\log_2(2^{2(l-j+1)} + 1) \leq 2(l-j+1) + \frac{1}{2^{2(l-j+1)}\ln(2)} \tag{4.10}$$

bits, where the estimate is due to the concavity of the logarithm. If higher accuracy is desired, the number of bits can be estimated by scaling δ in the previous computation.

Remark 4.2.3. The L^∞-approximation error for discretization by linear finite elements can be estimated as

$$\|y - y_h\|_{L^\infty(\Omega)} \leq ch^2 |\ln h| |y|_{2,\infty,\Omega}, \tag{4.11}$$

which behaves like $\mathcal{O}(h^{2-\varepsilon})$ for any $\varepsilon > 0$ [15, 32]. Hence, the number of bits needed for achieving discretization error accuracy will be slightly smaller than estimated above.

For a uniformly refined grid, there are approximately $c2^{dj}$ vertices on level j, with $c2^{dj} - c2^{d(j-1)}$ new vertices on that level. The overall number of bits needed can thus be estimated as

$$\begin{aligned}
&\sum_{j=1}^{l} c\left(2^{dj} - 2^{d(j-1)}\right)\left(2(l-j+1) + \frac{1}{2^{2(l-j+1)}\ln(2)}\right) + c\left(2(l+1) + \frac{1}{2^{2(l+1)}\ln(2)}\right) \\
&\qquad \approx c2^{dl}\left(\frac{(2^{3+2d} - 2^{d+1})\ln(2) + 2^{2d} - 2^{d+1} + 1}{(2^d - 1)(2^{d+2} - 1)\ln(2)}\right)
\end{aligned} \tag{4.12}$$

For the last estimate, terms with $-dl$ contributing to the exponent were dropped. With $c2^{dl}$ vertices in the finest mesh, the above estimate yields for $d = 2$ approximately 2.96 bits/vertex on average, which is a compression factor of 21.6 compared to storing double precision floating point data at 64 bit per value. In Figure 4.4, the resulting relation between error and compression is shown for 2D and 3D.

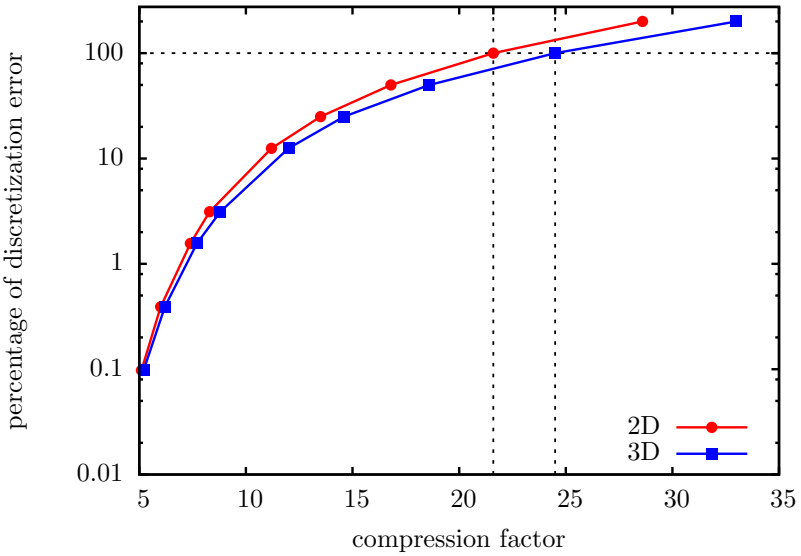

Figure 4.4.: Error vs. compression factor: a-priori estimates for spatial compression

Remark 4.2.4. If we just assume $y \in H^2(\Omega)$, it holds

$$\|y - I_h y\|_{L^\infty(\Omega)} \leq c h^{2-d/2} |y|_{H^2(\Omega)}$$

as $H^2(\Omega) \subset W^{2-d/2,\infty}(\Omega)$ for $d = 2, 3$, see, e.g., [32]. The estimated number of bits per vertex needed to reach discretization error accuracy can then be computed as

$$\sum_{j=1}^{l} c\left(2^{dj} - 2^{d(j-1)}\right)\left(\left(2 - \frac{d}{2}\right)(l - j + 1) + \frac{1}{2^{(2-d/2)(l-j+1)}\ln(2)}\right) + c\left(\left(2 - \frac{d}{2}\right)(l+1) + \frac{1}{2^{(2-d/2)(l+1)}\ln(2)}\right).$$

For 2D, this yields approximately 1.95 bits/vertex, and 1.55 bits/vertex for 3D. Compared to the estimate (4.12) for $y \in W^{2,\infty}(\Omega)$, each additional uniform mesh refinement requires only $2 - d/2$ bits per new node to keep the discretization error accuracy, due to worse interpolation properties. As before, the number of new vertices is $c2^{dj} - c2^{d(j-1)}$; thus, for a fixed discretization, less bits are needed than in the previous case of $y \in W^{2,\infty}(\Omega)$.

Impact of adaptive mesh refinement. Using adaptive mesh refinement instead of uniformly refined grids can lead to a drastic reduction of degrees of freedom, and thus to less data. As there are less nodes on higher levels in the mesh hierarchy, the expected compression factors will be smaller than in the uniform refinement case. It might appear that adaptivity renders compression ineffective. This, however, is not true, see Section 6.1 for a numerical example. As in the uniform refinement case, the prediction errors tend to be smaller on finer levels, such that the range to be quantized contains fewer intervals. Moreover, while adaptivity is used to control the error in the solution of the state equation, the quantization error affects the solution of the adjoint equation only. The different error propagation mechanisms might lead to different tolerances to be chosen.

4.3. Temporal Correlations

Up to now, only spatial correlations have been exploited for compression. For sufficiently smooth domain and data in the PDE, it holds $y \in C(\Omega \times [0,T])$. Thus the range of prediction errors on each level will usually not differ much between timesteps. If gradient-based methods, like steepest-descent or quasi-Newton methods, are used, the state solution is only accessed backwards in time, and no random access is required. The temporal correlation can be used to construct a second predictor, and use delta-encoding to further reduce the storage requirements. In the simplest case, the temporal predictor assumes the quantized spatial prediction error to be constant from one timestep to the next, i.e. the difference to be entropy coded for $k \in \mathcal{N}_j(t_n) \backslash \mathcal{N}_{j-1}(t_n)$ is calculated for $t_n < T$ as

$$d_k(t_n) = \begin{cases} i_k(t_n) - i_k(t_{n+1}), & k \in (\mathcal{N}_j(t_n) \backslash \mathcal{N}_{j-1}(t_n)) \cap (\mathcal{N}_j(t_{n+1}) \backslash \mathcal{N}_{j-1}(t_{n+1})) \\ i_k(t_n), & \text{otherwise} \end{cases}. \tag{4.13}$$

Care has to be taken, as grids may change between timesteps, if adaptive refinement is used. At final time,

$$d_k(T) = i_k(T) \quad \forall k \in \mathcal{N}_j(T) \backslash \mathcal{N}_{j-1}(T). \tag{4.14}$$

This ensures that decoding is possible backward in time. The number of different residuals to be encoded is reduced by this approach, increasing the performance of the range coder. For avoiding error accumulation due to quantization, the prediction at the timestep t_n is performed not for the reconstructed solution $\hat{y}(t_n)$, but for the quantized coefficients of the prediction error.

Note that when using backwards in time prediction, the quantized finite element solution of at least one previous timestep has to be kept in memory, as it is encoded

only after the following timestep is performed. For higher order predictors (linear, quadratic), more than two timesteps have to be kept in memory, which is easily implemented, but only feasible if the spatial discretization is not too fine.

Remark 4.3.1. Here, decoding is only possible backward in time, as only the final state is stored completely. For second-order optimization methods, like Newton-CG, the state solution needs to be reconstructed forward in time as well for the evaluation of Hessian-vector products. In this case temporal correlations can only be used at some larger computational cost: Instead of storing only the final timestep entirely, the complete information has to be stored at additional timesteps to be able to start delta-decoding from there (so called *intra*-frames in video compression [92]). Accessing the stored solution at an arbitrary timestep then requires to start decoding at the next fully stored step, and continue delta-decoding until the requested time is reached. While for that case delta-encoding still reduces the demand in storage space, the reduction of memory bandwidth is smaller, and the computational overhead increases.

4.4. Algorithms

For convenience, we present pseudo-code versions of the basic encoding and decoding methods.

Algorithm 1 Encoding

Input: coefficient vector $(y_k)_{k=0,\dots,|\mathcal{N}_l|-1}$, prolongation operators P_k, quantization error tolerance δ

for all $k \in \mathcal{N}_0$ **do**
  $i_k \leftarrow \text{floor}\left(\frac{y_k+\delta}{2\delta}\right)$
  $\hat{y}_k \leftarrow 2\delta i_k$
end for
for grid level $j = 1$ to maximum grid level l **do**
  for all $k \in \mathcal{N}_j \backslash \mathcal{N}_{j-1}$ **do**
    $\xi_k \leftarrow y_k - P_k(\hat{y}_{m_1}, \hat{y}_{m_2})$
    $i_k \leftarrow \text{floor}\left(\frac{\xi_k+\delta}{2\delta}\right)$
    $\hat{y}_k \leftarrow 2\delta i_k + P_k(\hat{y}_{m_1}, \hat{y}_{m_2})$
  end for
end for
write $i_k, k = 0, \dots, |\mathcal{N}_l| - 1$ to disk using entropy coding

Algorithm 1 saves a coefficient vector of the state equation solution at a single

timestep to disk for a given mesh. No delta-encoding in time is used. The quantization error tolerance δ is equal for all nodes. The prolongation operators P_k transfer a nodal basis vector from level l to $l+1$ and are typically available in finite element codes at no additional expense.

Algorithm 2 Reconstruction

Input: prolongation operators P_k, quantization error tolerance δ

read $i_k, k = 0, \ldots, |\mathcal{N}_l| - 1$ from disk using entropy coding
for all $k \in \mathcal{N}_0$ **do**
  $\hat{y}_k \leftarrow 2\delta i_k$
end for
for grid level $j = 1$ to maximum grid level l **do**
  for all $k \in \mathcal{N}_j \backslash \mathcal{N}_{j-1}$ **do**
    $\hat{y}_k \leftarrow 2\delta i_k + P_k(\hat{y}_{m_1}, \hat{y}_{m_2})$
  end for
end for

Additional storage of the grids is required for de-compression.

In Algorithm 2, the finite element solution is reconstructed on a given mesh. The prolongation operators P_k for the corresponding grids are required as input. When solving the adjoint equation using adaptively refined meshes, these operators need to be re-computed for the current state grid. For linear interpolation, this requires access to father-child relations of elements in the hierarchical mesh, and leads to an increase in computation time. For reasonable data structures and implementations this is only a minor overhead.

5. Adaptive Error Control

Due to the inexact reconstruction, and thus inexact data for the adjoint equation, the error induced in the reduced gradient, and reduced Hessian, has to be controlled, to not impede convergence of the optimization. In this chapter, we analyze accuracy requirements of different optimization methods. We derive error estimates and computable approximations for the influence of lossy trajectory storage on the reduced gradient and reduced Hessian, and propose techniques for the adaptive choice of quantization tolerances.

After deriving worst-case error bounds in Section 5.1, in Section 5.2 we are concerned with a simple steepest-descent method, where fulfilling the so-called angle condition is sufficient to maintain convergence. As the convergence speed of steepest-descent is too slow for the method to be of practical use, we consider a BFGS-quasi-Newton-method in Section 5.3. There, the computed reduced gradients are used to update an approximation of the reduced Hessian, such that the inexactness in the gradient influences not only a single iteration, but also the subsequent ones. Finally, we analyze the behavior of the Newton-CG method under the influence of lossy compression. Here, not only the state, but also adjoint and linearized-state solutions have to be stored, which poses additional complications. Especially compression of the linearized-state trajectory leads to errors in the reduced Hessian-vector products, which vary every CG iteration. We deal with these difficulties in Section 5.4.

In this thesis we restrict the discussion to the three mentioned exemplary optimization methods. Analysis of inexact problem information or inexact step computation for some other algorithms can be found in the literature, for example for Trust-Region SQP methods (e.g. by Heinkenschloss and Vicente [50]) or Interior Point methods (e.g. by Schiela and Günther [108]). The influence of lossy trajectory storage can be analyzed for these algorithms in a similar fashion, using the tools provided in this chapter.

Remark. Adaptivity here refers to the choice of quantization accuracy during the progress of the optimization. In each optimization iteration, the trajectories are quantized uniformly. We do not consider "adaptive quantization" in the sense of choosing quantization tolerances varying in time and space.

The findings of this chapter are mainly published in [39].

5.1. Worst-Case Error Bounds

In this section we analyze the influence of quantization errors on the reduced gradient and Hessian-vector products. We derive error equations and propose worst-case estimates. The following notations are used to distinguish between different errors as well as exact and inexact quantities:

- $\varepsilon_\cdot$ denotes the quantization error, e.g. ε_y is the quantization error of the state variable y.
- $\hat{\cdot}$ denotes a inexactness due to compression, e.g. $\hat{y} = y + \varepsilon_y$.
- $\tilde{\cdot}$ denotes an inexact quantity, where the inexactness is due to compression of an *input* quantity. E.g., $\tilde{\lambda}$ denotes the adjoint equation using inexact state values $\hat{y}$ as an input (as opposed to λ using the exact state solution y).
- $e_\cdot$ denotes the error in quantities computed with inexact input, e.g. $\tilde{\lambda} = \lambda + e_\lambda$.

5.1.1. Reduced Gradient

As introduced in Section 2.2, the reduced gradient can be computed via the implicit function theorem, yielding

$$j'(u) = J_u(y,u) + c_u(y,u)^\star \lambda, \tag{5.1}$$

where λ solves the adjoint equation

$$c_y(y,u)^\star \lambda = -J_y(y,u). \tag{5.2}$$

Due to compression, only an inexact reduced gradient $\tilde{j'}$ can be computed, with y replaced by its reconstruction $\hat{y}$ in (5.1) and (5.2).

Theorem 5.1.1. *The error in the reduced gradient $e_{j'} = \tilde{j'} - j'$ is given by*

$$e_{j'} = c_u(\hat{y},u)^\star e_\lambda, \tag{5.3}$$

where the error in the adjoint equation $e_\lambda = \tilde{\lambda} - \lambda$ fulfills

$$\left(c_y(\hat{y},u)^\star - (c_{yy}(\hat{y},u)\varepsilon_y)^\star\right) e_\lambda = -J_{yy}(\hat{y},u)\varepsilon_y - (c_{yy}(\hat{y},u)\varepsilon_y)^\star \tilde{\lambda}, \tag{5.4}$$

up to $\mathcal{O}(\|\varepsilon_y\|^2)$.

Proof. Subtracting the adjoint equations for exact and inexact input gives

$$c_y(\hat{y},u)^\star\tilde{\lambda} - c_y(y,u)^\star\lambda = -J_y(\hat{y},u) + J_y(y,u). \tag{5.5}$$

Using Taylor expansion, we have that

$$\begin{aligned} J_y(y,u) &= J_y(\hat{y},u) - J_{yy}(\hat{y},u)\varepsilon_y + \mathcal{O}(\|\varepsilon_y\|^2), \text{ and} \\ c_y(y,u) &= c_y(\hat{y},u) - c_{yy}(\hat{y},u)\varepsilon_y + \mathcal{O}(\|\varepsilon_y\|^2). \end{aligned}$$

Thus (5.5) becomes

$$c_y(\hat{y},u)^\star e_\lambda + (c_{yy}(\hat{y},u)\varepsilon_y)^\star\,(\tilde{\lambda} - e_\lambda) = -J_{yy}(\hat{y},u)\varepsilon_y,$$

which shows (5.4). As by assumption (2.2) $J_u(\hat{y},u) = J_u(y,u)$, as well as $c_u(\hat{y},u) = c_u(y,u)$ for (RDS), the claim follows. □

5.1.2. Reduced Hessian-Vector Products

To evaluate the action of the reduced Hessian $j''(u)$, given by equation (2.10), on some vector $\delta u \in U$, the following computations are needed:

1. Solve the linearized-state equation $c_y(y,u)v = c_u(y,u)\delta u$ for $v \in Y$.
2. Set $z := J_{yy}(y,u)v + \langle c_{yy}(y,u)(v,\cdot),\lambda\rangle_{Z^\star,Z}$.
3. Solve the adjoint-for-Hessian equation $c_y(y,u)^\star w = z$ for $w \in Z$.
4. Set $j''(u)\delta u := J_{uu}(y,u)\delta u + c_u(y,u)^\star w + c_{uu}(y,u)^\star\lambda\delta u$.

In terms of storage, either v or z have to be kept. Both variants have the same implementation complexity, and similar error analysis. Here we choose to store v during Step 1, and generate z on-the-fly in Step 3 from the stored quantities.

In the following we analyze in detail the errors introduced by lossy trajectory compression.

Step 1. Due to compression of y, the exact equation is not available. Instead, $c_y(\hat{y},u)\tilde{v} = c_u(\hat{y},u)\delta u$ is solved for $\tilde{v}$.

Lemma 5.1.2. *The error $e_v = \tilde{v} - v$ fulfills*

$$\big(c_y(\hat{y},u) - c_{yy}(\hat{y},u)\varepsilon_y\big)e_v = -c_{yy}(\hat{y},u)\varepsilon_y\tilde{v} \tag{5.6}$$

up to $\mathcal{O}(\|\varepsilon_y\|^2)$.

Proof. Subtracting exact and inexact equation, and using Taylor expansion as in the proof of Theorem 5.1.1 we get

$$c_y(\hat{y}, u)e_v + c_{yy}(\hat{y}, u)\varepsilon_y v = 0,$$

as for (RDS) c_u is independent of y. Replacing $v = \tilde{v} - e_v$ the claim follows. □

Step 2. Instead of z, only $\tilde{z} = J_{yy}(\hat{y}, u)\hat{\tilde{v}} + \langle c_{yy}(\hat{y}, u)\hat{\tilde{v}}, \hat{\tilde{\lambda}}\rangle$ can be formed. Sources of the inexactness here are not only the compression of y, but also the inexactly computed and compressed trajectories λ and $\tilde{v}$.

Lemma 5.1.3. *The error $e_z = \tilde{z} - z$ is given by*

$$\begin{aligned} e_z = {} & \big(J_{yy}(\hat{y}, u) - J_{yyy}(\hat{y}, u)\varepsilon_y\big)(e_v + \varepsilon_v) + J_{yyy}(\hat{y}, u)\varepsilon_y\hat{\tilde{v}} \\ & + \langle\big(c_{yy}(\hat{y}, u) - c_{yyy}(\hat{y}, u)\varepsilon_y\big)(e_v + \varepsilon_v), \hat{\tilde{\lambda}}\rangle + \langle c_{yyy}(\hat{y}, u)\varepsilon_y\hat{\tilde{v}}, \hat{\tilde{\lambda}}\rangle \\ & + \langle\big(c_{yy}(\hat{y}, u) - c_{yyy}(\hat{y}, u)\varepsilon_y\big)\hat{\tilde{v}}, e_\lambda + \varepsilon_\lambda\rangle \\ & - \langle\big(c_{yy}(\hat{y}, u) - c_{yyy}(\hat{y}, u)\varepsilon_y\big)(e_v + \varepsilon_v), e_\lambda + \varepsilon_\lambda\rangle \end{aligned} \tag{5.7}$$

up to $\mathcal{O}(\|\varepsilon_y\|^2)$.

Proof. Computing $\tilde{z} - z$ gives

$$e_z = J_{yy}(\hat{y}, u)\hat{\tilde{v}} - J_{yy}(y, u)v + \langle c_{yy}(\hat{y}, u)\hat{\tilde{v}}, \hat{\tilde{\lambda}}\rangle - \underbrace{\langle c_{yy}(y, u)v, \lambda\rangle}_{(\star)},$$

Using Taylor expansion, we have

$$\begin{aligned} J_{yy}(y, u)v &= J_{yy}(\hat{y}, u)v - J_{yyy}(\hat{y}, u)\varepsilon_y v + \mathcal{O}(\|\varepsilon_y\|^2) \\ c_{yy}(y, u)v &= c_{yy}(\hat{y}, u)v - c_{yyy}(\hat{y}, u)\varepsilon_y v + \mathcal{O}(\|\varepsilon_y\|^2). \end{aligned}$$

Thus $(\star)$ becomes

$$\langle c_{yy}(\hat{y}, u)v - c_{yyy}(\hat{y}, u)\varepsilon_y v, \lambda\rangle.$$

By inserting $\lambda = \hat{\tilde{\lambda}} - e_\lambda - \varepsilon_\lambda$ and $v = \hat{\tilde{v}} - e_v - \varepsilon_v$ as well as recombining the duality products, the claim is shown. □

Step 3. As y and z are available only inexactly, we can only solve $c_y(\hat{y},u)^\star \tilde{w} = \tilde{z}$ for $\tilde{w}$.

Lemma 5.1.4. *The error $e_w = \tilde{w} - w$ fulfills*

$$\left(c_y(\hat{y},u)^\star - \left(c_{yy}(\hat{y},u)\varepsilon_y\right)^\star\right) e_w = e_z - \left(c_{yy}(\hat{y},u)\varepsilon_y\right)^\star \tilde{w} \tag{5.8}$$

up to $\mathcal{O}(\|\varepsilon_y\|^2)$.

Proof. Subtracting exact and inexact equation, and using Taylor as before, we get

$$c_y(\hat{y},u)^\star e_w + \left(c_{yy}(\hat{y},u)\varepsilon_y\right)^\star w = e_z.$$

Substituting $w = \tilde{w} - e_w$ gives the desired result. □

Step 4. Finally, only $\tilde{w}$ is available instead of w.

Lemma 5.1.5. *The error $e_{mv} = \tilde{j''}(u)\delta u - j''(u)\delta u$ in the Hessian-vector product is given by*

$$e_{mv} = c_u(\hat{y},u)^\star e_w + c_{uu}(\hat{y},u)^\star(e_\lambda + \varepsilon_\lambda)\delta u. \tag{5.9}$$

Proof. For (RDS) and by assumption (2.2), J_{uu}, c_u are independent of y. Subtracting exact and inexact equation shows the lemma. □

5.1.3. Computable Error Estimates

While the error can, in principle, be estimated up to $\mathcal{O}(\|\varepsilon_y\|^2)$ by solving the equations derived in the previous sections, this can not directly be used algorithmically for two reasons. First, the equations should only be solved on coarse, fixed grids to keep both the computational overhead and the storage demand small. Second, for adaptively choosing the quantization tolerances to store state, adjoint, and linearized-state, the error equations have to be solved before the actual computation, thus computationally unavailable quantities have to be replaced by estimates.

A worst case estimate for the error e_z in the right-hand-side of the adjoint-for-Hessian error equation (5.8) can easily be derived by taking absolute values, or a suitable norm, and applying the Cauchy-Schwarz inequality. Here, and below, taking the absolute value for the worst case is motivated by the parabolic nature of

the involved PDEs, which damp out oscillatory errors. Splitting the error into the different contributions we arrive at

$$\begin{aligned}\|e_z\| \leq{}& \|J_{yy}(\hat{y},u) - J_{yyy}(\hat{y},u)\varepsilon_y\| \, \|e_v + \varepsilon_v\| + \|J_{yyy}(\hat{y},u)\varepsilon_y\| \, \big\|\hat{\tilde{v}}\big\| \\ &+ \|c_{yy}(\hat{y},u) - c_{yyy}(\hat{y},u)\varepsilon_y\| \, \|e_v + \varepsilon_v\| \, \big\|\hat{\tilde{\lambda}}\big\| + \|c_{yyy}(\hat{y},u)\varepsilon_y\| \, \big\|\hat{\tilde{v}}\big\| \big\|\hat{\tilde{\lambda}}\big\| \\ &+ \|c_{yy}(\hat{y},u) - c_{yyy}(\hat{y},u)\varepsilon_y\| \, \big\|\hat{\tilde{v}}\big\| \, \|e_\lambda + \varepsilon_\lambda\| \\ &+ \|c_{yy}(\hat{y},u) - c_{yyy}(\hat{y},u)\varepsilon_y\| \, \|e_v + \varepsilon_v\| \, \|e_\lambda + \varepsilon_\lambda\| \, .\end{aligned} \tag{5.10}$$

We now turn to evaluating the errors e_v, e_λ, and e_w. The errors for adjoint and adjoint-for-Hessian, given by (5.4) and (5.8), are governed by similar equations with different right-hand-sides. We make use of Theorem A.2 to get an upper bound for the error e in the adjoint and adjoint-for-Hessian equations. The right-hand-side of the error equation is given by

$$r(x,t,e_\iota) = \big(c_{yy}(\hat{y},u)\varepsilon_y\big)^\star e_\iota + \varphi - \big(c_{yy}(\hat{y},u)\varepsilon_y\big)^\star \psi, \tag{5.11}$$

where

$$\varphi = \begin{cases} -J_{yy}(\hat{y},u)\varepsilon_y, & \iota = \lambda \text{ (adjoint)} \\ e_z, & \iota = w \text{ (adjoint-for-Hessian)}, \end{cases}$$

and

$$\psi = \begin{cases} \tilde{\lambda}, & \iota = \lambda \text{ (adjoint)} \\ \tilde{w}, & \iota = w \text{ (adjoint-for-Hessian)}. \end{cases}$$

For the error in the linearized-state e_v given by equation (5.6), we note that this equation can be transformed to a similar structure, using the standard time substitution $t = T - \tau$ (to transform the equation to a backward-in-time equation like the adjoint equations), and setting $\varphi = 0, \psi = \tilde{v}$.

Exemplarily, consider a generic tracking-type objective functional

$$\begin{aligned}J(y,u) = \frac{1}{2}\|y - y_Q\|^2_{L^2(\Omega\times(0,T))} + \frac{1}{2}\|y - y_\Sigma\|^2_{L^2(\partial\Omega\times(0,T))} \\ + \frac{1}{2}\|y(T) - y_\Omega\|^2_{L^2(\Omega)} + \frac{\alpha}{2}\|u\|^2_U \, ,\end{aligned}$$

which is to be minimized subject to the reaction-diffusion system (RDS). Then, the error equation for the error in the adjoint with right-hand-side (5.11) corresponds to the strong formulation

$$\begin{aligned} -[e_\lambda]_t - D\nabla\cdot(\sigma\nabla e_\lambda) - f_y(\hat{y})e_\lambda + f_{yy}(\hat{y})\varepsilon_y e_\lambda &= f_{yy}(\hat{y})\varepsilon_y\tilde{\lambda} - \varepsilon_y && \text{in } \Omega\times(0,T) \\ B\partial_\nu e_\lambda + Ce_\lambda &= -[\varepsilon_y]_{|\partial\Omega\times(0,T)} && \text{on } \partial\Omega\times(0,T) \\ e_\lambda(\cdot,T) &= -\varepsilon_y(T) && \text{in } \Omega. \end{aligned} \tag{5.12}$$

For the error in the adjoint-for-Hessian, a similar strong formulation can be obtained, see Section 6.3 for a scalar example. Thus, for (RDS) only the reaction function $f(y)$ contributes to the operator $c_{yy}(\hat{y}, u)$.

Consider the Nemytskii-operator $F : y(\cdot,\cdot) \mapsto f(y(\cdot,\cdot))$ generated by the nonlinearity f. If f is sufficiently regular, the Nemytskii-operator F is twice continuously Fréchet differentiable in $L^\infty(\Omega \times (0,T))$ and the second derivative can be evaluated by

$$\left[F''(y)\, y_1\, y_2\right](x,t) = f_{yy}(y(x,t))y_1(x,t)y_2(x,t),$$

i.e. the derivative $F''(y)$ can be identified with the real-valued function $f_{yy}(y(x,t))$. Moreover we have

$$\left\|F''(y)\right\|_{\mathcal{L}(L^\infty(\Omega\times(0,T)),\mathcal{L}(L^\infty(\Omega\times(0,T))))} = \left\|f_{yy}(y(\cdot,\cdot))\right\|_{L^\infty(\Omega\times(0,T))},$$

where $\mathcal{L}(X_1, X_2)$ denotes the normed space of all linear and continuous mappings from X_1 into X_2. See, e.g., [127, Sections 4.3, 4.9] for details and a thorough discussion.

For derivation of computable error estimates, we thus re-interpret Nemytskii-operators like $c_{yy}(\hat{y}, u)$ as coefficient functions and denote by $|c_{yy}(\hat{y}, u)^\star|$ etc. the pointwise absolute value in $\Omega \times (0,T)$.

To continue the discussion, we have to distinguish between scalar equations and systems of reaction-diffusion equations.

Scalar equations. In the scalar case, the strong formulation (5.12) motivates the following error bound, making use of a comparison principle for classical solutions to the error equations.

Theorem 5.1.6. *Let $m = 1$, and $\overline{e_\iota}$ be the solution of*

$$c_y(\hat{y}, u)^\star \overline{e_\iota} = \overline{r}(x,t,\overline{e_\iota}), \quad \iota \in \{\lambda, w\} \tag{5.13}$$

with

$$\overline{r}(x,t,e) = |c_{yy}(\hat{y}, u)^\star|\varepsilon_y^{max} e + |\varphi| + |c_{yy}(\hat{y}, u)^\star \psi|\varepsilon_y^{max}, \tag{5.14}$$

and an upper bound on the state quantization error $\varepsilon_y^{max} \geq \varepsilon_y(x,t)\ \forall (x,t) \in \Omega \times (0,T)$. Then $e_\iota \leq \overline{e_\iota}$.

Proof. The error estimate $\overline{e_\iota}$ is the solution of a backward linear parabolic equation, where the source terms, boundary- and terminal values are non-negative. By

the parabolic maximum principle we get $\overline{e_\iota} \geq 0$. Thus, for all $\overline{e_\iota}$ satisfying equation (5.13), we have

$$\overline{r}(x,t,\overline{e_\iota}(x,t)) \geq r(x,t,\overline{e_\iota}(x,t)),$$

and $\overline{e_\iota}$ is a super-solution to equation (5.4) or (5.8), respectively. With the standard time transformation $\tau = T - t$ the backward-in-time equations (5.4) or (5.8) and (5.13) are transformed to forward equations. Then by Theorem A.2 in combination with Remark A.5 the claim follows. □

Remark 5.1.7. The estimates $\overline{e_\iota}$ are still not computable error bounds, as they depend on $\hat{y}$, and $\tilde{\lambda}$ or $\tilde{w}$. Possible remedies are the use of upper bounds of these quantities specific to the actual equations being used, or heuristic choices like using quantities from previous optimization iterations. In Section 5.4 we give more details on the actual realization and sketch an algorithm (Algorithm 5, p. 65). See also Sections 6.2, 6.3 for numerical examples.

Reaction-diffusion systems. Again motivated by the strong formulation (5.12), in the case of reaction-diffusion systems we can construct a super-reaction function by following [10], and apply the comparison theorem Theorem A.2.

Theorem 5.1.8. *Let $\underline{e}$ be a sub-solution to the adjoint error equation* (5.4) *or* (5.8) *for $\iota = \lambda, w$, respectively. Define $\overline{r}$ by*

$$\overline{r}_i(x,t,e) = \sup_{\{\eta \mid \underline{e} \leq \eta \leq e,\ \eta_i = e_i\}} r_i(x,t,\eta),\ i = 1,\dots,m. \tag{5.15}$$

As in the scalar case, let $\overline{e_\iota}, \iota \in \{\lambda, w\}$ be the solution of

$$c_y(\hat{y},u)^\star \overline{e_\iota} = \overline{r}(x,t,\overline{e_\iota}), \tag{5.16}$$

Then $e_\lambda \leq \overline{e_\iota}$.

Proof. The function $\overline{r}$ constructed by (5.15) is uniformly Lipschitz continuous in e. It satisfies $\overline{r}(x,t,e) \geq r(x,t,e)\ \forall e \in \mathbb{R}^m$ and is quasi-monotone non-decreasing, see [10]. Thus $e_\iota \leq \overline{e_\iota}$ by Theorem A.2. □

Remark 5.1.9. The construction of super-reaction functions by (5.15) needs the derivation of a sub-solution to the original error equation, and thus is problem-dependent. For the monodomain equations describing the electrical activity of the heart this has been carried out in detail in [37] as well as in Section 6.4.

5.2. Steepest-Descent

Descent methods are a common class of algorithms for computing solutions of the optimization problem. In combination with additional requirements on the step size, convergence can be shown if the descent directions δj satisfy the so-called angle-condition

$$\left\langle j'(u), \delta j \right\rangle_{U^\star, U} \leq -\alpha \left\| j'(u) \right\|_{U^\star} \left\| \delta j \right\|_U \tag{5.17}$$

for some fixed $\alpha > 0$ [55].

In this section we need to distinguish notationally between the reduced derivative $j'(u) \in U^\star$ and the reduced gradient $\nabla j(u) \in U$, defined with aid of the Riesz isomorphism, as both of them are required in the following (cf. Remark 2.2.6).

Using the negative gradient $\delta j = -\nabla j(u)$ of the objective functional as a descent direction yields the so-called steepest-descent method. Due to lossy compression the reduced gradient can not be computed exactly, i.e. only $\delta j = -\nabla j(u) + e$ can be chosen. For convergence of the inexact steepest-descent method, we have the following theorem.

Theorem 5.2.1. *Let $\varepsilon < \frac{1}{2}$ and compute $\delta j = -\nabla j(u) + e$ such that $\|e\| \leq \varepsilon \|\delta j\|$. Then δj satisfies the angle-condition* (5.17).

Proof. As $\|\delta j\| \leq \|-\nabla j(u)\| + \|e\| \leq \|-\nabla j(u)\| + \varepsilon \|\delta j\|$, we get

$$\begin{aligned} \left\langle j'(u), \delta j \right\rangle &= -\left\|\nabla j(u)\right\|^2 + \left\langle j'(u), e \right\rangle \\ &\leq -(1-\varepsilon) \left\|\nabla j(u)\right\| \left\|\delta j\right\| + \left\|\nabla j(u)\right\| \left\|e\right\| \\ &\leq -(1-2\varepsilon) \left\|\nabla j(u)\right\| \left\|\delta j\right\|, \end{aligned}$$

and thus (5.17) with $\alpha = 1 - 2\varepsilon > 0$. □

Example 5.2.2. We consider again boundary control of the linear heat equation (see also Example 2.1.1):

$$\min \frac{1}{2} \left\| y - y_d \right\|^2_{L^2(\Omega \times (0,T))} + \frac{\alpha}{2} \left\| u \right\|^2_{L^2(\partial\Omega \times (0,T))}$$

subject to

$$y_t - \Delta y = f \text{ on } \Omega \times (0,T), \quad \partial_\nu y + y - u = 0 \text{ on } \partial\Omega \times (0,T), \quad y(\cdot, 0) = 0 \text{ in } \Omega.$$

The reduced gradient is given as

$$\nabla j(u) = \alpha u + \lambda_{|\partial\Omega}$$

with the adjoint variable λ solving

$$-\lambda_t - \Delta\lambda = y - y_d \text{ on } \Omega \times (0,T), \quad \partial_\nu \lambda + \lambda = 0 \text{ on } \partial\Omega \times (0,T), \quad \lambda(\cdot,T) = 0 \text{ in } \Omega.$$

As derived above in Section 5.1, for a perturbation of the state by a quantization error, $\hat{y} = y + \varepsilon_y$, the error in the reduced gradient amounts to $e_\lambda = \tilde{\lambda} - \lambda$, with $\tilde{\lambda}$ the adjoint solution to the right hand side $\hat{y} - y_d$. Due to the linearity, e_λ is the solution of

$$-[e_\lambda]_t - \Delta e_\lambda = \varepsilon_y \text{ on } \Omega \times (0,T), \quad \partial_\nu e_\lambda + e_\lambda = 0 \text{ on } \partial\Omega \times (0,T), \quad e_\lambda(\cdot,T) = 0 \text{ in } \Omega.$$

Here the error norm is bounded by the norm of the quantization error in the state values, e.g. for $\varepsilon_y \in L^2(\Omega \times (0,T))$, $\|e_\lambda\|_{L^2(0,T,H^1(\Omega))} \leq c \|\varepsilon_y\|_{L^2(\Omega\times(0,T))}$ [127].

Using Theorems 5.1.1 and 5.2.1 combined with Theorem 5.1.6 or 5.1.8, an adaptive strategy for choosing an upper bound δ for the quantization error in dependence on the progress of the optimization can be derived (Algorithm 3), see also Section 6.2 for an example.

Algorithm 3 Adaptive quantization for steepest descent

1. Compute the solution e_λ to equation (5.13) or (5.16) for scalar equations or systems, respectively, using some $\varepsilon_y^{\max}$, and keep $\|c_u(\hat{y},u)^\star e_\lambda\|_U$.

2. In iteration i of the inexact steepest-descent method, set

$$\delta_{i+1} < \frac{1}{2} \frac{\theta_{i+1}}{\|c_u(\hat{y},u)^\star e_\lambda\|_U} \varepsilon_y^{\max}, \tag{5.18}$$

   where

$$\theta_{i+1} = \frac{\left\|\widetilde{\nabla j}(u_i)\right\|_U^2}{\left\|\widetilde{\nabla j}(u_{i-1})\right\|_U} \tag{5.19}$$

   is an estimate for the gradient norm of the next step derived from the linear convergence of the gradient descent method.

If, for linear problems, the equations in step 1 of Algorithm 3 are solved exactly, the error estimate is reliable. For practical realization, the equations are only solved approximately, typically on a rather coarse mesh to keep the computational overhead reasonably small. This inaccuracy turns out to be of little consequence, as on one hand the error bounds tend to be rather smooth and well represented on coarse meshes, and on the the other hand, the computed error bounds are not particularly sharp.

Example 5.2.3. In the setting of Example 5.2.2 we have the following result: Let δ be an upper bound for the quantization error in the state values, i.e. $|\varepsilon_y| \leq \delta$, and ξ be the solution of

$$-\xi_t - \Delta\xi = \delta \text{ on } \Omega \times (0,T), \quad \partial_\nu \xi + \xi = 0 \text{ on } \partial\Omega \times (0,T), \quad \xi(\cdot,T) = 0 \text{ in } \Omega.$$

Then

$$\left\| e_{j'} \right\|_{L^2(\partial\Omega\times(0,T))} \leq \|\xi\|_{L^2(\partial\Omega\times(0,T))},$$

where

$$e_{j'} = \alpha u + \tilde{\lambda}_{|\partial\Omega} - (\alpha u + \lambda_{|\partial\Omega}) = e_{\lambda|\partial\Omega}.$$

This can be shown as follows: By the parabolic maximum principle [100], the maximum M of $\eta := e_\lambda - \xi$ is attained either at $t = 0$ or on $\partial\Omega \times (0,T)$. Assume $M > 0$. Then there is $(\bar{x},\bar{t}) \in \partial\Omega \times (0,T)$ with $\eta(\bar{x},\bar{t}) = M$. But then, again due to the strong maximum principle, $\partial_\nu \eta(\bar{x},\bar{t}) > 0$, contradicting the homogeneous Robin boundary condition. Hence, $M \leq 0$ and $e_\lambda \leq \xi$.

5.3. BFGS-Quasi-Newton

Quasi-Newton methods aim to increase convergence speed by constructing approximations to the Hessian from gradient information. We restrict ourselves to the well-known BFGS-method (named after its inventors Broyden, Fletcher, Goldfarb, Shanno). It is one of the most efficient quasi-Newton methods due to fast theoretical convergence and numerical experience, see, e.g., [21, 8]. The algorithm computes an approximation to the reduced Hessian using rank-two modifications of an initial approximation, see equation (5.23) below. The control iterates are created by the formula

$$u_{i+1} = u_i - \alpha_i B_i^{-1} j'(u_i) = u_i + \alpha_i p_i, \tag{5.20}$$

where the step-size α_i fulfills the Wolfe-Powell conditions [98]

$$j(u_{i+1}) - j(u_i) \leq c_1 \alpha_i \langle j'(u_i), p_i \rangle_{U^\star,U} \tag{5.21}$$

$$\langle j'(u_{i+1}), p_i \rangle_{U^\star,U} \geq c_2 \langle j'(u_i), p_i \rangle_{U^\star,U}, \tag{5.22}$$

$0 < c_1 < \frac{1}{2}$, $c_1 < c_2 < 1$. Due to the Hilbert space setting, we identify U with its dual $U^\star$, and use the scalar product $(\cdot,\cdot)_U$ instead of the duality product $\langle \cdot,\cdot \rangle_{U^\star,U}$.

For simplicity, in the following we assume a given, fixed discretization, and comment on quasi-Newton methods in function space later. To distinguish the notation, we abbreviate the approximated Hessian in iteration i of the quasi-Newton algorithm

by B_i and the gradient by g_i. Defining $\gamma_i = g_{i+1} - g_i$ and $s_i = u_{i+1} - u_i$, the update for the Hessian approximation is given by

$$B_{i+1} = B_i - \frac{(B_i s_i, \cdot) B_i s_i}{(s_i, B_i s_i)} + \frac{(\gamma_i, \cdot)\gamma_i}{(s_i, \gamma_i)}. \tag{5.23}$$

If B_i is symmetric positive definite, B_{i+1} has these properties as well.

With inexact gradient computation, $\tilde{g}_i = g_i + e_{g_i}$, instead of the search direction $p_i = -B_i^{-1} g_i$ we only have an inexact direction $\tilde{p}_i = -B_i^{-1}\tilde{g}_i$. Further, algorithmically we can only satisfy an inexact variant of the Wolfe-Powell conditions:

$$j(u_{i+1}) - j(u_i) \leq \tilde{c}_1 \tilde{\alpha}_i (\tilde{g}_i, \tilde{p}_i) \tag{5.24}$$

$$(\tilde{g}_{i+1}, \tilde{p}_i) \geq \tilde{c}_2 (\tilde{g}_i, \tilde{p}_i), \tag{5.25}$$

$0 < \tilde{c}_1 < \frac{1}{2}$, $\tilde{c}_1 < \tilde{c}_2 < 1$. Thus, only the inexact quantities $\tilde{s}_i = \tilde{\alpha}_i \tilde{p}_i$ and $\tilde{\gamma}_i = \tilde{g}_{i+1} - \tilde{g}_i$ are available.

Throughout the section, we assume that the reduced Hessian $j''(u)$ is positive definite and Lipschitz-continuous.

Remark. A convergence analysis for BFGS with inexact gradients can also be found in [33] by Felgenhauer. While the proof idea for superlinear convergence is similar—making use of the Dennis-Moré condition (5.34)—the proof given in Section 5.3.1 is somewhat simpler. There, we show that under certain conditions the inexact search direction $\tilde{p}_i$ is a descent direction satisfying (5.17), thus leading to convergence of the method. Moreover, we give accuracy requirements in Lemma 5.3.1 and Lemma 5.3.9 which are computationally available and thus can be used for implementation of the algorithms.

5.3.1. Convergence of Inexact BFGS

To prove convergence of the inexact BFGS-quasi-Newton method, we need to show that $\tilde{p}_i$ is a descent direction, and that symmetry and positive definiteness of B_i is preserved by the update (5.23) with inexact quantities.

Lemma 5.3.1. *Assume that B is symmetric positive definite with bounded condition number $\kappa(B)$, and $p = B^{-1}\tilde{g}$. Choose $\varepsilon < \frac{1}{2}$ and let the gradient error e_g fulfill*

$$\|e_g\| \leq \frac{\varepsilon}{\kappa(B)^{1/2}} \|\tilde{g}\|. \tag{5.26}$$

Then

$$(g, B^{-1}\tilde{g}) \geq \frac{1 - 2\varepsilon}{\kappa(B)^{1/2}} \|g\| \left\|B^{-1}\tilde{g}\right\|. \tag{5.27}$$

Proof. From the proof of Theorem 5.2.1 we have the implication

$$\|e_g\| \leq \varepsilon \|\tilde{g}\| \quad \Rightarrow \quad \frac{(g, \tilde{g})}{\|g\| \|\tilde{g}\|} \geq (1 - 2\varepsilon). \tag{5.28}$$

With this starting point we follow the steps of the proof of Prop. 2.2 in [41]. To treat the BFGS search direction $-B^{-1}\tilde{g}$, we replace e_g, g and $\tilde{g}$ in (5.28) by $B^{-1/2}e_g, B^{-1/2}g$ and $B^{-1/2}\tilde{g}$, respectively. A short calculation similar to Theorem 5.2.1 shows that the transformed implication

$$\left\|B^{-1/2}e_g\right\| \leq \varepsilon \left\|B^{-1/2}\tilde{g}\right\| \quad \Rightarrow \quad \frac{(B^{-1/2}g, B^{-1/2}\tilde{g})}{\left\|B^{-1/2}g\right\| \left\|B^{-1/2}\tilde{g}\right\|} \geq (1 - 2\varepsilon) \tag{5.29}$$

holds. Multiplying by $\left\|B^{-1/2}\right\|^{-1}$ and using the inequalities

$$\left\|B^{-1/2}e_g\right\| \leq \left\|B^{-1/2}\right\| \|e_g\|$$

and

$$\left\|B^{-1/2}\right\| \left\|B^{-1/2}\tilde{g}\right\| \geq \left\|B^{-1}\tilde{g}\right\|$$

yields

$$\|e_g\| \leq \varepsilon \frac{\left\|B^{-1/2}\tilde{g}\right\|}{\left\|B^{-1/2}\right\|} \quad \Rightarrow \quad \frac{(g, B^{-1}\tilde{g})}{\left\|B^{-1/2}g\right\| \left\|B^{-1}\tilde{g}\right\|} \geq \frac{(1 - 2\varepsilon)}{\left\|B^{-1/2}\right\|}. \tag{5.30}$$

Denoting the smallest singular value of $B^{-1/2}$ by $\sigma_{\min}(B^{-1/2})$, with $\left\|B^{-1/2}g\right\| \leq \sigma_{\min}(B^{-1/2}) \|g\|$ the right-hand side of the implication (5.30) becomes

$$\frac{(g, B^{-1}\tilde{g})}{\sigma_{\min}(B^{-1/2}) \|g\| \left\|B^{-1}\tilde{g}\right\|} \geq \frac{(1 - 2\varepsilon)}{\left\|B^{-1/2}\right\|}. \tag{5.31}$$

Extending $\varepsilon \frac{\left\|B^{-1/2}\tilde{g}\right\|}{\left\|B^{-1/2}\right\|}$ on the left-hand side by $\left\|B^{-1/2}\right\| / \left\|B^{-1/2}\right\|$ and using the condition number $\kappa(B^{1/2}) = \left\|B^{1/2}\right\| \left\|B^{-1/2}\right\|$ we have

$$\varepsilon \frac{\left\|B^{-1/2}\tilde{g}\right\|}{\left\|B^{-1/2}\right\|} \geq \varepsilon \frac{\left\|B^{1/2}B^{-1/2}\tilde{g}\right\|}{\kappa(B^{1/2})} = \frac{\varepsilon}{\kappa(B^{1/2})} \|\tilde{g}\|. \tag{5.32}$$

Combining these intermediate results with $\left\|B^{-1/2}\right\| = \sigma_{\max}(B^{-1/2})$, $\kappa(B^{1/2}) = \sigma_{\max}(B^{1/2})/\sigma_{\min}(B^{1/2})$ and $\kappa(B^{1/2}) = \kappa(B)^{1/2}$ we arrive at

$$\|e_g\| \leq \frac{\varepsilon}{\kappa(B)^{1/2}} \|\tilde{g}\| \quad \Rightarrow \quad \frac{(g, B^{-1}\tilde{g})}{\|g\| \left\|B^{-1}\tilde{g}\right\|} \geq \frac{(1 - 2\varepsilon)}{\kappa(B)^{1/2}}, \tag{5.33}$$

which shows the claim. □

This lemma shows that $-B^{-1}\tilde{g}$ is a descent direction and fulfills the angle condition (5.17) with $\alpha = (1-2\varepsilon)/\sqrt{\kappa(B)}$, if the error in the reduced gradient is small enough. For the convergence speed as well as the effectivity of the compression method it is important that the condition number of the updated Hessian approximation B does not become too large.

With this result it is easy to ensure that the inexact BFGS update preserves positive definiteness of B_i.

Lemma 5.3.2. *Let B_0 be symmetric positive definite. Then all Hessian approximations $B_{i+1}, i = 0, 1, \ldots$ generated by (5.23) with inexact $\tilde{s}_i, \tilde{\gamma}_i$ are symmetric and positive definite.*

Proof. Let B_i be symmetric positive definite. Then $\tilde{p}_i$ is a descent direction, and there exists a step length $\tilde{\alpha}_i$ such that the inexact Wolfe-Powell conditions (5.24), (5.25) hold (see Remark 5.3.3). By (5.25)

$$(\tilde{\gamma}_i, \tilde{s}_i) \geq \tilde{\alpha}_i(\tilde{c}_2 - 1)(\tilde{g}_i, \tilde{p}_i) > 0,$$

such that the BFGS update is well-defined. As B_i and the update are symmetric, B_{i+1} is symmetric as well.

As B_i is positive definite we get $(x, B_{i+1}x) > 0$ for an arbitrary $x \neq 0$ by application of the Cauchy-Schwarz inequality, as in the exact case. Using the update formula, we evaluate

$$(x, B_{i+1}x) = (x, B_i x) - \frac{(B_i\tilde{s}_i, x)^2}{(\tilde{s}_i, B_i\tilde{s}_i)} + \frac{(\tilde{\gamma}_i, x)^2}{(\tilde{\gamma}_i, \tilde{s}_i)}$$

If $x, \tilde{s}_i$ are linearly independent, we estimate

$$(B_i\tilde{s}_i, x)^2 = (B_i^{1/2}\tilde{s}_i, B_i^{1/2}x)^2 < (B_i^{1/2}\tilde{s}_i, B_i^{1/2}\tilde{s}_i)(B_i^{1/2}x, B_i^{1/2}x) = (B_i\tilde{s}_i, \tilde{s}_i)(B_i x, x),$$

hence $(x, B_{i+1}x) > 0$. If on the other hand $x = \sigma\tilde{s}$ for some $\sigma \in \mathbb{R}$, $(B_i\tilde{s}_i, x)^2 = (B_i\tilde{s}_i, \tilde{s}_i)(B_i x, x)$ and $(x, B_{i+1}x) > 0$ as $(\tilde{\gamma}_i, x)^2/(\tilde{s}_i, \tilde{\gamma}_i) > 0$. □

Remark 5.3.3. Felgenhauer [33] showed that every step length α satisfying the exact Wolfe-Powell conditions with constants c_1, c_2 fulfills the inexact Wolfe-Powell conditions with some other constants $\tilde{c}_1, \tilde{c}_2$, and vice versa, if the error in the gradient is sufficiently small. In the present setting, existence of α, and thus of $\tilde{\alpha}_i$ in the proof of Lemma 5.3.2, follows from standard arguments, see, e.g., [98].

Now global convergence—for convex problems—follows.

Theorem 5.3.4. *Let the reduced Hessian $j''(u)$ be positive definite for all $u \in U$. For the BFGS-quasi-Newton method* (5.20), (5.23) *with inexact gradients, let the error bound* (5.26) *hold. Then*

$$\lim_{i\to\infty} j'(u_i) = 0.$$

Proof. As B_i is symmetric and positive definite, $\tilde{p}_i = -B_i^{-1}\tilde{g}_i$ is a descent direction and fulfills the angle condition (5.17). This immediately yields convergence of the BFGS-method with inexact gradients. □

5.3.2. Superlinear Convergence

Superlinear convergence for the BFGS method with exact quantities is a well-known result [98]. Some work is also dealing with perturbed quasi-Newton methods, e.g. [33, 88].

In the exact case, superlinear convergence is usually characterized by the condition

$$\lim_{i\to\infty} \frac{\left\|(B_i - H_\star)p_i\right\|}{\|p_i\|} = 0, \tag{5.34}$$

with $p_i = -B_i^{-1}g_i$ and $H_\star$ denoting the Hessian at the minimizer $u^\star$.

For the inexact case, we need an additional condition for the gradient error. In the following we abbreviate the Hessian at u_i by $H_i = H(u_i)$.

Theorem 5.3.5. *Let u_i be the sequence of iterates generated by $u_{i+1} = u_i + \tilde{p}_i$, $\tilde{p}_i = -B_i^{-1}\tilde{g}_i$, i.e. the update* (5.20) *with step length $\alpha_i = 1$. Assume that $u_i \to u^\star$ linearly, and the Hessian is positive definite and Lipschitz continuous at the minimizer $u^\star$. Let the following two conditions hold:*

$$\lim_{i\to\infty} \frac{\left\|(B_i - H_\star)\tilde{p}_i\right\|}{\|\tilde{p}_i\|} = 0 \tag{5.35}$$

$$\lim_{i\to\infty} \frac{\left\|e_{g_i}\right\|}{\|\tilde{g}_i\|} = 0. \tag{5.36}$$

Then $u_i \to u^\star$ superlinearly.

Proof. We follow the proof of [98, Thm. 3.7]. With the Newton step $p_i^N = H_i^{-1}g_i$ the conditions (5.35), (5.36) imply

$$\lim_{i\to\infty} \frac{\left\|\tilde{p}_i - p_i^N\right\|}{\|\tilde{p}_i\|} = 0$$

as

$$\tilde{p}_i - p_i^N = H_i^{-1}\big(H_i\tilde{p}_i + g_i\big) = H_i^{-1}\big(H_i\tilde{p}_i - B_i\tilde{p}_i - e_{g_i}\big),$$

and $\lim_{i\to\infty} \|e_{g_i}\|/\|\tilde{p}_i\| = 0$ by (5.36) combined with $\|\tilde{p}_i\| \geq \sigma_{\min}(B^{-1})\|\tilde{g}_i\|$. By quadratic convergence of the exact Newton method and using

$$\|u_i + \tilde{p}_i - u^\star\| \leq \|u_i + p_i^N - u^\star\| + \|\tilde{p}_i - p_i^N\|$$

we get

$$\lim_{i\to\infty} \frac{\|u_i + \tilde{p}_i - u^\star\|}{\|u_i - u^\star\|} = 0.$$

□

Lemma 5.3.6. *For the BFGS-quasi-Newton method* (5.20), (5.23) *with inexact gradients,*

$$\lim_{i\to\infty} \frac{\big\|\big(B_i - H_\star\big)\tilde{s}_i\big\|}{\|\tilde{s}_i\|} = 0 \tag{5.37}$$

holds, if $u_i \to u^\star$ linearly.

Proof. The proof follows along the lines of [98, Thm. 6.6] for the exact case. Define

$$\overline{B}_i = H_\star^{-\frac{1}{2}} B_i H_\star^{-\frac{1}{2}}, \quad \overline{\tilde{s}}_i = H_\star^{\frac{1}{2}}\tilde{s}_i, \quad \overline{\tilde{\gamma}}_i = H_\star^{-\frac{1}{2}}\tilde{\gamma}_i,$$

and, for brevity,

$$\cos\overline{\tilde{\Theta}}_i = \frac{(\overline{\tilde{s}}_i, \overline{B}_i\overline{\tilde{s}}_i)}{\big\|\overline{\tilde{s}}_i\big\|\,\big\|\overline{B}_i\overline{\tilde{s}}_i\big\|}, \quad \overline{\tilde{q}}_i = \frac{(\overline{\tilde{s}}_i, \overline{B}_i\overline{\tilde{s}}_i)}{\big\|\overline{\tilde{s}}_i\big\|^2}, \quad \overline{\tilde{M}}_i = \frac{\big\|\overline{\tilde{\gamma}}_i\big\|^2}{(\overline{\tilde{\gamma}}_i, \overline{\tilde{s}}_i)}, \quad \overline{\tilde{m}}_i = \frac{(\overline{\tilde{\gamma}}_i, \overline{\tilde{s}}_i)}{\big\|\overline{\tilde{s}}_i\big\|^2}.$$

Multiplying the inexact variant of the BFGS update (5.23) by $H_\star^{-\frac{1}{2}}$ from left and right yields the update

$$\overline{B}_{i+1} = \overline{B}_i - \frac{(\overline{B}_i\overline{\tilde{s}}_i, \cdot)\overline{B}_i\overline{\tilde{s}}_i}{(\overline{\tilde{s}}_i, \overline{B}_i\overline{\tilde{s}}_i)} + \frac{(\overline{\tilde{\gamma}}_i, \cdot)\overline{\tilde{\gamma}}_i}{(\overline{\tilde{s}}_i, \overline{\tilde{\gamma}}_i)}.$$

Using

$$\overline{\tilde{\gamma}}_i - \overline{\tilde{s}}_i = H_\star^{-\frac{1}{2}}(\tilde{\gamma}_i - H_\star\tilde{s}_i)$$

and $\tilde{\gamma}_i = \overline{H}_i\tilde{s}_i$ with $\overline{H}_i = \displaystyle\int_0^1 H(u_i + \tau\tilde{\alpha}_i\tilde{p}_i)\, d\tau$ yields after a short computation

$$\big\|\overline{\tilde{\gamma}}_i - \overline{\tilde{s}}_i\big\| \leq \big\|H_\star^{-\frac{1}{2}}\big\|^2\,\big\|\overline{\tilde{s}}_i\big\|\,Ld_i,$$

where L is the Lipschitz constant of $H_\star$ and $d_i = \max\{\|u_{i+1} - u^\star\|, \|u_i - u^\star\|\}$. For a constant $c_0 \geq L\big\|H_\star^{-\frac{1}{2}}\big\|^2$ we thus get

$$\frac{\big\|\overline{\tilde{\gamma}}_i - \overline{\tilde{s}}_i\big\|}{\big\|\overline{\tilde{s}}_i\big\|} \leq c_0 d_i,$$

and by the triangle inequality

$$\begin{aligned}\big\|\overline{\tilde{\gamma}}_i\big\| - \big\|\overline{\tilde{s}}_i\big\| &\leq c_0 d_i \big\|\overline{\tilde{s}}_i\big\| \\ \big\|\overline{\tilde{s}}_i\big\| - \big\|\overline{\tilde{\gamma}}_i\big\| &\leq c_0 d_i \big\|\overline{\tilde{s}}_i\big\|.\end{aligned}$$

Together this gives

$$(1 - c_0 d_i)\big\|\overline{\tilde{s}}_i\big\| \leq \big\|\overline{\tilde{\gamma}}_i\big\| \leq (1 + c_0 d_i)\big\|\overline{\tilde{s}}_i\big\|,$$

and the estimate $\overline{\tilde{m}}_i \geq 1 - c_0 d_i$. Using $u_i \to u^\star$, there exists a constant $c \geq c_0$ such that $\overline{\tilde{M}}_i \leq 1 + c d_i$ for sufficiently large i.

As we work in a finite dimensional setting due to the fixed discretization, with $\psi(B) = \operatorname{trace}(B) - \ln(\det(B))$ we have

$$\psi(\overline{B}_{i+1}) = \psi(\overline{B}_i) + \left(\overline{\tilde{M}}_i - \ln(\overline{\tilde{m}}_i) - 1\right) + \left(1 - \frac{\overline{\tilde{q}}_i}{\cos^2\overline{\tilde{\Theta}}_i} + \ln\big(\frac{\overline{\tilde{q}}_i}{\cos^2\overline{\tilde{\Theta}}_i}\big)\right) + \ln(\cos^2\overline{\tilde{\Theta}}_i).$$

For sufficiently large i, we can assume $c_0 d_i < \frac{1}{2}$ and

$$\ln(\overline{\tilde{m}}_i) \geq \ln(1 - c_0 d_i) \geq -2 c_0 d_i > -2 c d_i.$$

Thus,

$$0 < \psi(\overline{B}_{i+1}) \leq \psi(\overline{B}_i) - 3 c d_i + \ln(\cos^2\overline{\tilde{\Theta}}_i) + \left(1 - \frac{\overline{\tilde{q}}_i}{\cos^2\overline{\tilde{\Theta}}_i} + \ln\big(\frac{\overline{\tilde{q}}_i}{\cos^2\overline{\tilde{\Theta}}_i}\big)\right).$$

Summing up, again using that by linear convergence $\sum_{i=1}^{\infty} \|u_i - u^\star\| < \infty$, we arrive at

$$\sum_{i=1}^{\infty} \underbrace{-\ln(\cos^2\overline{\tilde{\Theta}}_i)}_{\geq 0} \underbrace{- \left(1 - \frac{\overline{\tilde{q}}_i}{\cos^2\overline{\tilde{\Theta}}_i} + \ln\big(\frac{\overline{\tilde{q}}_i}{\cos^2\overline{\tilde{\Theta}}_i}\big)\right)}_{(\star)} \leq \psi(\overline{B}_0) + 3c \sum_{i=1}^{\infty} d_i < \infty.$$

As $\overline{\tilde{q}}_i / \big(\cos^2\overline{\tilde{\Theta}}_i\big) > 0$ and $1 - x + \ln(x) \leq 0 \ \forall x > 0$, the term $(\star)$ is non-positive. We conclude that $\lim\limits_{i\to\infty} \cos\overline{\tilde{\Theta}}_i = 1, \quad \lim\limits_{i\to\infty} \overline{\tilde{q}}_i = 1.$

Finally, we have

$$\frac{\left\|(\overline{B}_i - \mathrm{Id})\bar{\tilde{s}}_i\right\|}{\left\|\bar{\tilde{s}}_i\right\|} = \frac{\bar{\tilde{q}}_i^2}{\cos^2 \bar{\tilde{\Theta}}_i} - 2\bar{\tilde{q}}_i + 1 \to 0 \quad \text{for } i \to \infty.$$

As $(\overline{B}_i - \mathrm{Id})\bar{\tilde{s}}_i = H_\star^{-\frac{1}{2}}(B_i - G_\star)\tilde{s}_i$, the claim follows. □

Additionally, we have to show that the step length $\alpha_i = 1$ is admissible for all $i \geq i_0$ for a certain iteration index i_0.

Lemma 5.3.7. *Let the assumptions of Theorem 5.3.5 hold. Then there exists an iteration index i_0 such that $\alpha_i = 1$ satisfies the exact Wolfe-Powell conditions* (5.21), (5.22) *for all iterations $i > i_0$.*

Proof. Consider

$$\frac{\left\|g_i + H_i\tilde{p}_i\right\|}{\|\tilde{p}_i\|} = \frac{\left\|-B_i\tilde{p}_i - e_{g_i} + H_i\tilde{p}_i\right\|}{\|\tilde{p}_i\|} \leq \frac{\left\|-B_i\tilde{p}_i + H_i\tilde{p}_i\right\|}{\|\tilde{p}_i\|} + \frac{\|e_{g_i}\|}{\|\tilde{p}_i\|}.$$

As $u_i \to u^\star, i \to \infty$, we have $H_i \to H_\star$ and thus

$$\lim_{k\to\infty} \frac{\left\|-B_i\tilde{p}_i + H_i\tilde{p}_i\right\|}{\|\tilde{p}_i\|} = 0$$

due to (5.37). As $\lim_{i\to\infty} \|e_{g_i}\|/\|\tilde{p}_i\| = 0$ by (5.36) we have

$$\lim_{i\to\infty} \frac{\left\|g_i + H_i\tilde{p}_i\right\|}{\|\tilde{p}_i\|} = 0.$$

This well-known condition yields the claim, see, e.g., [21, Thm. 6.4]. □

Remark 5.3.8. By Lemma 5.3.7, $\alpha_i = 1$ satisfies the exact Wolfe-Powell conditions. By Remark 5.3.3 it satisfies the inexact Wolfe-Powell conditions as well, for slightly different constants. With $\tilde{\alpha}_i = 1$, $\tilde{s}_i = \tilde{p}_i$ and the condition (5.37) in Lemma 5.3.6 is the same as (5.35) in Theorem 5.3.5.

To fulfill the second condition (5.36) of Theorem 5.3.5 concerning the inexactness of the gradients, we have to use a tighter error bound for the gradient error. This can be achieved by letting $\varepsilon \to 0$ for $i \to \infty$ in condition (5.26), yielding a condition comparable to accuracy requirements for the inexact Newton-CG method, see Section 5.4.

Lemma 5.3.9. *In iteration i of the BFGS-quasi-Newton method, let*

$$\varepsilon_i = \min\{1/2, \sqrt{\|\tilde{g}_i\|}\} \quad \textit{and} \quad \|e_{g_i}\| \leq \frac{\varepsilon_i}{\kappa(B_i)^{1/2}} \|\tilde{g}_i\| . \tag{5.38}$$

Then $\lim_{i\to\infty} \|e_{g_i}\|/\|\tilde{g}_i\| = 0$.

Proof. By convergence of the inexact BFGS method, we have $\|\tilde{g}_i\| \to 0$, such that, after a certain iteration i_0, $\sqrt{\|\tilde{g}_i\|} < 1/2 \ \forall i \geq i_0$. Thus $\varepsilon_i \to 0$ for $i \to \infty$ and the result follows, as

$$\lim_{i\to\infty} \frac{\|e_{g_i}\|}{\|\tilde{g}_i\|} \leq \lim_{i\to\infty} \frac{\varepsilon_i}{\kappa(B_i)^{1/2}},$$

and $\kappa(B_i)$ is bounded. □

5.3.3. Remarks

For the evaluation of the quantization tolerance we need to estimate the condition number of B_i. An update formula for the condition number was derived by Hoh Phua [58]. They make use of a Cholesky-factorization of B_i^{-1} to derive the condition number of B_{i+1}^{-1} for SR1- and BFGS-quasi-Newton updates. From the BFGS-update (5.23) it is possible to construct methods updating the Cholesky-factors directly, which then can be used for condition estimation as well as step computation [98].

From the BFGS update formula (5.23), a corresponding update for the inverse of the Hessian approximation can be computed via the Sherman-Morrison-Woodbury formula, allowing to compute $B_i^{-1}\tilde{g}_i$ without solving the linear system. When, for large scale problems, storing this typically dense matrix is prohibitive, we can resort to a matrix free implementation, computing the matrix-vector product from a initial approximation B_0^{-1} and the update vectors $\tilde{s}_i, \tilde{\gamma}_i, i = 0, 1, \ldots$, see, e.g., [97, 137]. Besides keeping all update vectors, this can be used as a limited memory version (L-BFGS) by discarding all but M most recent vectors, at the cost of slower convergence speed.

If the reduced Hessian is not positive definite, the BFGS-quasi-Newton method might fail, for example due to violation of the condition $(y, s) > 0$. Exemplary strategies to overcome such problems are the use of damped BFGS updates (cf. [98] and the references therein), or regularization strategies (e.g. [83]).

While the BFGS-update formula in outer-product form, as given in equation (5.23) can directly be used to construct a function space algorithm, the convergence analysis becomes more involved. Early works giving convergence criteria include [90, 44, 106,

71, 72]. Kupfer [77] investigates reduced SQP methods, using quasi-Newton updates of the convex Broyden family (including BFGS). The following conditions for the choice of the update sequences $\{\gamma_i\}_i, \{s_i\}_i$ are needed to prove convergence in the infinite dimensional setting:

$$(\gamma_i, s_i)_U > 0, \qquad \left\|\gamma_i - H_\star s_i\right\| \leq \varepsilon_i \left\|s_i\right\| \quad \forall i, \quad \text{and} \sum_i \varepsilon_i < \infty.$$

These conditions can serve as a starting point to develop accuracy requirements for inexact gradients due to discretization errors and lossy trajectory storage.

If the initial Hessian approximation B_0 is self-adjoint and positive, and $B_0 - H_\star$ small enough, the quasi-Newton method yields convergence $u_i \to u^\star$. If further $B_0 - H_\star$ is compact, the convergence is two-step superlinear, i.e.

$$\lim_{i\to\infty} \frac{\left\|u_{i+1} - u^\star\right\|}{\left\|u_{i-1} - u^\star\right\|} = 0.$$

For optimal control problems, the compactness condition can be fulfilled using

$$B_0 = \mathcal{L}_{uu}(y^\star, u^\star, \lambda^\star)$$

[64], where $\mathcal{L}$ denotes the Lagrange functional (see Section 2.2). For typical examples with a quadratic tracking-type objective functional and state equations with linearly entering controls, $\mathcal{L}_{uu}(y^\star, u^\star, \lambda^\star) = \alpha\mathrm{Id}$, where α here denotes the regularization parameter and Id the identity operator of the control space U.

In this section we restricted the exposition to the BFGS-quasi-Newton method. The convergence behavior of other updates using inexact gradient information remains to be investigated.

5.4. Newton-CG

In this section we analyze the quantization accuracy required for the convergence of Newton-CG methods in detail. Specific to an optimal control problem in cardiac defibrillation, adaptive quantization for the Newton-CG method was introduced in [37], see also Chapter 6. Here, we generalize and extend these results, based on the error equations of Section 5.1.

We assume that we are in a neighborhood of a local minimizer, such that the reduced Hessian $j''(u)$ is positive definite. In the Newton-CG algorithm, the Newton

direction is approximately computed by applying the conjugate gradient method to the Newton equation

$$j''(u)\delta u = -j'(u).$$

Due to termination of the CG algorithm with a non-zero residual, as well as lossy compression of state, adjoint, and linearized-state trajectories, we compute

$$j''(u)\delta u = -j'(u) + e_{j'} + \tilde{r},$$

where $\tilde{r}$ is the inexactly computed residual. For convergence we require for the true residual

$$\|r\| \leq \rho \left\|j'(u)\right\|, \ 0 < \rho < 1,$$

with $\rho \to 0$ for super-linear convergence [28, 23]. As $\|r\| \leq \|\tilde{r}\| + \|\tilde{r} - r\|$, we need to control three error contributions. Thus we have to ensure

$$\|e_{j'}\| + \|\tilde{r}\| + \|\tilde{r} - r\| \leq \rho \|j'(u)\|. \tag{5.39}$$

As $\|j'(u)\| = \|\tilde{j}'(u) - e_{j'}\| \geq \|\tilde{j}'(u)\| - \|e_{j'}\|$, equation (5.39) is replaced by

$$(1+\rho)\|e_{j'}\| + \|\tilde{r}\| + \|\tilde{r} - r\| \leq \rho \|\tilde{j}'(u)\|, \tag{5.40}$$

which is fulfilled, if for $\zeta_1, \zeta_2 \in (0,1), \ \zeta_1 + \zeta_2 < 1$

$$\left\|e_{j'}\right\| \leq \zeta_1 \rho \left\|\tilde{j}'(u)\right\| / (1+\rho), \ \ \|\tilde{r}\| \leq \zeta_2 \rho \left\|\tilde{j}'(u)\right\|, \ \ \|\tilde{r} - r\| \leq (1 - \zeta_1 - \zeta_2)\rho \left\|\tilde{j}'(u)\right\| \tag{5.41}$$

hold.

We discuss these three accuracy conditions in the following.

5.4.1. Adaptive Quantization for Gradient Computation

To satisfy the accuracy requirement $\|e_{j'}\| \leq \zeta_1 \rho \|j'(u)\|$ for the reduced gradient, we have to determine a suitable quantization tolerance δ^y *before* solving state and adjoint equations.

Theorem 5.4.1. *In iteration i of the Newton method, define*

$$\mu_i = \|c_u(\hat{y}, u_i)^\star \overline{e_\lambda}\| \tag{5.42}$$

with the error estimate $\overline{e_\lambda}$ from Theorem 5.1.6 or 5.1.8 for $\varepsilon_y^{max} = 1$. Let $\theta \leq \left\|\tilde{j}'(u_{i+1})\right\|$ be an estimate for the inexact reduced gradient norm in iteration $i+1$. If the state quantization tolerance δ_{i+1}^y satisfies

$$\delta_{i+1}^y \leq \frac{\theta \zeta_1 \rho_{i+1}}{(1+\rho_{i+1})\mu_i}, \tag{5.43}$$

$\left\|e_{j',i+1}\right\| \leq \zeta_1 \rho_{i+1} \left\|\tilde{j}'(u_{i+1})\right\| / (1+\rho_{i+1})$ *holds.*

Proof. For the error in the adjoint we have $e_\lambda \leq \overline{e_\lambda}$ for $\varepsilon_y^{\max} = 1$. Thus by scaling and using monotonicity of $c_u(\hat{y}, u)^\star$, we get $\|c_u(\hat{y}, u_i)^\star e_\lambda\| \leq \delta \|c_u(\hat{y}, u_i)^\star \overline{e_\lambda}\|$ for $\varepsilon_y^{\max} = \delta$. This yields

$$\left\|e_{j',i+1}\right\| \leq \delta_{i+1}^y \|c_u(\hat{y}, u_i)^\star \overline{e_\lambda}\| \leq \frac{\theta\zeta_1\rho_{i+1}}{1+\rho_{i+1}} \leq \frac{\zeta_1\rho_{i+1}}{1+\rho_{i+1}} \left\|\tilde{j}'(u_{i+1})\right\|.$$

□

For a computationally available approximation of μ_i, we refer to Section 5.4.3, see especially equation (5.55).

Remark 5.4.2. For implementation, we point to the following difficulties:

1. As a computationally available approximation of θ, we can choose

$$\tilde{\theta} = \frac{\left\|\tilde{j}'(u_i)\right\|^2}{\left\|\tilde{j}'(u_{i-1})\right\|},$$

   assuming linear convergence. If we aim at super-linear convergence of the Newton-CG method, the gradient-norm estimate $\tilde{\theta}$ has to be adapted accordingly, for example using $\tilde{\theta} = \rho_i \left\|\tilde{j}'(u_i)\right\|$ (see [98]).

2. As we only *approximate* the worst-case error $\overline{e_\lambda}$ and the gradient norm of the next iteration, in practice we can not guarantee to keep the error bound $\|e_{j'}\| \leq \zeta_1\rho_{i+1} \|j'(u_{i+1})\| / (1+\rho_{i+1})$. Multiplication of δ_{i+1}^y by some safety factor might be necessary to avoid impeding the convergence, depending on the actual problem. However, as typically the error is significantly over-estimated, no safety factor was needed in the numerical examples.

5.4.2. Adaptive Quantization for Hessian-Vector Products

In the CG method, we have to ensure that on exit the remaining two bounds

$$\|\tilde{r}\| \leq \zeta_2\rho \left\|\tilde{j}'(u)\right\|, \quad \|\tilde{r} - r\| \leq (1-\zeta_1-\zeta_2)\rho \left\|\tilde{j}'(u)\right\|$$

are satisfied. While the condition for the inexact residual is fulfilled by using it as a termination criterion for the CG, the bound on the inexactness of the computed residuals is more demanding.

The quantization error in state, adjoint, and linearized-state leads to an inexact Hessian-vector product in the CG iterations, see step 3 of Algorithm 4. Consequently, approximate residuals $\tilde{r}^k$ are computed in iteration k of the CG algorithm instead of the true residuals r^k.

Algorithm 4 CG for solving $j''(u)\delta u = -\tilde{j}'(u)$ with inexact matrix-vector products

set $k = 0, \delta u^0 = 0, \tilde{r}^0 = \tilde{j}'(u), p^0 = -\tilde{r}^0$
while $\|\tilde{r}^k\| > \text{TOL}$ **do**
  $q^k = j''(u)p^k + e^k_{\text{mv}}$
  $\alpha^k = (\tilde{r}^k, \tilde{r}^k)/(q^k, p^k)$
  $\delta u^{k+1} = \delta u^k + \alpha^k p^k$
  $\tilde{r}^{k+1} = \tilde{r}^k + \alpha^k q^k$
  $\beta^k = (\tilde{r}^{k+1}, \tilde{r}^{k+1})/(\tilde{r}^k, \tilde{r}^k)$
  $p^{k+1} = -\tilde{r}^{k+1} + \beta^k p^k$
  $k \leftarrow k + 1$
end while

5.4.2.1. Quantization of v

First, we consider only the error contribution of the quantization of the linearized-state solution. Due to compression of this trajectory, the Hessian-vector products contain an error which might change in every CG iteration. Krylov subspace methods with inexact matrix-vector products have been discussed, e.g., by Simoncini and Szyld [114], and van den Eshof and Sleijpen [130]. Adapted to our problem setting, the theory presented there leads to the following Lemma.

Lemma 5.4.3. *If, for a certain value l_m, in all CG iterations $i < m$,*

$$\left\|e^k_{\text{mv}}\right\| \leq l_m \frac{\varepsilon}{\|\tilde{r}^k\|} \tag{5.44}$$

holds, then $\|\tilde{r}^m - r^m\| \leq \varepsilon$.

In [37], we adapted their work to our setting, proposing a quantization tolerance

$$\delta^v_k \leq \frac{l_m}{\mu_i} \frac{(1 - \zeta_1 - \zeta_2)\rho_i \left\|\tilde{j}'(u_i)\right\|}{\|\tilde{r}^k\|} \tag{5.45}$$

for the linearized-state trajectory in iteration k of the CG (i denotes the Newton iteration). Similar to Theorem 5.4.1, in equation (5.45) $\mu_i = \|c_u(\hat{y}, u_i)^\star \overline{e_w}\|$ with a worst case error bound $\overline{e_w}$ for the error in the adjoint-for-Hessian solution.

The choice (5.45) for the quantization tolerance suffers from the fact that neither l_m nor μ_i is known exactly. The value of l_m given in [114] is, unfortunately, computationally unavailable. In order to avoid computational overhead, the error bound for e_w is best computed on a coarse mesh, which leads to an inexact value of μ_i. Consequently, $\|r\| \leq \text{TOL}_{\text{CG}}$ can not be guaranteed in practice. This may cause

the norm of the true residual to stay far above the required tolerance, while $\|\tilde{r}\|$ decreases further.

In practice, the inaccuracy of μ_i turns out to be of little consequence, see Section 5.2. The other factor, l_m, is of more importance. A heuristic value $l_m = \lambda_{\min}/m_{\max}$ has been proposed, where $\lambda_{\min}$ denotes the smallest eigenvalue of the reduced Hessian matrix j''. As a computational estimate thereof, the (inexact) Rayleigh quotient can be used,

$$\lambda_{\min} \lesssim \min_k \frac{\left(j''(u_i)p^k + e^k_{\mathrm{mv}}, p^k\right)}{(p^k, p^k)} = \min_k \frac{(q^k, p^k)}{(p^k, p^k)},$$

where the minimum is taken over all CG iterations. Note that due to the inexactness of the matrix-vector-product, underestimation of $\lambda_{\min}$ is possible, leading to a smaller-than-necessary quantization tolerance.

Residual replacement. As discussed in the previous paragraph, $\lambda_{\min}/m$ is used as a heuristic for the unavailable, problem-dependent value l_m. Combination with a restart strategy—whenever significantly smaller value for $\lambda_{\min}$ is encountered, the CG method is started new using the current δu^k instead of δu^0—yielded good results, see [37]. Such a restart approach was necessary, as due to the unknown true values of l_m and $\lambda_{\min}$ the accuracy requirement can not be guaranteed to hold. In the following, we replace this heuristic restart strategy by a different, theoretically better justified approach, that avoids a complete restart of the CG by tracking the computed residual error and re-computing the residual if needed. It is motivated by the analysis of CG in finite precision presented in Greenbaum [43], and Gutknecht and Strakoš [48].

If we consider inexact Hessian-vector products, the iterates in the CG satisfy

$$\delta u^{k+1} = \delta u^k + \alpha^k p^k \tag{5.46}$$

$$\tilde{r}^{k+1} = \tilde{r}^k - \alpha^k j''(u_i)p^k + \xi^{k+1} \tag{5.47}$$

with direction p^k and $\alpha^k = (\tilde{r}^k, \tilde{r}^k)/(j''(u_i)p^k + e^k_{\mathrm{mv},v}, p^k)$. Here $\xi^{k+1} = -\alpha^k e^k_{\mathrm{mv},v}$ with $e^k_{\mathrm{mv},v}$ denoting the error in the computed product $j''(u_i)p^k$ due to compressed storage of the linearized-state v. By equation (5.46) we can evaluate the true residual belonging to the iterate δu^{k+1}, and calculate the difference to the updated residual $\tilde{r}^{k+1}$ using the recurrence (5.47) as

$$\underbrace{\tilde{j}'(u_i) + j''(u_i)\delta u^{k+1}}_{=r^{k+1}} - \tilde{r}^{k+1} = \tilde{j}'(u_i) + j''(u_i)\delta u^0 - \tilde{r}^0 - \sum_{j=1}^{k+1} \xi^j.$$

Choosing $\delta u^0 = 0$ allows to estimate the error in the residual as

$$\left\| r^{k+1} - \tilde{r}^{k+1} \right\| \leq \sum_{j=0}^{k} |\alpha^j| \left\| e^j_{\mathrm{mv},v} \right\| =: E^{k+1}. \tag{5.48}$$

Thus by estimating an upper bound for $\left\| e^k_{\mathrm{mv},v} \right\|$ we can cheaply monitor the error in the computed residual, and re-compute the residual from the current iterate δu^{k+1} when the error in the residual becomes too large, thus avoiding a restart strategy based on $\lambda_{\min}$.

Residual replacement strategies were developed, e.g., by Sleijpen and van der Vorst [115], as well as van der Vorst and Ye [131]. Analogously to the latter, we trigger the restart in iteration k, when the estimated, accumulated residual error E fulfills

$$E^k > \epsilon \left\| \tilde{r}^k \right\|, \quad E^k > 1.1 E^{\mathrm{init}}, \tag{5.49}$$

where ϵ is a given threshold parameter, and E^{init} is the estimated error at the last restart (respectively the estimated error of the initial Hessian-vector product, if no restart was triggered before). On a restart, we replace the current residual $\tilde{r}^k$ by $j''(u_i)\delta u^k + \tilde{j}'(u_i)$. For the evaluation of the inexact Hessian-vector product, δ^v is multiplied by some factor $s^v < 1$, such that the linearized state is stored more accurately.

A numerical comparison of the restart approach and residual replacement strategy can be found in Section 6.3.2.

5.4.2.2. Quantization of y and λ

Besides the inexact linearized-state solution, quantization of the state y and the adjoint λ contribute to the error in the Hessian-vector products. Choosing suitable tolerances δ^y, δ^λ before solving state and adjoint equations poses the main difficulty.

The error e_λ contributes only to the error e_z in the right-hand side of the adjoint-for-Hessian error equation (5.8). Considering the estimate (5.10), and neglecting products of errors, $\|e_\lambda\|$ is weighted by $\|c_{yy}(\hat{y}, u) - c_{yyy}(\hat{y}, u)\varepsilon_y\| \left\| \hat{\tilde{v}} \right\|$. Thus, for iteration $i+1$ of the Newton method, we seek to fulfill the bound

$$\|e_\lambda\| \leq \frac{\mathrm{TOL}_\lambda}{\|c_{yy}(\hat{y}, u_i) - c_{yyy}(\hat{y}, u_i)\varepsilon_y\| \left\| \hat{\tilde{v}} \right\|}$$

by choosing δ^y as

$$\delta^y_{i+1} \leq \frac{\mathrm{TOL}_\lambda}{\|c_{yy}(\hat{y},u_i) - c_{yyy}(\hat{y},u_i)\varepsilon_y\| \left\|\hat{\tilde{v}}\right\|} \frac{1}{\|\overline{e_\lambda}\|}. \tag{5.50}$$

As before, $\overline{e_\lambda}$ is the worst case error in the adjoint given by Theorem 5.1.6 or Theorem 5.1.8, respectively.

For the evaluation of equation (5.50), an estimate for $\left\|\hat{\tilde{v}}\right\|$ has to be provided. Apart from restarts, $\tilde{v}$ is determined by the linear parabolic equation

$$c_y(\hat{y},u_i)\tilde{v} = c_u(\hat{y},u_i)p^k$$

in CG iteration k. As $p^0 = r^0 = -\tilde{j}'(u_i)$ we estimate $\left\|\hat{\tilde{v}}\right\| \leq c\left\|\tilde{j}'(u_i)\right\|$, where the unknown constant c is replaced by some $\tilde{c}$ large enough, depending on the actual problem. This is motivated by the fact that for exact CG, $\left\|p^k\right\| \sim \left\|r^k\right\|$. For the choice of TOL_λ, we aim to achieve the same error level as in the reduced gradient, i.e. $\mathrm{TOL}_\lambda = \zeta_1\rho_{i+1}\left\|\tilde{j}'(u_{i+1})\right\|/(1+\rho_{i+1})$. Combined, equation (5.50) becomes

$$\delta^y_{i+1} \leq \frac{\zeta_1\rho_{i+1}}{\tilde{c}(1+\rho_{i+1})\|c_{yy}(\hat{y},u_i) - c_{yyy}(\hat{y},u_i)\varepsilon_y\|} \frac{1}{\|\overline{e_\lambda}\|}. \tag{5.51}$$

For the quantization of the adjoint λ it is sufficient to keep the quantization error ε_λ well below the error e_λ. This can be achieved by choosing

$$\delta^\lambda_{i+1} \leq s^\lambda \frac{\mathrm{TOL}_\lambda}{\|c_{yy}(\hat{y},u_i) - c_{yyy}(\hat{y},u_i)\varepsilon_y\| \left\|\hat{\tilde{v}}\right\|} \tag{5.52}$$

for some $0 < s^\lambda \ll 1$.

5.4.3. Realization

To fix the details, we present an algorithm for a Newton-CG method using lossy compression with adaptive quantization tolerances. The residual replacement strategy is used to avoid complete restarts. For better readability, we focus on the important steps and quantities and do not give a complete algorithm. Moreover, we restrict the discussion to scalar problems; for reaction-diffusion systems, only the right-hand-sides of the error equations need to be changed to suitable super-reaction functions as given in Theorem 5.1.8.

In line 5 of Algorithm 5 we solve the error equation

$$c_y(\hat{y},u_i)^\star e = |c_{yy}(\hat{y},u_i)^\star|\delta^y_i e + \mathbf{1}. \tag{5.53}$$

Algorithm 5 Newton-CG with adaptive quantization

Input: $\delta_0^y, \delta_0^\lambda$, initial guess for control u_0

1: **for** $i = 0, 1, \dots$ **do**
2: solve state equation $c(y, u_i) = 0$, encode y using δ_i^y
3: solve adjoint equation $c(\hat{y}, u_i)^\star \tilde{\lambda} = -J_y(\hat{y}, u_i)$, encode $\tilde{\lambda}$ using δ_i^λ
4: check optimality condition; if optimal: stop
5: solve $c_y(\hat{y}, u_i)^\star e = \left| c_{yy}(\hat{y}, u_i)^\star \right| \delta_i^y e + \mathbf{1}$, keep $\|e\|, \|c_u(\hat{y}, u_i)^\star e\|$
6: compute Newton step using CG: set $\delta u^0 = 0, p^0 = \tilde{r}^0 = -\tilde{j}', E = 0, s^v = 1$, estimate $\lambda_{\min}$
7: **while** $\tilde{r}^k > \zeta_2 \rho_i \left\| \tilde{j}'(u_i) \right\|$ **and** $k < m$ **do**
8: solve linearized-state equation $c_y(\hat{y}, u_i)\tilde{v} = c_u(\hat{y}, u_i)p^k$, encode $\tilde{v}$ using $s^v \delta^v$, with δ^v given by (5.45)
9: solve adjoint-for-Hessian equation $c_y(\hat{y}, u_i)^\star \tilde{w} = \tilde{z}$
10: compute α^k, update $\delta u^{k+1}, \tilde{r}^{k+1}, p^{k+1}$
11: estimate error of Hessian-vector product $\left\| e_{\mathrm{mv},v}^k \right\|$
12: update $E \leftarrow E + |\alpha^k| \left\| e_{\mathrm{mv},v}^k \right\|$
13: **if** E satisfies the restart conditions (5.49) **then**
14: decrease safety factor $s^v \leftarrow 0.1 s^v$
15: restart CG by evaluating the residual $\tilde{r}^{k+1} = \tilde{j}'(u_i) + j''(u_i)\delta u^{k+1}$
16: **end if**
17: $k \leftarrow k + 1$
18: **end while**
19: estimate new values for $\delta_{i+1}^y, \delta_{i+1}^\lambda$
20: compute suitable step size s_i and update $u_{i+1} = u_i + s_i \delta u^k$
21: **end for**

Compared to Theorem 5.1.6, the terms $|\phi|$ and $|c_{yy}(\hat{y}, u_i)^\star \psi| e_y^{\max}$ are replaced by the constant **1**-function, allowing to re-use the solution by scaling with the appropriate right-hand-sides.

For the initialization of the CG method (line 6), an estimate for the smallest eigenvalue of the reduced Hessian is required. As proposed in [37], this can be done at the cost of an additional Hessian-vector product (computed on a coarse, fixed grid) by the Rayleigh quotient

$$\lambda_{\min} \leq \frac{\left(j''(u_i)p^0, p^0\right)}{(p^0, p^0)}.$$

During the CG, $\lambda_{\min}$ can be updated in each iteration k using

$$\lambda_{\min} \leq \min_{j=1,\dots,k} \frac{\left(j''(u_i)p^j + e_{\mathrm{mv}}^j, p^j\right)}{(p^j, p^j)},$$

as all quantities needed for the inexact Rayleigh quotient are available at no additional cost. For the determination of δ^v in line 8, $l_m = \lambda_{\min}/m$ is used in equation (5.45).

Further, μ_i needs to be specified. An estimate of $\overline{e_w}$ taking e_z into account is not available at this stage of the algorithm. We thus use $\mu_i \approx \|c_u(\hat{y}, u_i)^\star e\|$ computed in line 5 ignoring error contributions other than the quantization of the linearized-state trajectory.

For the estimation of $\left\|e^k_{\mathrm{mv},v}\right\|$ in line 11 we have to evaluate the error in the linearized-state by solving

$$c_y(\hat{y}, u_i)\overline{e_v} = \left|c_{yy}(\hat{y}, u_i)\right|\delta^y_i\overline{e_v} + \left\|c_{yy}(\hat{y}, u_i)\hat{\tilde{v}}\right\|_{L^\infty}\delta^y_i,$$

which can be done by scaling the result of equation (5.53) by $\left\|c_{yy}(\hat{y}, u_i)\hat{\tilde{v}}\right\|_{L^\infty}\delta^y_i$.

Remark 5.4.4. For scalar equations where the spatial differential operator is not self-adjoint, or systems of reaction-diffusion equations, the error equation for e_v differs from the error equation for the adjoint, and has to be solved additionally. As before, this can be done on a coarser mesh to keep the computational overhead small.

Additionally, we can evaluate $\overline{e_z}$ by equation (5.10), using $\overline{e_v}, \delta^y_i, \delta^\lambda_i, \delta^v$ as well as $\overline{e_\lambda}$. The latter is computed by scaling $\|e\|$ (from the solution of equation (5.53)) by

$$\left\|-J_{yy}(\hat{y}, u_i) - c_{yy}(\hat{y}, u_i)^\star\tilde{\lambda}\right\|_{L^\infty}\delta^y_i,$$

a quantity which can be cheaply computed during solution of the adjoint equation. With this we can estimate

$$\overline{e_{\mathrm{mv}}} = \|c_u(\hat{y}, u_i)^\star e\|\left(\left\|\overline{e_z}\right\| + \left\|c_{yy}(\hat{y}, u_i)^\star\tilde{w}\right\|_{L^\infty}\delta^y_i\right). \tag{5.54}$$

Remark 5.4.5. While theoretically this allows to estimate the overall error in the Hessian-vector products, in practice the error is over-estimated significantly. For determination of the quantization tolerances this decreases the performance of the compression, but has no influence on the convergence of the optimization. To algorithmically assert the condition on the error in the residual, $\|\tilde{r} - r\| \leq (1-\zeta_1-\zeta_2)\rho\left\|\tilde{j}'(u)\right\|$ these bounds are not sharp enough.

Before starting the next Newton iteration, new values for δ^y_{i+1} and δ^λ_{i+1} have to be provided in line 19. The state quantization tolerance is computed by using the minimum of the values given by equations (5.51), (5.43). In the latter,

$$\mu_i = \left\|c_u(\hat{y}, u_i)^\star e\right\|\left\|-J_{yy}(\hat{y}, u_i) - c_{yy}(\hat{y}, u_i)^\star\tilde{\lambda}\right\|_{L^\infty}, \tag{5.55}$$

where—as a heuristic—the value of $\tilde{\lambda}$ from the current iteration is used as an approximation for the next iteration. For the adjoint, δ_{i+1}^{λ} is determined by equation (5.52).

Remark 5.4.6. The computed error bounds are very coarse, leading to smaller-than-necessary quantization tolerances. Whenever problem-dependent information allows better estimates, the performance of the lossy compression algorithm will increase. However, in practice the quantization error will be oscillatory in almost all cases, so the true error will be significantly smaller than the worst-case estimates, even if the error equations would be solved with high accuracy.

5.5. Discussion

For fixed discretizations, all optimization methods discussed in this chapter ultimately require $\|e_{j'}\| \to 0$ for convergence, i.e. the trajectories have to be stored lossless. In the final optimization steps this results in the same storage *space* demand as the algorithms without compression. However, during the course of optimization, the reduction of required memory *bandwidth* is significant. Transferring less data to storage media may result in an overall decrease in runtime, when memory access is expensive, e.g. when having to use tape drives.

Typically, one is interested in convergence to the continuous solution instead of convergence to the solution of the discretized problem. In order to achieve this, the discretization errors in the reduced gradient have to be controlled via a-posteriori error estimates and mesh refinement, exemplarily we refer to [91, 108, 140, 145, 146]. Adaptivity for optimal control problems is an active field of research and beyond the scope of this thesis. Obviously, such adaptivity influences the trajectory compression: higher discretization accuracy in the later iterations of the optimization methods allow for storage reduction despite smaller quantization tolerances due to finer grids. A thorough analysis of this influence remains as future work.

When storage space is limited the conditions on the accuracy, e.g. Theorem 5.2.1, can be used as a stopping criterion—when these conditions cannot be fulfilled, no meaningful progress can be achieved anymore. In order not to stop the optimization prematurely, it is important to derive error bounds as sharp as possible. Thus the worst-case estimates presented in this chapter have to be refined. To achieve this, error control in norms other than L^∞ is desirable, particularly as numerical experience shows that typically the quantization error is rather oscillatory.

6. Numerical Results

In this chapter, a variety of numerical results are presented, illustrating the compression and error control techniques. We start with finite element interpolation of some test functions in Section 6.1, showing the validity of the a-priori estimates of Section 4.2. Additionally we compare our approach to fpzip (see Section 3.1.1). In Section 6.2 we present results for the example of boundary control for the linear heat equation, using the steepest-descent method. For optimal control of the semilinear Kolmogorov equation, the performance of steepest-descent and Newton-CG methods using lossy compression is discussed in Section 6.3. For the Newton-CG method with adaptive quantization we compare restart and residual replacement strategies during the inexact CG method. To conclude this chapter, in Section 6.4 we study in more detail optimal control of the monodomain equations introduced in Example 2.1.2. Results are shown using Newton-CG and BFGS-quasi-Newton methods.

All numerical examples were implemented using the C++ finite element toolbox Kaskade 7 [40]. Computation times were measured running the examples on a Dual-Core AMD Opteron 8220 CPU with 2.8 GHz, without using parallelization.

The results of Sections 6.1, 6.2, and 6.3.1 are mainly published in [141]. Parts of the results in the remaining Sections can be found in [37, 38, 39].

6.1. Auxiliary Test Functions

To demonstrate the effectivity of the lossy compression in a simple setting, we consider the same two functions as Schröder-Pander et al. [110] (see also Section 3.1.2, page 18),

$$f_1(x) = \sin(12(x_0 - 0.5)(x_1 - 0.5))$$
$$f_2(x) = \begin{cases} \sin(x_0)\cos(x_1), & x_0 > 0.5 \\ \cos(x_0)\sin(x_1), & \text{otherwise} \end{cases}$$

refine-ments	function	interpolation error	avg. bits/node	overall compression factor
7	f_1	$7.8 \cdot 10^{-4}$	2.56	22.8 (24.3)
	f_3	$1.69 \cdot 10^{-2}$	2.81	22.1 (28.1)
	f_4	$1.5 \cdot 10^{-5}$	1.55	42.3 (22.8)
8	f_1	$1.9 \cdot 10^{-4}$	2.56	24.9 (27.7)
	f_3	$4.2 \cdot 10^{-3}$	2.87	22.8 (29.7)
	f_4	$3.8 \cdot 10^{-6}$	1.49	48.3 (24.5)
9	f_1	$4.9 \cdot 10^{-5}$	2.56	26 (27.9)
	f_3	$1.1 \cdot 10^{-3}$	2.87	23.9 (31.2)
	f_4	$9.5 \cdot 10^{-7}$	1.49	49.1 (25.2)

Table 6.1.: L^∞-interpolation errors and compression factors for the different test functions $f_i(x)$, $i = 1, 3, 4$. The average bits/node are counted after quantization, based on the actual entropy of the data, the overall compression factor contains some overhead like interval bounds, and benefits from entropy coding. The numbers in brackets in the last column are the compression factors for the quantize-then-predict approach, compare Remark 4.1.3.

as well as two additional functions with different curvatures,

$$f_3(x) = \sin(50(x_0 - 0.5)(x_1 - 0.5))$$
$$f_4(x) = \frac{1}{2}(x_0^2 + x_1^2).$$

In all cases we take $x \in [0, 1]^2$. As the functions do not depend on time, only spatial prediction and lossy encoding of the prediction errors is performed. All functions are interpolated with linear finite elements on different grids. The grids are generated by uniform red refinement from an initial coarse mesh with 2 elements, resulting in 32 768 cells and 16 641 nodes on the finest level for seven refinements, 131 072 cells/66 049 vertices for eight refinement steps, and 524 288 elements/263 169 nodes after nine refinements.

This setting allows to compare the a-priori estimates from Section 4.2 with the numerical results, except for f_2, which is discontinuous. The approximate L^∞-interpolation errors are shown in Table 6.1 together with the compression factors for a quantization error tolerance of the same magnitude. One can notice a rather good agreement with the a-priori estimates. For function f_4, which has a very slight curvature, the linear interpolation used as a predictor performs expectedly good, leading to higher compression factors.

Compression factors for f_1 and f_2 using different tolerances δ are shown in Tables 6.2 and 6.3.

refinements	δ	compression factor	saved storage
7	10^{-6}	3.0	66.7%
	10^{-5}	5.2	80.8%
	10^{-4}	9.7	89.7%
9	10^{-6}	7.8	87.2%
	10^{-5}	13.6	92.6%
	10^{-4}	30.1	96.7%

Table 6.2.: Errors and compression factors for test function f_1

refinements	δ	compression factor	saved storage
7	10^{-6}	7.0	85.7%
	10^{-5}	13.9	92.8%
	10^{-4}	34.1	97.1%
9	10^{-6}	22.6	95.6%
	10^{-5}	83.9	98.8%
	10^{-4}	213.3	99.5%

Table 6.3.: Errors and compression factors for test function f_2

As expected, for a fixed error bound δ the compression factor increases with the number of grid levels, as the prediction error gets smaller each level, and fewer bits need to be stored. In Figure 6.1, the reconstructed functions are shown for 7 refinement levels and $\delta = 10^{-2}$.

For further comparison, we use fpzip based on [82], a publicly available lossy floating-point compression algorithm to store the double precision nodal values. As fpzip is explicitly designed for structured, cartesian grid data, it is a good benchmark for this special case. The results can be found in Table 6.4. Note that fpzip does not keep a specific error bound, but can only be set to use a certain amount of bits for storing a floating point value. The settings were 16 bit/value, leading to $\delta = 3.1 \cdot 10^{-2}$, and 32 bit/value, leading to $\delta = 4.8 \cdot 10^{-7}$. As $\max_{x\in\Omega} |f_i(x)| = 1 \forall i$, relative and absolute L^∞-errors coincide and can be compared for all test functions. For f_4, and 32 bit/value, fpzip reconstructs the values exactly, as in 2D the Lorenzo predictor used by fpzip is exact for functions $f(x) = g(x_0) + h(x_1)$. For the larger error tolerance, our algorithm is clearly better. For $\delta = 4.8 \cdot 10^{-7}$, fpzip performs better for some settings. However, the interpolation error in that case is larger than $\delta = 4.8 \cdot 10^{-7}$.

Adaptive mesh refinement can be seen as a means of data compression itself. To

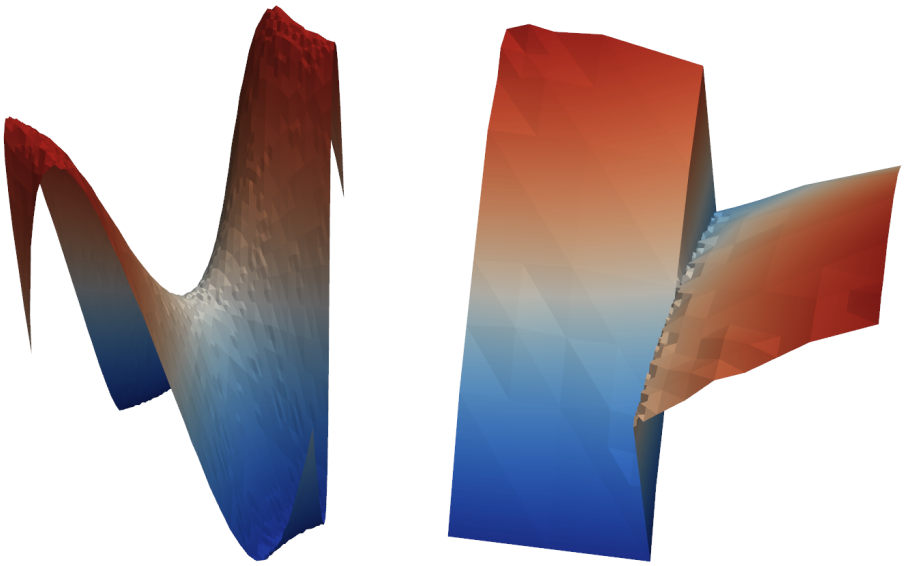

Figure 6.1.: Reconstructed functions f_1 (left) and f_2 (right) at quantization error 10^{-2} and compression factors of 45 (f_1) and 127 (f_2) for 7 uniform refinements.

study the effect on the lossy storage approach, we consider adaptive interpolation of f_1. To reach an L^∞-interpolation error of $1.9 \cdot 10^{-4}$, eight uniform refinements were needed, leading to 66 049 vertices. Using an adaptive grid, the degrees of freedom could be reduced to 42 893, a compression factor of merely 1.5. Lossy storage of the nodal values led to overall compression factors of 24.9 in the uniform case, and 13.8 for the adaptive interpolation.

For f_1, adaptive interpolation had a rather small impact. We now consider interpolation by linear finite elements of the 2D Gaussian function

$$f(x) = \exp\left(-\frac{(x_0 - 0.5)^2 + (x_1 - 0.5)^2}{2\sigma^2}\right),$$

with $\sigma = 0.025$ on $\Omega = [0, 1]^2$. As $f(x)$ exhibits a highly local peak, adaptive mesh refinement leads to a drastic reduction of degrees of freedom, and thus of the values which need to be stored. For reaching an L^∞-interpolation error of approximately 0.0015, a uniformly refined mesh consists of 263 169 vertices, whereas the adaptive grid just needs 4237 nodes. This amounts to a compression factor of approximately 62. The compression factor of the lossy storage approach for a quantization error bound of 0.0015 reduces from 54 on the uniform mesh to 12 for the adaptive grid.

δ	function	7 refinements		9 refinements	
		our algorithm	fpzip	our algorithm	fpzip
$3.1 \cdot 10^{-2}$	f_1	57.0	33.6	498.3	58.1
	f_2	214.4	81.2	1178.8	242.8
	f_3	32.1	28.2	64.0	46.5
	f_4	374.0	47.7	4896.2	103.2
$4.8 \cdot 10^{-7}$	f_1	2.6	4.3	6.7	6.3
	f_2	5.8	7.9	17.7	17.2
	f_3	1.9	3.7	3.8	5.0
	f_4	28.5	136.4*	31.6	587.8*

Table 6.4.: Compression factors for our algorithm and fpzip [82] for the different test functions on a mesh with 7 and 9 levels of refinement. For the entries marked with *, the predictor of fpzip is exact, such that no prediction errors need to be stored.

The latter still amounts to 91.7% saved storage space, the combination of adaptive mesh refinement and lossy storage saves 99.9% space and memory bandwidth.

6.2. Linear Heat Equation

We consider the simple model problem

$$\min \frac{1}{2} \|y - y_d\|^2_{L^2(\Omega\times(0,T))} + \frac{\alpha}{2} \|u\|^2_{L^2(\partial\Omega\times(0,T))} \tag{6.1}$$

subject to

$$\begin{aligned} y_t - \Delta y &= f \quad \text{in } \Omega \times (0,T) \\ \partial_\nu y + y &= u \quad \text{on } \partial\Omega \times (0,T) \\ y(\cdot,0) &= 0 \quad \text{in } \Omega, \end{aligned} \tag{6.2}$$

see also Examples 2.1.1, 5.2.2.

The above optimal control problem is solved with the aid of the compression algorithm in its basic form, i.e. spatial prediction by linear interpolation between the grid levels, and constant prediction in time.

The given data are

$$\begin{aligned} \Omega = (0,1)^2, \quad T = 1, \quad \alpha = 10^{-5}, \\ y_d(x,t) = t((x_0-1)^2 + (x_1-1)^2), \quad f(x,t) = (x_0-1)^2 + (x_1-1)^2 - 4t. \end{aligned}$$

We apply an implicit Euler method for time stepping, with a fixed step size $dt = 0.05$, and a spatial discretization with linear finite elements on a grid with 32 768 cells on the finest level, generated by 7 uniform refinement steps from the coarse grid. For minimization, a simple steepest-descent algorithm with an Armijo step size rule is used [55, 98]. The discretization errors in the reduced gradient and control are estimated by using a solution of the problem on a fine mesh as reference.

For comparison, we stop the optimization after a certain number of iterations, and use a fixed error bound for the quantization error, see Figure 6.2 for the results. The qualitative behavior predicted by the a-priori estimates is clearly visible. In contrast to these estimates, which were derived with regard to the interpolation error of the state solution, in Figure 6.2 the discretization error of the reduced gradient and control are depicted. Also, the impact of the simple temporal prediction can be seen.

We apply the techniques for error control described in Section 5.2 to the above optimal control problem, summarized in Algorithm 6.

Algorithm 6 Inexact steepest-descent with adaptive quantization tolerance

1: fix initial δ_1^y (provided by user)
2: **for** $i = 1, \dots$ **do**
3: solve the state equations (6.2), encode y using δ_i^y
4: compute inexact gradient $\widetilde{\nabla j}(u_i)$ as given in Example 5.2.2, using decoded $\hat{y}$
5: check optimality conditions, if optimal: end
6: update the control $u_{i+1} = u_i + \alpha_i \widetilde{\nabla j}(u_i)$ using the Armijo rule to determine the step size α_i
7: determine new quantization tolerance δ_{i+1}^y using Algorithm 3
8: **end for**

Figure 6.3 shows a comparison for the evolution of the gradient norm during the optimization with and without lossy storage of the state values, as well as the corresponding compression factors. Clearly, lossy compression has no influence on the convergence behavior, as long as the error is controlled. With an optimized implementation of the lossy compression algorithm, one iteration of the optimization algorithm (state solve, adjoint solve, gradient computation and control update) took $192s$ on average, with $1.3s$ for compressed writing and reading of the state values. One optimization iteration without compression took $190.5s$ on average, with $0.03s$ for writing and reading. The overall runtime increase for lossy compression thus is approximately 1%. Note that despite the relatively small size all state values are written to disk, and are not kept in RAM.

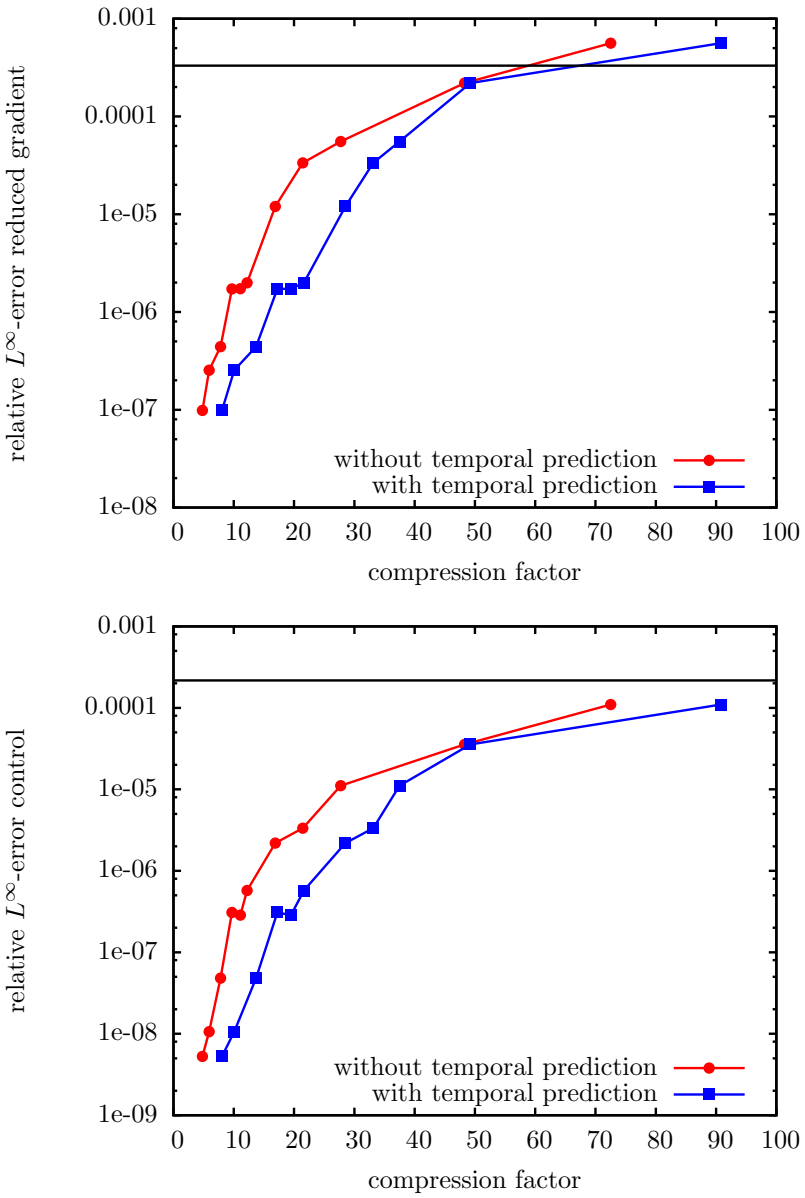

Figure 6.2.: Relative error vs. compression factor for gradient (top) and control (bottom) after 100 iterations for the linear boundary control problem (6.1), (6.2), for different tolerances for the quantization error. The horizontal line shows the approximated discretization error. The temporal predictor yields a noticeable increase of the compression factor at virtually no computational cost.

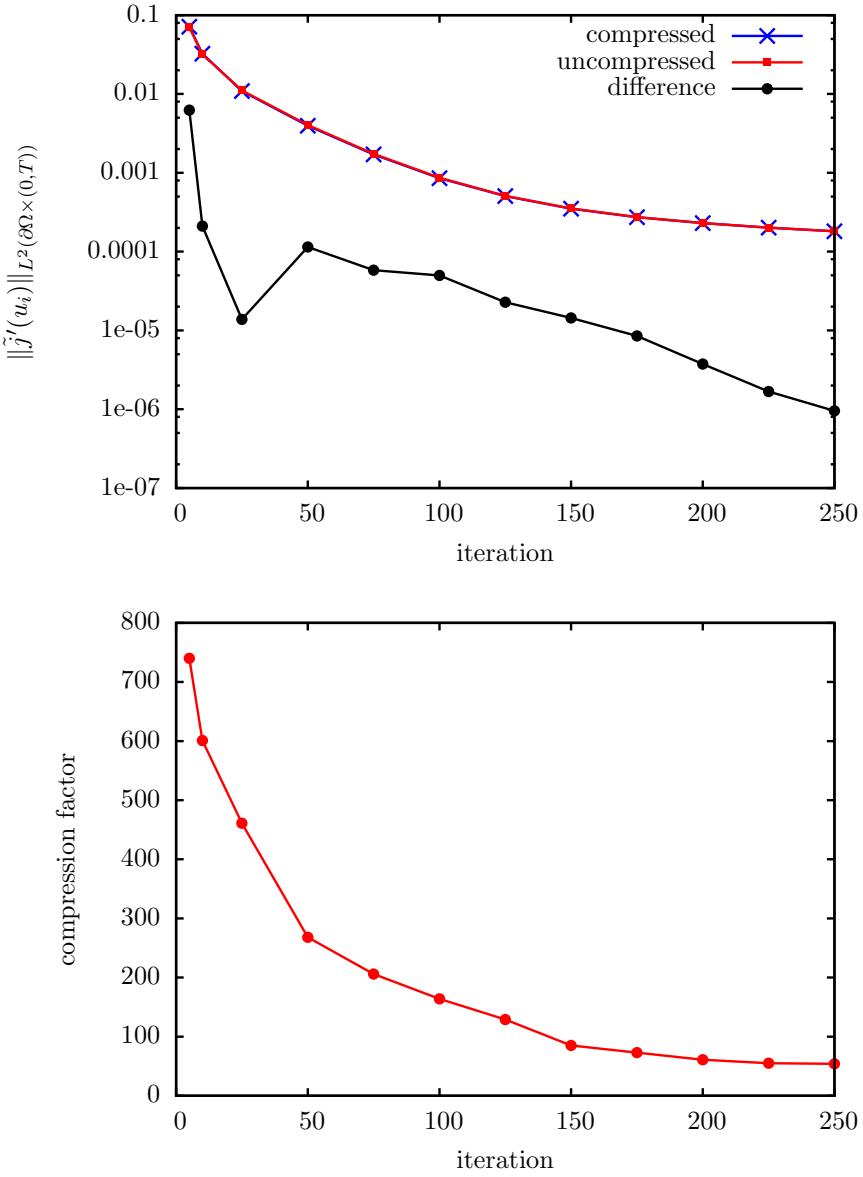

Figure 6.3.: Progress of the optimization algorithm (boundary control of linear heat equation (6.1), (6.2)) with and without using compression (top), and the corresponding adaptively chosen compression factors (bottom).

6.3. Kolmogorov Equation

As a second example we consider an optimization problem governed by the semi-linear Kolmogorov equation. The control is only varying in time and is constant in space on each of five control domains. The optimal control problem is given by

$$\min \frac{1}{2} \|y - y_d\|^2_{L^2(\Omega\times(0,T))} + \frac{\alpha}{2} \|u\|^2_{L^2(0,T;\mathbb{R}^5)} \tag{6.3}$$

subject to

$$\begin{aligned} y_t - \sigma^2 \Delta y &= f(y) + \sum_{i=1}^{5} \chi_{\Omega_{c_i}} u_i(t) && \text{in } \Omega \times (0,T) \\ \partial_\nu y &= 0 && \text{on } \partial\Omega \times (0,T) \\ y(\cdot,0) &= y_0 && \text{in } \Omega \end{aligned} \tag{6.4}$$

with $f(y) = y(y-a)(b-y)$ and

$$\Omega = (0,1)\times(0,1), \quad T = 10, \quad a = 0.1, \quad b = 1, \quad \sigma = 0.15,$$
$$y_d(x,t) = \frac{1}{1 + e^{\left((\|x\| - \frac{1}{3})\cdot\frac{1}{\sigma\sqrt{2}} - t\right)}}, \quad y_0(x) = y_d(x,0), \quad \alpha = 10^{-5}.$$

The control domain is given by $\Omega_c = \bigcup_{i=1}^{5} \Omega_{c_i}$ with

$$\begin{aligned} \Omega_{c_1} &= [0.125, 0.25] \times [0.75, 0.875], & \Omega_{c_2} &= [0.75, 0.875]^2, \\ \Omega_{c_3} &= [0.4375, 0.5625]^2, & \Omega_{c_4} &= [0.125, 0.25]^2, \\ \Omega_{c_5} &= [0.75, 0.875] \times [0.125, 0.25]. \end{aligned}$$

This problem can be seen as a mock-up of cardiac defibrillation (see Section 6.4). Without control, the solution to the state equation (6.4) is a wavefront traveling through the domain until $y \approx 1$ everywhere; the control problem aims at a certain speed and shape of that wavefront.

6.3.1. Steepest-Descent

As before, the optimal control problem is solved by a steepest-descent method, with the PDEs being discretized in time by a linearly implicit, extrapolated Euler method with fixed step size $dt = 0.1$, and linear finite elements in space. The grid hierarchy consists of 8 levels, with 32 768 cells and 16 641 nodes on the finest level. Again, we obtain good compression rates with relative errors in the reduced

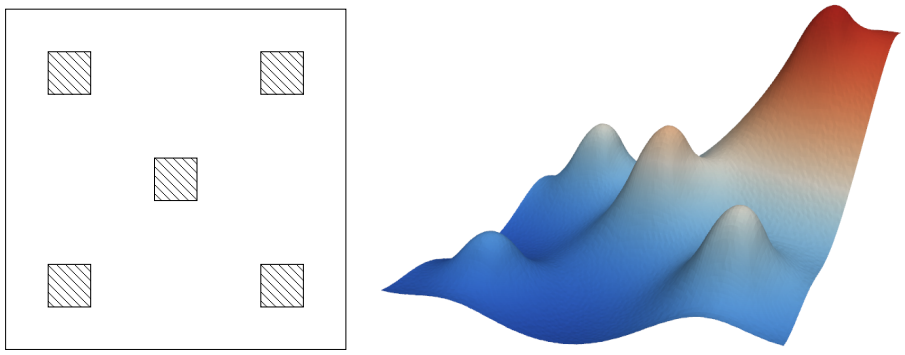

Figure 6.4.: Domain Ω (left) and reconstructed state (right) at $t = 0.8$ after 50 steepest-descent iterations ($\delta = 5 \cdot 10^{-4}$, compression factor 56, 1.14 bits/double).

gradient and computed control well below the discretization error (see Figure 6.5), approximated as in Example 6.2. The optimization algorithm is stopped after 50 iterations to be able to compare results for different compression rates. Note that here the quantization error tolerance is fixed, and not chosen according to the current size of the gradient norm in each iteration.

6.3.2. Newton-CG

Here we present results for the Newton-CG method using adaptive quantization. Following the theory presented in Section 5.1, the error in the adjoint equation e_λ fulfills

$$-[e_\lambda]_t - \sigma^2 \Delta e_\lambda = f_y(\hat{y})e_\lambda - f_{yy}(\hat{y})\varepsilon_y e_\lambda - \varepsilon_y + f_{yy}(\hat{y})\varepsilon_y \tilde{\lambda}.$$

The error e_v in the linearized-state v due to inexact storage of y satisfies the equation

$$[e_v]_t - \sigma^2 \Delta e_v = f_y(\hat{y})e_v - f_{yy}(\hat{y})\varepsilon_y e_v + f_{yy}(\hat{y})\varepsilon_y \tilde{v}.$$

In the adjoint-for-Hessian, the error e_w is determined by

$$-[e_w]_t - \sigma^2 \Delta e_w = f_y(\hat{y})e_w - f_{yy}(\hat{y})\varepsilon_y e_w + f_{yy}(\hat{y})\varepsilon_y \tilde{w} + e_z$$

where e_z is given by

$$\begin{aligned} e_z = e_v + \varepsilon_v &- \langle \big(f_{yy}(\hat{y}) - f_{yyy}(\hat{y})\varepsilon_y\big)(e_v + \varepsilon_v), \hat{\tilde{\lambda}} \rangle - \langle f_{yyy}(\hat{y})\varepsilon_y \hat{\tilde{v}}, \hat{\tilde{\lambda}} \rangle \\ &- \langle \big(f_{yy}(\hat{y}) - f_{yyy}(\hat{y})\varepsilon_y\big)(e_v + \varepsilon_v), e_\lambda + \varepsilon_\lambda \rangle \\ &- \langle \big(f_{yy}(\hat{y}) + f_{yyy}(\hat{y})\varepsilon_y\big)\hat{\tilde{v}}, e_\lambda + \varepsilon_\lambda \rangle. \end{aligned}$$

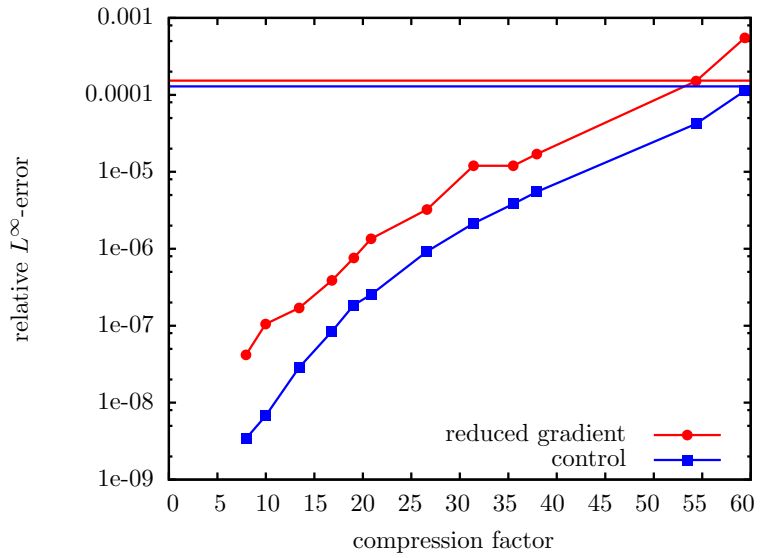

Figure 6.5.: Relative error vs. compression factor for reduced gradient and control after 50 steepest descent iterations for the Kolmogorov problem (6.3), (6.4). The horizontal line shows the approximated discretization errors for the reduced gradient (solid) and the control (dashed).

We use the algorithm sketched in Section 5.4.3, without any additional a-priori information, on a fixed finite element grid with 32 768 elements and 16 641 vertices. Figure 6.6 shows the progress of the optimization. No significant difference between compressed and uncompressed storage is visible. The corresponding quantization tolerances for state and adjoint are shown in Figure 6.7, where $s^\lambda = 1$ for the determination of δ^λ by equation (5.52). Before iteration 5, the state quantization tolerance δ^y is determined by equation (5.51), afterwards the tolerance given by equation (5.43) is smaller. We observe that, due to the worst-case error estimation, the quantization tolerance is reduced severely in the final Newton iterations, such that (nearly) no compression can be achieved in the last two iterations. A remedy for this issue is to use more problem-dependent information for the error estimation, to achieve tighter error bounds. To some extent the reduction of the compression factors is due to the fixed grid, as the requested accuracy for quantization is far below the discretization error.

For one solution of the state equation, 254.5s CPU time were needed (averaged over all iterations), whereas the compression required 6.5s. Solution of the adjoint equation required 267.6s on average, with additional 10.7s for compression. The error equations were solved on a mesh with 4225 vertices; error estimation required

38.6s CPU time per iteration. Overall, encoding/decoding incurred an overhead of 3.3%, with additionally less than 1% overhead for error estimation (the latter compared to the overall step computation time).

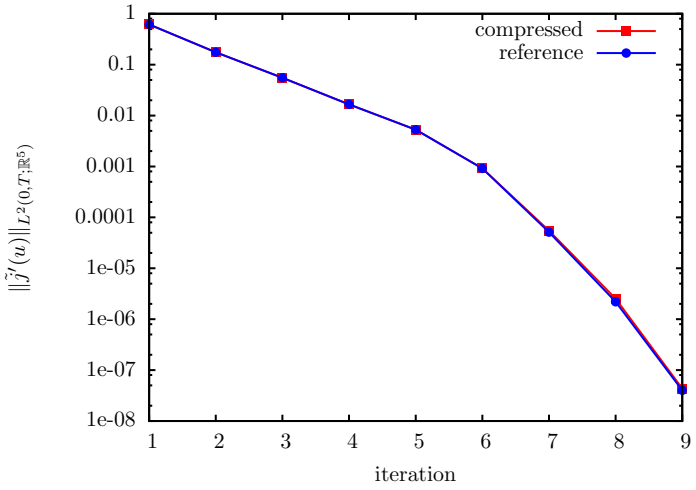

Figure 6.6.: L^2-norm of the reduced gradient with and without compression for the Kolmogorov example (6.3), (6.4) using Newton-CG. No significant difference is notable.

For triggering the residual replacement during the CG method, the threshold

$$\epsilon = \left((1 - \zeta_1 - \zeta_2)\rho_i \left\|\tilde{j}'(u_i)\right\|\right)^{1/2}$$

is used (see equation (5.49)). The safety factor s^v is set to one in the beginning, and multiplied by 0.1 on each re-computation of the residual. Figure 6.8 shows the behavior of the residual during the CG in Newton iteration 6. Compression factors ranging between 8.0 and 117.4 were achieved. Two residual replacements were triggered, in CG iterations 3 and 6. For the re-computation of the residual from the current iterate, the compression factors were 2.8 and 1.4, respectively. Overall, 142 CG iterations were needed during the 8 Newton iterations, with additional 12 residual replacement computations. At most three re-computations were necessary per Newton iteration.

With the restart strategy from [37], 9 Newton iterations with 189 CG iterations and 33 restarts were required for convergence. The behavior of both strategies is compared in Figure 6.9. With residual replacement, re-computations were required in CG iterations 3 and 4, whereas the restart approach leads to restarts in CG

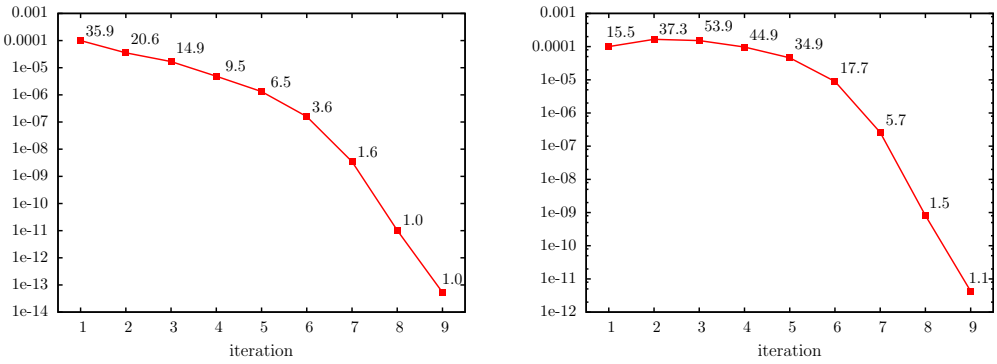

Figure 6.7.: Adaptively chosen quantization tolerances and corresponding compression factors for state (left) and adjoint (right) for the Kolmogorov example (6.3), (6.4) using Newton-CG.

iteration 2,3,4, and 6. Overall, using the restart strategy, more CG iterations are required. The convergence of CG is impeded by the old approach, as, due to the complete restart, the superlinear CG convergence is cut off. With the residual replacement this perturbation is minimal, as all other quantities except the residual itself are kept.

6.4. Monodomain Equations

As an example for reaction-diffusion systems, we consider an optimal control problem for the monodomain equations (see, e.g., [96, 76] and Example 2.1.2) on a simple 2D unit square domain $\Omega = (0,1)^2$. This system describing the electrical activity of the heart consists of a parabolic PDE for the transmembrane voltage v, coupled to pointwise ODEs for the gating variable w. As membrane model, we use the Rogers-McCulloch variant of the Fitzhugh-Nagumo model [104]. As in Example 2.1.2, the state system is given by

$$\begin{aligned} v_t &= \nabla \cdot \sigma \nabla v - I_{\text{ion}}(v,w) + I_{\text{e}} && \text{in } \Omega \times (0,T) \\ w_t &= G(v,w) && \text{in } \Omega \times (0,T), \end{aligned} \tag{6.5}$$

with

$$\begin{aligned} I_{\text{ion}}(v,w) &= gv\big(1 - \frac{v}{v_{th}}\big)\big(1 - \frac{v}{v_p}\big) + \eta_1 vw \\ G(v,w) &= \eta_2\big(\frac{v}{v_p} - \eta_3 w\big) \end{aligned}$$

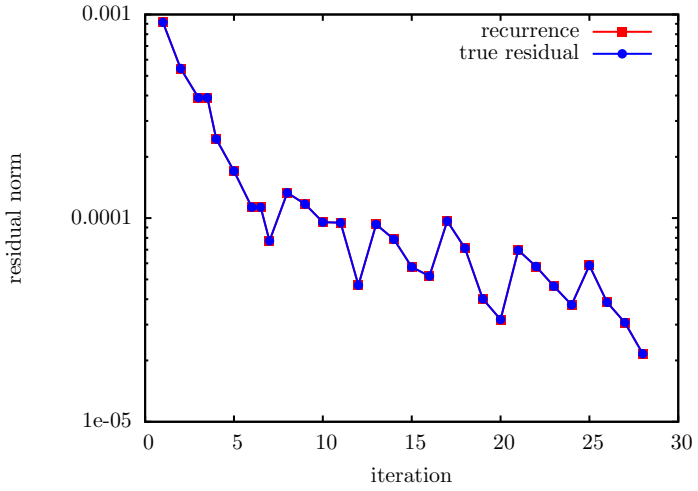

Figure 6.8.: Behavior of CG in Newton iteration 6 for the Kolmogorov example (6.3), (6.4). The *true residual* denotes the residual without quantization of the linearized state. In iterations 3 and 6 the residual was re-computed.

and homogeneous Neumann boundary conditions and initial values. In this 2D model $\sigma : \mathbb{R}^2 \to \mathbb{R}^{2\times 2}$ and $g, \eta_i, v_p, v_{th} \in \mathbb{R}_+$ are given parameters (see Table 6.5). For details, see [94].

σ_{il} $[\Omega^{-1}\text{cm}^{-1}]$	σ_{it} $[\Omega^{-1}\text{cm}^{-1}]$	g $[\text{mS/cm}^2]$	v_{th} [mV]	v_p [mV]	η_1 $[\text{mS/cm}^2]$	η_2	η_3
$3 \cdot 10^{-3}$	$3.1525 \cdot 10^{-4}$	1.5	13	100	4.4	0.012	1

Table 6.5.: Electrophysiological parameters (adapted from [35]).

Before turning to the actual 2D optimal control problem, a snapshot of a 3D simulation is shown in Figure 6.10. An excitation $I_e = 100$ for 1ms in a small ball of radius 1 around the coordinate $(-0.9, 1.7, 1.1)$ leads to a excitation wave traveling through the domain. This simulation was performed using heart geometry from [59]. In this snapshot, the mesh consists of 2 251 410 elements/403 192 vertices on 5 levels. Encoding of the 806 384 degrees of freedom took 4.9s, achieving a compression factor of 15.4 for a relative L^∞-error of order 10^{-4} in the transmembrane voltage v and 10^{-2} in the gating variable w.

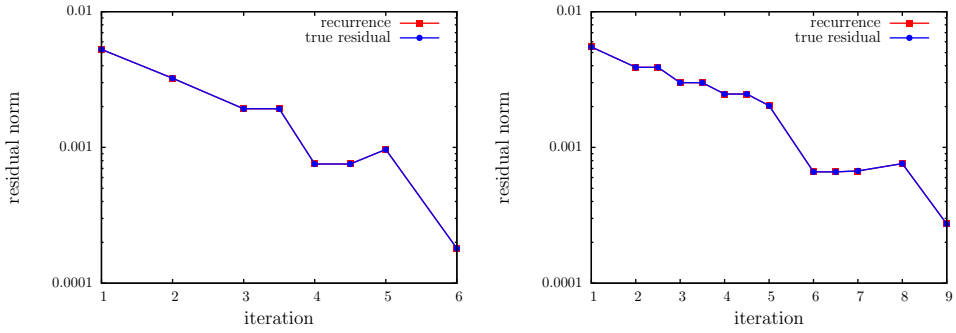

Figure 6.9.: Behavior of CG in Newton iteration 5 for the Kolmogorov example (6.3), (6.4). The *true residual* denotes the residual without quantization of the linearized state. Left: new residual replacement. Right: restart strategy from [37].

For the optimal control problem initial values

$$v(x,0) = \begin{cases} 101.0 & \text{in} \quad \Omega_{\text{exi}} \\ 0 & \text{otherwise} \end{cases}$$
$$w(x,0) = 0 \quad \text{in } \Omega.$$

are prescribed. Here, Ω_{exi} is a circle with midpoint $(0.5, 0.5)$ and radius 0.04. The external current stimulus $I_{\text{e}}(x,t) = \chi_{\Omega_c}(x)u(t)$, where the control u is spatially constant on the control domain $\Omega_c = [0.37, 0.4] \times [0.45, 0.55] \cup [0.6, 0.63] \times [0.45, 0.55]$.

The evolution of the transmembrane voltage for $u \equiv 0$ is depicted in Figure 6.11. For encoding of the computational grids we used an implementation the algorithm presented in [69, 136], which is available in JavaView [1], a toolkit for mathematical geometry processing and visualization. Corresponding compressed factors for state values and adaptive mesh are presented in Figure 6.12, both with and without delta-encoding. Using delta-encoding in time more than doubles the achieved overall compression factor for the state values. The bits/vertex for connectivity encoding are reduced to 66% compared to compressing each timestep separately, see also [38].

For the objective functional we choose

$$J(y,u) = \frac{1}{2} \|v\|^2_{L^2(\Omega_{\text{obs}} \times (0,T))} + \frac{\alpha}{2} \|u\|^2_{L^2(0,T)}, \tag{6.6}$$

i.e. we aim at damping out the excitation wave. We set

$$\Omega_{\text{obs}} = \Omega \setminus \big([0.35, 0.42] \times [0.43, 0.57] \cup [0.58, 0.65] \times [0.43, 0.57]\big),$$

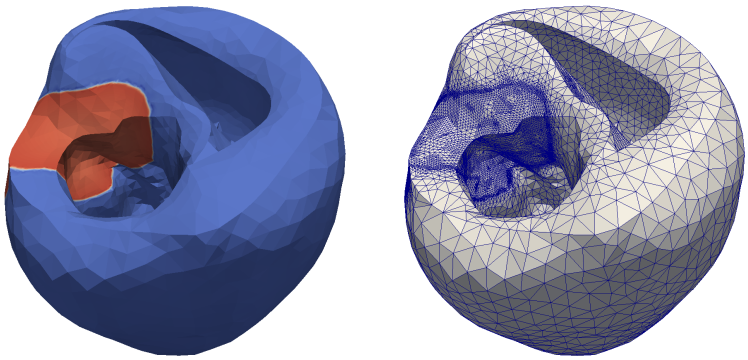

Figure 6.10.: Reconstructed 3D solution 15.3 ms after excitation (compression factor 15.4, relative L^∞-error 10^{-4}). Left: transmembrane voltage. Right: computational grid (403192 vertices).

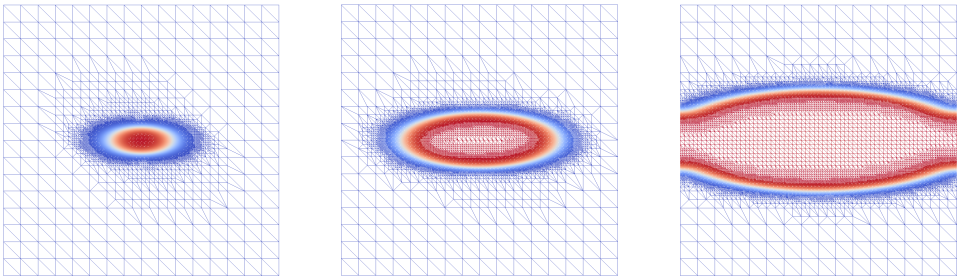

Figure 6.11.: Uncontrolled solution v at 1ms, 3ms and 6ms for the monodomain equations (6.5).

$T = 4$, and $\alpha = 3 \cdot 10^{-6}$.

The optimality condition $j'(u) = 0$ is given by

$$\alpha u + \int_{\Omega_c} p \, dx = 0 \quad \text{a.e. in } (0, T), \tag{6.7}$$

where p is defined through the adjoint equations

$$\begin{aligned} p_t &= -\nabla \cdot \sigma \nabla p + [I_{\text{ion}}]_v p + G_v q - v|_{\Omega_{\text{obs}}} && \text{in } \Omega \times (0, T) \\ q_t &= -[I_{\text{ion}}]_w p - G_w(v, w) q && \text{in } \Omega \times (0, T) \end{aligned} \tag{6.8}$$

with homogeneous terminal and Neumann boundary conditions. For application of the reduced Hessian to a vector δu the following linearized-state equations (6.9) and

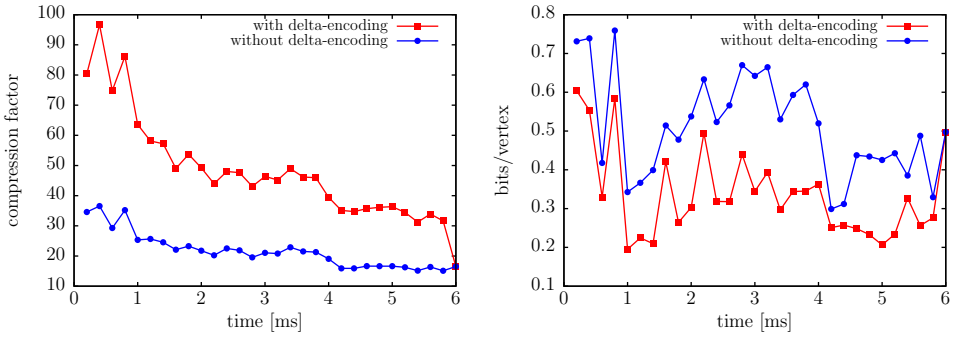

Figure 6.12.: Compression factor of the state values (uncontrolled solution) for $\delta = 10^{-2}$ (left) and bits/vertex for connectivity encoding (right), both with and without delta-encoding between timesteps.

adjoint-for-Hessian equations (6.10) have to be solved for $\delta p, \delta q$:

$$\begin{aligned} \delta v_t &= \nabla \cdot (\sigma \nabla \delta v) - \left([I_{\text{ion}}]_v \, \delta v + [I_{\text{ion}}]_w \, \delta w\right) + \chi_{\Omega_c} \delta u \quad \text{in } \Omega \times (0,T) \\ \delta w_t &= \frac{\eta_2}{v_p} \delta v - \eta_2 \eta_3 \delta w \qquad\qquad\qquad\qquad\qquad\quad \text{in } \Omega \times (0,T) \end{aligned} \tag{6.9}$$

with homogeneous initial and Neumann boundary conditions, and

$$\begin{aligned} \delta p_t &= -\nabla \cdot \sigma \nabla \delta p + [I_{\text{ion}}]_v \delta p + \frac{\eta_2}{v_p} \delta q + z_1 \quad \text{in } \Omega \times (0,T) \\ \delta q_t &= -[I_{\text{ion}}]_w \delta p + \eta_2 \eta_3 \delta q + z_2 \qquad\qquad\quad \text{in } \Omega \times (0,T), \end{aligned} \tag{6.10}$$

also with homogeneous initial and Neumann boundary conditions. Here z_1, and z_2 are given by $z_1 = \delta v|_{\Omega_{\text{obs}}} - [I_{\text{ion}}]_{vv} \, p \, \delta v - \eta_1 \, \delta w \, p$ and $z_2 = -\eta_1 \, \delta v \, p$. A detailed discussion of this optimal control problem and the derivation of the optimality system can be found in [93, 37] and the references therein.

6.4.1. Error Estimation

For ease of notation, let us consider the abstract semi-linear problem

$$\begin{aligned} \frac{\partial y_1}{\partial t} - \nabla \cdot \sigma \nabla y_1 &= g_1(y) \quad \text{in } \Omega \times (0,T) \\ \frac{\partial y_2}{\partial t} &= g_2(y) \quad \text{in } \Omega \times (0,T), \end{aligned} \tag{6.11}$$

together with homogeneous initial- and Neumann boundary conditions.

We apply our findings of Section 5.1 to the defibrillation problem. For the error in the reduced gradient, the error equations are given by

$$\begin{aligned}\frac{\partial e_p}{\partial t} - \nabla \cdot \sigma \nabla e_p &= -[\hat{I}_{\text{ion}}]_v e_p - G_v e_q + [\hat{I}_{\text{ion}}]_{vv}\varepsilon_v e_p + [\hat{I}_{\text{ion}}]_{vw}\varepsilon_w e_p \\ &\quad + \varepsilon_v|_{\Omega_{\text{obs}}} - [\hat{I}_{\text{ion}}]_{vv}\varepsilon_v \tilde{p} - [\hat{I}_{\text{ion}}]_{vw}\varepsilon_w \tilde{p} \qquad \text{in } \Omega \times (0,T), \\ \frac{\partial e_q}{\partial t} &= [\hat{I}_{\text{ion}}]_w e_p + G_w e_q + [\hat{I}_{\text{ion}}]_{wv}\varepsilon_v \tilde{p} + [\hat{I}_{\text{ion}}]_{wv}\varepsilon_v e_p \qquad \text{in } \Omega \times (0,T),\end{aligned} \tag{6.12}$$

where $\hat{I}_{\text{ion}} = I_{\text{ion}}(\hat{v}, \hat{w})$.

For application of comparison principles, the backward-in-time system was transformed to a forward system by the standard change of variables $t = T - \tau$.

Using the abbreviations

$$\begin{aligned}a(x,t) &= -[\hat{I}_{\text{ion}}]_v(x,t) + [\hat{I}_{\text{ion}}]_{vv}\varepsilon_v + [\hat{I}_{\text{ion}}]_{vw}\varepsilon_w \\ b(x,t) &= -\eta_2/v_p = \text{const} < 0 \\ c(x,t) &= \chi_{\Omega_{\text{obs}}}(x)\varepsilon_v(x,t) - [\hat{I}_{\text{ion}}]_{vv}(x,t)\varepsilon_v(x,t)\tilde{p}(x,t) - [\hat{I}_{\text{ion}}]_{vw}\varepsilon_w(x,t)\tilde{p}(x,t) \\ d(x,t) &= [\hat{I}_{\text{ion}}]_w + [\hat{I}_{\text{ion}}]_{wv}\varepsilon_v \\ e(x,t) &= -\eta_2\eta_3 = \text{const} < 0 \\ f(x,t) &= [\hat{I}_{\text{ion}}]_{wv}\varepsilon_v(x,t)\tilde{p}(x,t),\end{aligned}$$

we can formulate the reaction function $g(y)$ in the abstract system (6.11) as

$$g_1(y_1, y_2) = ay_1 + by_2 + c, \quad g_2(y_1, y_2) = dy_1 + ey_2 + f. \tag{6.13}$$

In our case, the transmembrane potential $\hat{v}$ and thus $d(x,t)$ may be negative. Moreover, we have

$$\frac{\partial g_1}{\partial y_2} = b < 0\,.$$

Hence g is not quasi-monotone non-decreasing. With the aid of Theorem 5.1.8 and a sub-solution $\underline{y}$, we construct a super-reaction function:

$$\begin{aligned}\overline{g}_1(y) &= \sup_{\{z \mid \underline{y} \le z \le y,\ z_1 = y_1\}} g_1(z) = ay_1 + c + \sup_{\{z_2 \mid \underline{y}_2 \le z_2 \le y_2\}} bz_2 \overset{b<0}{=} ay_1 + c + b\underline{y}_2 \\ \overline{g}_2(y) &= \sup_{\{z \mid \underline{y} \le z \le y,\ z_2 = y_2\}} g_2(z) = ey_2 + f + \sup_{\{z_1 \mid \underline{y}_1 \le z_1 \le y_1\}} dz_1.\end{aligned} \tag{6.14}$$

We need to derive a sub-solution $\underline{y}$ to problem (6.11) with right-hand-side (6.13). For simplicity, $\underline{y}$ should be constant in time and space.

Lemma 6.4.1. *Let* $\varepsilon_v^{\max}, \varepsilon_w^{\max} > 0$ *and*

$$\begin{aligned}\underline{c}(x,t) &= -\left(\varepsilon_v^{\max} + \left\|[\hat{I}_{\text{ion}}]_{vv}\tilde{p}\right\|_{L^\infty(\Omega\times(0,T))}\varepsilon_v^{\max} + \eta_1 \left\|\tilde{p}\right\|_{L^\infty(\Omega\times(0,T))}\varepsilon_w^{\max}\right),\\ \underline{f}(x,t) &= -\eta_1 \left\|\tilde{p}\right\|_{L^\infty(\Omega\times(0,T))}\varepsilon_v^{\max}.\end{aligned}$$

Then the constant function $\underline{y} = \left(0, \min\{-\underline{c}/b,\ -\underline{f}/e\}\right)^{\mathsf{T}}$ *is a sub-solution to* (6.11) *with right-hand-side* (6.13).

Proof. As $\underline{y}$ is constant in x, and t, the derivatives vanish. It remains to show that $0 \le a\underline{y}_1 + b\underline{y}_2 + c$ and $0 \le d\underline{y}_1 + e\underline{y}_2 + f$. As $b, e < 0$, by definition $\underline{c}, \underline{f} \le 0$, and thus $\underline{y}_2 \le 0$. We have:

$$\begin{aligned}a\underline{y}_1 + b\underline{y}_2 + c &= b\min\{-\underline{c}/b,\ -\underline{f}/e\} + c \ge b\min\{-\underline{c}/b,\ -\underline{f}/e\} + \underline{c}\\ &\begin{cases} = -\underline{c} + \underline{c} = 0, & \text{if } -\underline{c}/b \le -\underline{f}/e\\ = -b(\underline{f}/e) + \underline{c} \ge -b(\underline{c}/b) + \underline{c} \ge 0, & \text{if } -\underline{c}/b > -\underline{f}/e,\end{cases}\end{aligned}$$

and

$$\begin{aligned}d\underline{y}_1 + e\underline{y}_2 + f &= e\min\{-\underline{c}/b,\ -\underline{f}/e\} + f \ge e\min\{-\underline{c}/b,\ -\underline{f}/e\} + \underline{f}\\ &\begin{cases} = -e(\underline{c}/b) + \underline{f} \ge -e(\underline{f}/e) + \underline{f} \ge 0, & \text{if } -\underline{c}/b \le -\underline{f}/e\\ = -\underline{f} + \underline{f} = 0, & \text{if } -\underline{c}/b > -\underline{f}/e.\end{cases}\end{aligned}$$

□

Remark 6.4.2. Note that in Lemma 6.4.1 $\underline{c}, \underline{f}$ are defined using $\tilde{p}$ instead of $\hat{\tilde{p}}$. In the implementation, the occurring norms can be evaluated during the adjoint solve, such that these quantities are available.

With this sub-solution at hand, we note for the super-reaction function $\overline{g}$, that

$$\sup_{\{z_1 | 0 \le z_1 \le y_1\}} dz_1 \le \sup_{\{z_1 | 0 \le z_1 \le y_1\}} |d| z_1 = |d| y_1.$$

Thus we set

$$\begin{aligned}\overline{g}_1(y) &= ay_1 + b\underline{y}_2 + \overline{c}\\ \overline{g}_2(y) &= |d|y_1 + ey_2 + \overline{f},\end{aligned} \tag{6.15}$$

where we modify $\overline{g}$ given by (6.14) further, replacing c and f by upper bounds

$$\begin{aligned}\overline{c}(x,t) &= \chi_{\Omega_{\text{obs}}}(x)\varepsilon_v^{\max} + |[\hat{I}_{\text{ion}}]_{vv}\tilde{p}(x,t)|\varepsilon_v^{\max} + |\eta_1\tilde{p}(x,t)|\varepsilon_w^{\max}\\ \overline{f}(x,t) &= |\eta_1\tilde{p}(x,t)|\varepsilon_v^{\max},\end{aligned} \tag{6.16}$$

where $\varepsilon_\cdot^{\max} \equiv \mathbf{1}$ are upper bounds for the quantization error of the state solutions.

Denote by $e_p^{\max}, e_q^{\max}$ the solution of the adjoint error equations (6.12) with modified right-hand side (6.15). Then, using Theorem A.2, we get that

$$e_p \leq e_p^{\max}, \quad e_q \leq e_q^{\max}.$$

The error in the adjoint-for-Hessian due to quantization of the linearized-state trajectory is estimated by the same equation with modified term $c(x,t)$,

$$c(x,t) = |\chi_{\Omega_{\text{obs}}}\varepsilon_v - [\hat{I}_{\text{ion}}]_{vv}\varepsilon_v\tilde{\delta p} - [\hat{I}_{\text{ion}}]_{vw}\varepsilon_w\tilde{\delta p}|.$$

As in the scalar case, the error equations are solved once with right-hand side $\mathbf{1}$, and scaled with the correct right-hand side for evaluation of the error.

6.4.2. Newton-CG

As in the previous examples, the optimization progress is not affected by lossy compression of the trajectories (Figure 6.13). We show the corresponding compression factors for state and adjoint in Figure 6.14. As before, the quantization tolerances decrease during the Newton iterations. In contrast to the results presented in [37], here the quantization tolerance for the adjoint is chosen adaptively as well. The method is summarized in Algorithm 7, adapting Algorithm 5 to the present example.

Algorithm 7 Newton-CG with adaptive quantization for the monodomain example

1: fix initial $\delta_1^y, \delta_1^\lambda$ (provided by user)
2: **for** $i = 1, \ldots$ **do**
3: solve the state equations (6.5), encode v, w using δ_i^y
4: solve the adjoint equations (6.8), decode v, w using δ_i^y, and encode p using δ_i^λ
5: check optimality conditions, if optimal: end
6: solve the error estimator equations (6.12) with right-hand-side (6.15), (6.16) on a coarse, fixed mesh
7: use the CG method (Algorithm 4) to compute the Newton update using the quantization tolerance provided by equation (5.45) for encoding and decoding the solutions $\delta v, \delta w$ of the linearized-state equations (6.9) (v, w, p are decoded using $\delta_i^y, \delta_i^\lambda$)
8: update the control I_e using the Armijo rule for the step size
9: estimate new values for $\delta_{i+1}^y, \delta_{i+1}^\lambda$ using equations (5.43), (5.51), (5.52)
10: **end for**

For one solution of the state equation on a fixed mesh with 16 641 vertices, 408s CPU time were needed (averaged over all iterations), additionally compression required 24.6s. Solution of the adjoint equation required 388.1s on average, with additional 11.2s for compression. The error equations were solved on a mesh with 4225 vertices; error estimation required 96.6s CPU time per iteration. Overall, encoding/decoding incurred an overhead of 4.5%. During the eight Newton iterations, 25 linearized-state and adjoint-for-Hessian solves were required. On average, one Hessian-vector product required 1272.5s CPU time; the overhead for error estimation thus amounts to less than 2% per iteration.

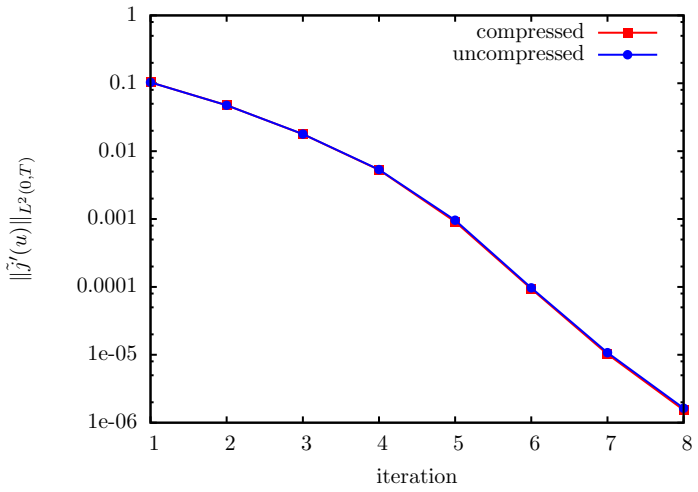

Figure 6.13.: L^2-norm of the reduced gradient with and without compression for the monodomain example (6.5), (6.6), using Newton-CG.

We show the behavior of the CG method during Newton iteration 7 in Figure 6.15. For comparison, the version without residual replacement is plotted as well, showing stagnation of the true residual (computed without quantization of the linearized-state trajectory) before the required accuracy is reached. Three residual replacements were computed, in CG iteration 3, 4, and 6 with compression factors of 13, 11.2, and 10.2. For this example, very high compression factors were achieved during the CG, ranging between 44 and 4758.7 over the course of the optimization. Overall, eight residual replacements had to be computed.

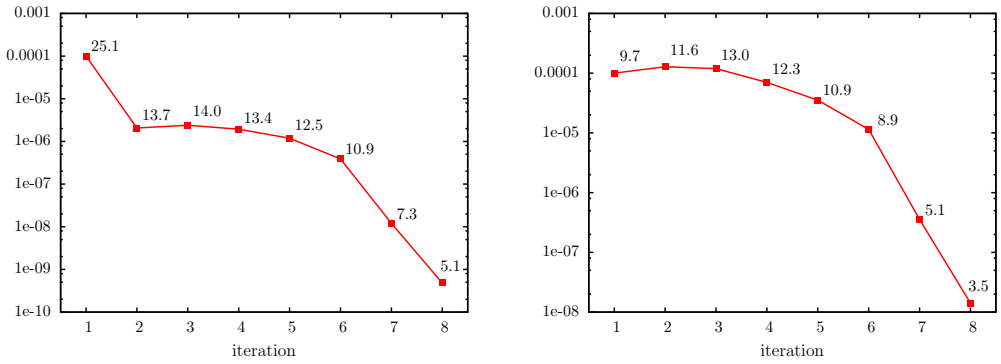

Figure 6.14.: Adaptively chosen quantization tolerances $\delta^{\cdot}$ and corresponding compression factors for state (left) and adjoint (right) for the monodomain example (6.5), (6.6) using Newton-CG.

6.4.3. BFGS-Quasi-Newton

The same optimization problem is solved using the BFGS-quasi-Newton method. For ease of implementation, the timestep size in the linearly implicit Euler method is fixed to $dt = 0.04$. Spatial adaptivity is performed individually for state and adjoint using the hierarchical DLY error estimator [26], with a restriction to at most 25 000 vertices in space. The adaptively refines grids were stored using the methods from [136], which reduced the storage space for the mesh to less than 1 bit/vertex.

Figure 6.16 shows the progress of the optimization method. For trajectory compression, different fixed quantization tolerances as well as the adaptively chosen δ were used. We estimate the spatial discretization error in the reduced gradient by using a solution on a finer mesh as a reference. Clearly, lossy compression has no influence on the optimization progress, up to discretization error accuracy.

The adaptively chosen quantization tolerances for the state values are shown in Figure 6.17. In the first iteration, a user-prescribed tolerance was used. The estimated condition number of the reduced Hessian varies between 200–230. Again we note that the adaptively chosen tolerances are too restrictive due to overestimation of the error and the fixed tolerance for the discretization.

For comparison of BFGS and Newton-CG, we apply both methods on a fixed mesh with 8321 vertices and $dt = 0.04$, with and without compression. We stop the optimization when the L^2-norm of the reduced gradient is below 10^{-6}.

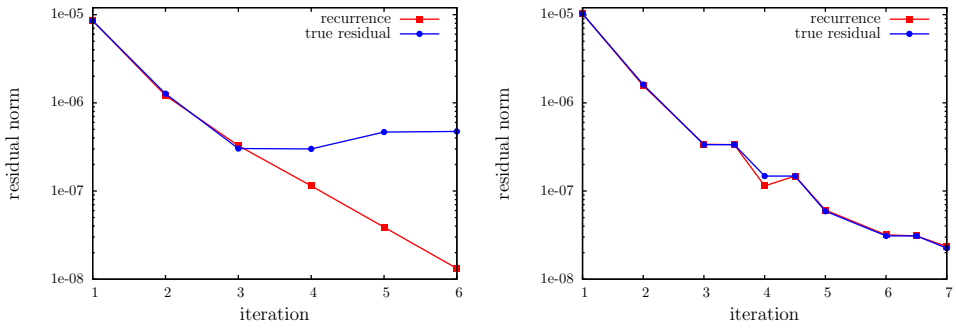

Figure 6.15.: Behavior of CG in the final Newton iteration of the monodomain example (6.5), (6.6). The *true residual* denotes the residual without quantization of the linearized state. Left: behavior without residual replacement. Right: with residual replacement.

While the Newton-CG method is faster in terms of convergence speed (10 Newton iterations vs. 15 BFGS-iterations), the number of PDE solves during the BFGS method is drastically smaller, see Figure 6.18. On average, 16 CG iterations per Newton iteration were necessary in this setting, each requiring the solution of two additional PDEs. The slightly larger number of PDE solves for Newton-CG with compression compared to the uncompressed version is due to residual re-computations. For both, Newton-CG and quasi-Newton, we used a fixed quantization tolerance of $\delta^y = 10^{-3}$ for the state, leading to a similar convergence behavior as in the uncompressed case. For the Newton-CG, the same quantization tolerance is used for the adjoint, $\delta^\lambda = \delta^y$, while the linearized-state is quantized adaptively as before. The computed controls of both methods show a good agreement (Figure 6.19).

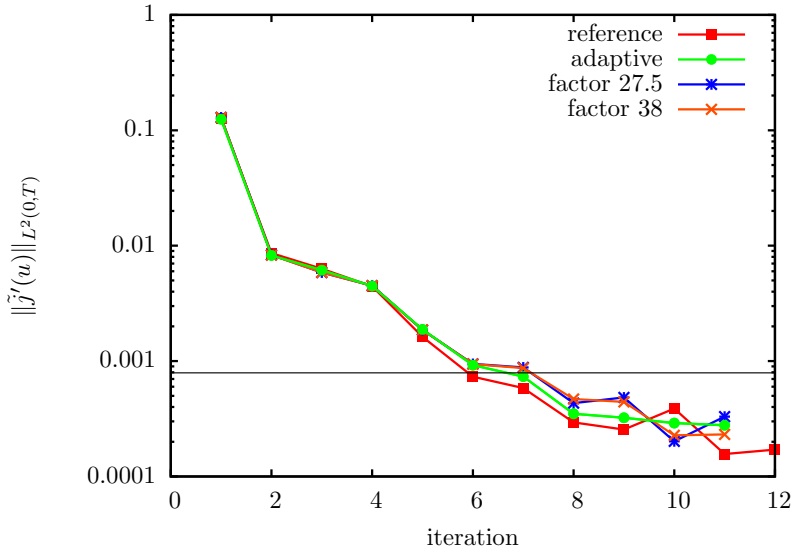

Figure 6.16.: Optimization progress of BFGS for the monodomain example (6.5), (6.6), using different quantization tolerances for the state trajectory. No delta-encoding between timesteps was used.

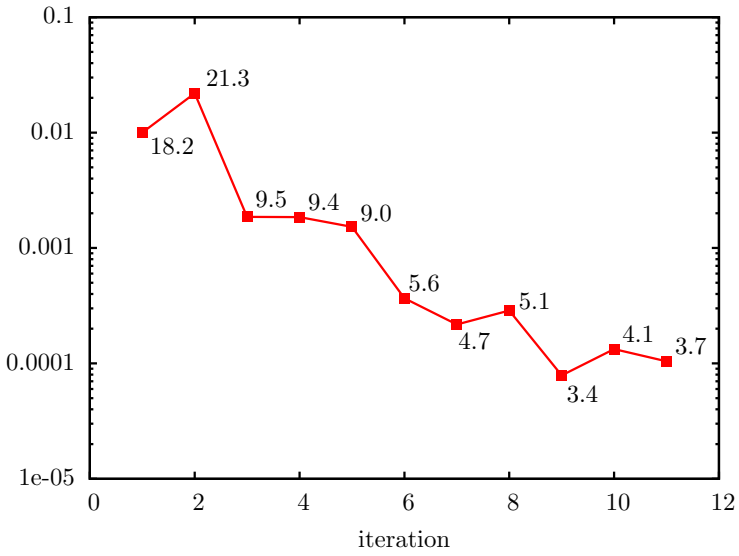

Figure 6.17.: Adaptively chosen quantization tolerances δ^y and corresponding compression factors for the monodomain example (6.5), (6.6) using BFGS.

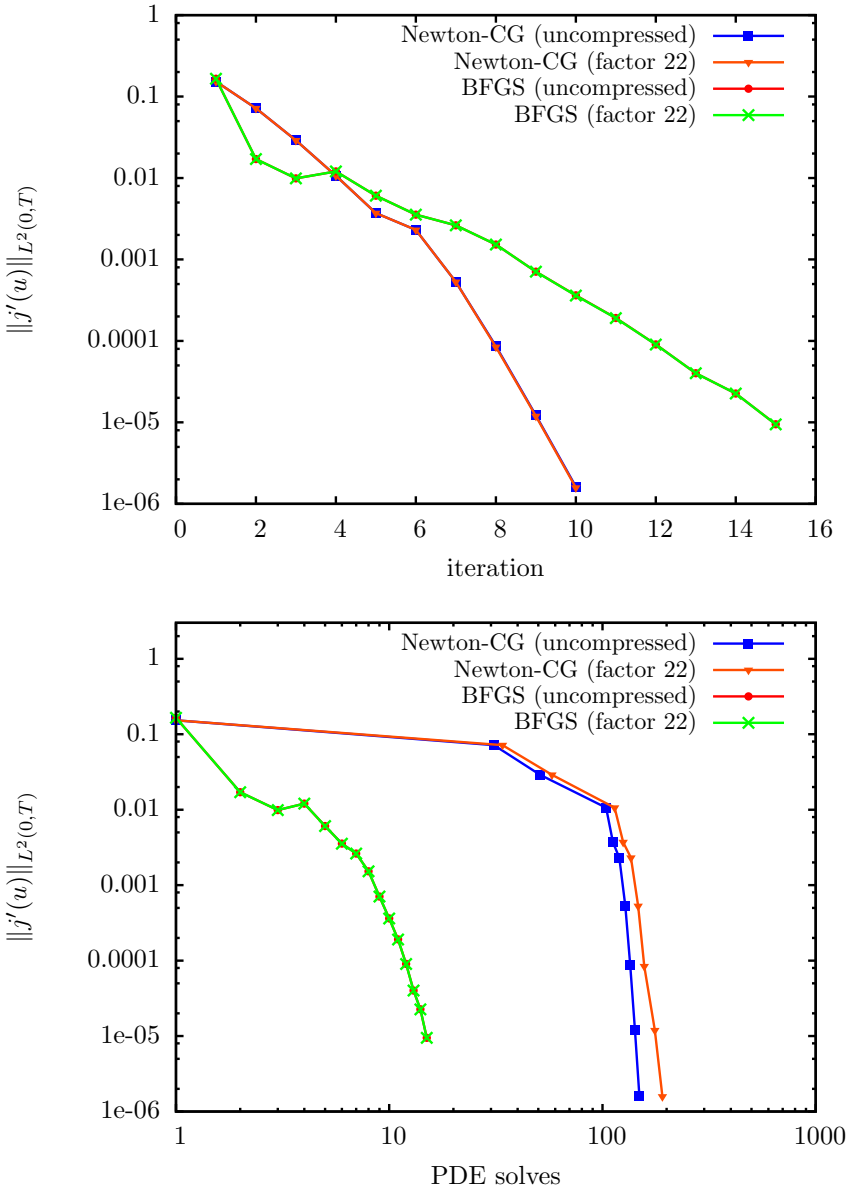

Figure 6.18.: Optimization progress of BFGS and Newton-CG for the monodomain example (6.5), (6.6). Top: iterations. Bottom: PDE solutions.

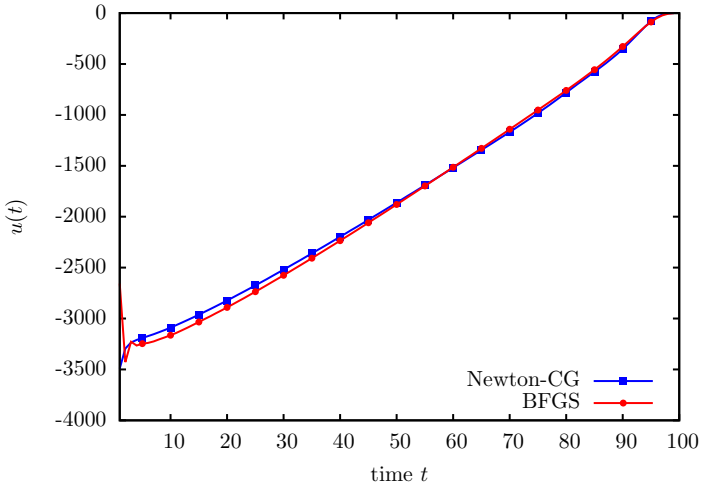

Figure 6.19.: Computed controls of BFGS and Newton-CG for the monodomain example (6.5), (6.6).

7. Beyond Pointwise Error Control

Partially due to the worst-case error estimation, the resulting adaptive quantization tolerances are too restrictive, see Chapters 5 and 6. While, in the estimates, we assume the quantization error to have a single sign, this is not the typical case in the numerical experiments. There, the error is mostly oscillatory, such that it is damped due to the smoothing properties of parabolic PDEs. To increase the efficiency of the compression, it is of interest to estimate and bound the error in different norms than L^∞. This can be achieved by modifying the *transform*-part of the encoding scheme, and generalizing the hierarchical basis-decomposition of Section 4.2 to a wavelet-decomposition.

Consider again the error in the adjoint equation due to lossy compression (Theorem 5.1.1). For the simplified case of a linear state equation and a tracking-type functional, we have

$$c_y(\hat{y}, u)^\star e_\lambda = -\varepsilon_y,$$

with homogeneous boundary- and terminal conditions (cf. Example 5.2.2). For solvability of this equation, we require only $\varepsilon_y \in L^2(0,T;V^\star)$, with $V^\star = H^{-1}(\Omega)$, and get the a-priori estimate [144]

$$\|e_\lambda\|_{W(0,T)} \leq C \, \|\varepsilon_y\|_{L^2(0,T;H^{-1}(\Omega))} ,$$

with $\|y\|_{W(0,T)} = \|y\|_{L^2(0,T;V)} + \|y'\|_{L^2(0,T;V^\star)}$ (see also Chapter 2).

In view of this function space setting, bounding the compression error in $H^{-1}(\Omega)$ would be preferable, penalizing constant parts of the error and allowing for larger, but oscillatory errors.

In this chapter we are again concerned with spatial compression only, as the temporal compression is lossless. In Section 7.1, we discuss the ingredients of wavelet-based compression in some detail, before turning to the actual construction of suitable wavelet bases in Section 7.2. First numerical results are given in Section 7.3.

7.1. Wavelet-Based Compression

Let H be a real Hilbert space with scalar product $(\cdot,\cdot)$, and let $\{S_j\}_{j\geq 0}$ be a sequence of subspaces $S_j \subset H$ such that

$$S_j \subset S_{j+1} \quad \text{and} \quad \operatorname{clos} \bigcup_{j=0}^{\infty} S_j = H, \tag{7.1}$$

i.e. the sequence is strictly increasing and dense in H. Let further

$$\mathbf{\Psi} = \{\psi_i \mid i \in \mathcal{I}\} \subset H$$

be a wavelet basis, where $\mathcal{I}$ denotes some suitable index set. In view of the hierarchical finite element spaces we split up the index $i = (j,k)$, where j later refers to the grid level and $k \in \mathcal{K}_j$ specifies the basis function. A function $f \in H$ can thus be decomposed as

$$f(x) = \sum_{j=0}^{\infty} \sum_{k\in\mathcal{K}_j} \xi_{j,k}\psi_{j,k}(x). \tag{7.2}$$

The collection $\mathbf{\Psi}$ is called H-stable if and only if the expansion (7.2) is unique for every $f \in H$ and for some fixed positive weights $w_{j,k}$ the norm equivalence

$$c_1\Big(\sum_{j=0}^{\infty}\sum_{k\in\mathcal{K}_j} w_{j,k}^2\ \xi_{j,k}^2\Big)^{1/2} \leq \Big\|\sum_{j=0}^{\infty}\sum_{k\in\mathcal{K}_j} \xi_{j,k}\ \psi_{j,k}\Big\|_H \leq c_2\Big(\sum_{j=0}^{\infty}\sum_{k\in\mathcal{K}_j} w_{j,k}^2\ \xi_{j,k}^2\Big)^{1/2} \tag{7.3}$$

holds, where the constants c_1, c_2 are independent of f, see Dahmen [19]. H-stability means that the collection $\mathbf{\Psi}$ is a Riesz basis of H. In the following we use $x \lesssim y$ to express that x can be bounded by a some constant multiple of y, and use $x \sim y$ if $x \lesssim y$ and $y \lesssim x$.

Fix some highest level l with subspace S_l in the sequence (7.1). For a function $y \in S_l$ we get the expansion

$$y(x) = \sum_{j=0}^{l} \sum_{k\in\mathcal{K}_j} \xi_{j,k}\psi_{j,k}(x).$$

The norm equivalence allows for two different compression strategies for such a function y, see, e.g., DeVore [30]. If we allow for a given maximum number of n coefficients, the smallest error is achieved by selecting those coefficients for which

$$|\xi_{j,k}|\,\|\psi_{j,k}\|_H$$

is largest. Typically this strategy is modified, using thresholding to avoid sorting: all coefficients for which

$$|\xi_{j,k}| \, \|\psi_{j,k}\|_H > \epsilon$$

for some given $\epsilon > 0$ are retained. This can be efficiently implemented using the EZW (*embedded zerotree wavelet*) algorithm [113].

The second method is more in the flavor of this thesis, and can be seamlessly included into the transform coding framework sketched in Figure 4.1. Here, the coefficients $\xi_{j,k}$ are replaced by an approximation $\hat{\xi}_{j,k}$, i.e. their quantized version, such that for some given tolerance ϵ

$$\|\hat{y} - y\|_H \leq \epsilon \tag{7.4}$$

holds. These coefficients $\hat{\xi}_{j,k}$ are then stored using entropy coding. From the norm equivalence (7.3) we can derive a quantization tolerance δ to ensure (7.4). Keeping the coarse grid values uncompressed, i.e. lossless, we have

$$\|\hat{y} - y\|_H^2 \sim \sum_{j=1}^{l} \sum_{k \in \mathcal{K}_j} w_{j,k}^2 (\hat{\xi}_{j,k} - \xi_{j,k})^2. \tag{7.5}$$

Error equilibration, i.e. requiring $w_{j,k}^2 (\hat{\xi}_{j,k} - \xi_{j,k})^2 \leq \delta^2 \ \forall j,k$, leads to a uniform quantization tolerance

$$\delta \lesssim \frac{\epsilon}{\left(\sum_{j=1}^{l} |\mathcal{K}_j|\right)^{1/2}}. \tag{7.6}$$

The sharpness of the estimated δ depends on the condition of the wavelet basis, i.e. the ratio c_2/c_1 of the constants in the norm equivalence. The correct choice for the weights $w_{j,k}$ depends on the actual wavelet constructions and the norm $\|\cdot\|_H$, and will be discussed in the numerical examples, Section 7.3. When a norm equivalence is not available, the quantization tolerance δ can be determined using the triangle inequality. This will lead to smaller δ and thus to worse compression factors. Using the wavelet basis, the error norm $\|\hat{y} - y\|_H$ can be estimated as

$$\|\hat{y} - y\|_H = \Big\| \sum_{j=1}^{l} \sum_{k \in \mathcal{K}_j} (\hat{\xi}_{j,k} - \xi_{j,k}) \psi_{j,k} \Big\|_H \leq \sum_{j=1}^{l} \sum_{k \in \mathcal{K}_j} |\hat{\xi}_{j,k} - \xi_{j,k}| \|\psi_{j,k}\|_H. \tag{7.7}$$

Requiring $|\hat{\xi}_{j,k} - \xi_{j,k}| \|\psi_{j,k}\|_H \leq \delta \ \forall j,k$ leads to a quantization tolerance

$$\delta \lesssim \frac{\epsilon}{\sum_{j=1}^{l} |\mathcal{K}_j|}, \tag{7.8}$$

again up to constants. For using this bound, an estimate for $\|\psi_{j,k}\|_H$ has to be available.

7.1.1. Multiresolution Analysis

Before turning to the construction of suitable wavelet bases, we recapitulate some facts on multiscale decompositions using wavelets as introduced above. There exists a large amount of literature on multiresolution analysis and wavelet theory; for a more detailed discussion than given here, we refer to Dahmen and coworkers [12, 18, 19] and Sweldens [122], on which this and the following section are based.

We start from the subspace sequence $\{S_j\}_{j\geq 0}$ satisfying (7.1). Let $\mathbf{\Phi}_j = \{\varphi_{j,k} \mid k \in \mathcal{K}_j\}$ be a Riesz basis of S_j with finite index sets $\mathcal{K}_j$. Such a sequence of closed subspaces is called *multiresolution analysis*, the functions φ are called *scaling functions*. A *dual* multiresolution analysis $\{\tilde{S}_j\}_{j\geq 0}$ consist of closed subspaces $\tilde{S}_j \subset H$ with Riesz bases given by functions $\tilde{\varphi}_{j,k}$ dual to $\varphi_{j,k}$, satisfying biorthogonality, i.e.

$$(\varphi_{j,k}, \tilde{\varphi}_{j,k'}) = \delta_{k,k'} \text{ for } k, k' \in \mathcal{K}_j.$$

The nestedness of the spaces $\{S_j\}_{j\geq 0}$ together with the stability of the bases $\mathbf{\Phi}_j$ implies the existence of *filter* coefficients $\{h_{j,k,l} \mid l \in \mathcal{K}_{j+1}\}$ such that the *refinement relation*

$$\varphi_{j,k} = \sum_{l\in\mathcal{K}_{j+1}} h_{j,k,l}\varphi_{j+1,l} \tag{7.9}$$

holds.

Due to (7.1), S_{j+1} can be decomposed as

$$S_{j+1} = S_j \oplus W_j, \tag{7.10}$$

with the complement space W_j given by

$$W_j = \operatorname{clos}\operatorname{span}\{\psi_{j,k} \mid k \in \mathcal{M}_j\}$$

for $\mathcal{M}_j = \mathcal{K}_{j+1} \setminus \mathcal{K}_j$. If $\mathbf{\Phi}_j \cup \mathbf{\Psi}_j$ with $\mathbf{\Psi}_j = \{\psi_{j,k} \mid k \in \mathcal{M}_j\}$ is uniformly stable, the functions $\psi_{j,k}$ are called *wavelets*.

On irregular, unstructured meshes, wavelet constructions yielding orthogonality $W_j \perp S_j$ typically are not available. Orthogonality thus has to be replaced by a less restrictive condition. One suitable possibility is to require orthogonality of W_j not to S_j, but to a dual space $\tilde{S}_j$, $W_j \perp \tilde{S}_j$.

Similar to $\tilde{S}_j$, the dual wavelet space $\tilde{W}_j$ is given by a basis consisting of dual wavelets $\tilde{\psi}_{j,m}$ biorthogonal to the primal wavelets,

$$(\psi_{j,m}, \tilde{\psi}_{j',m'}) = \delta_{j,j'}\delta_{m,m'} \text{ for } m, m' \in \mathcal{M}_j.$$

These dual spaces $\tilde{W}_j$ complement $\tilde{S}_j$ in $\tilde{S}_{j+1}$ and are orthogonal to S_j, $\tilde{W}_j \perp S_j$. For any function $f \in H$, the dual wavelets $\tilde{\psi}_{j,k}$ are used to define the coefficients $\xi_{j,k} = \langle f, \tilde{\psi}_{j,k} \rangle$ in the representation (7.2).

Like the scaling functions, the wavelets satisfy refinement relations

$$\psi_{j,m} = \sum_{k \in \mathcal{K}_{j+1}} g_{j,m,k} \varphi_{j+1,k}. \tag{7.11}$$

Additionally, also the dual scaling functions and dual wavelets fulfill similar refinement relations with coefficients $\tilde{h}, \tilde{g}$.

Repeating the decomposition (7.10), the space S_l for some fixed l can be written as the sum of complement spaces,

$$S_l = S_0 \bigoplus_{j=0}^{l-1} W_j,$$

with the multiscale basis

$$\mathbf{\Psi}_l = \{\varphi_{0,k} \mid k \in \mathcal{K}_0\} \bigcup_{j=0}^{l-1} \{\psi_{j,k} \mid k \in \mathcal{M}_j\}.$$

By density of the decomposition (7.1),

$$\mathbf{\Psi} = \{\varphi_{0,k} \mid k \in \mathcal{K}_0\} \bigcup_{j=0}^{\infty} \{\psi_{j,k} \mid k \in \mathcal{M}_j\}.$$

is a candidate for a basis of the space H.

7.1.2. Fast Wavelet Transform

To use wavelet-based compression during the numerical solution of optimal control problems, it is of utmost importance that the transform from the single scale representation of the finite element solution $y_h \in S_l$ to the multiscale representation (7.14) (the wavelet transform, in signal processing often referred to as *analysis*), and vice versa (the inverse wavelet transform, or *synthesis*) are computationally inexpensive. This is achieved with the fast wavelet transform.

Let be given a set of single-scale coefficients on a fixed, finest discretization level l,

$$\{y_{l,k} \mid k \in \mathcal{K}_l\} \text{ such that } y = \sum_{k \in \mathcal{K}_l} y_{l,k} \varphi_{l,k}.$$

Computing from this single scale representation the multiscale coefficients

$$\{\xi_{j,m} \mid 0 \le j < l, m \in \mathcal{M}_j\},\ \{y_{0,k} \mid k \in \mathcal{K}_0\}$$

such that

$$y = \sum_{k\in\mathcal{K}_0} y_{0,k}\varphi_{0,k} + \sum_{j=0}^{l-1}\sum_{m\in\mathcal{M}_j} \xi_{j,m}\psi_{j,m}$$

is performed by recursive application from $l-1$ to 0 of

$$y_{j,k} = \sum_{l\in\tilde{\mathcal{L}}(j,k)} \tilde{h}_{j,k,l}y_{j+1,l} \quad \text{and} \quad \xi_{j,m} = \sum_{l\in\tilde{\mathcal{L}}(j,m)} \tilde{g}_{j,m,l}y_{j+1,l}. \tag{7.12}$$

The inverse transform, converting a multiscale representation to the single scale, can be computed by

$$y_{j+1,l} = \sum_{k\in\mathcal{K}(j,l)} h_{j,k,l}y_{j,k} + \sum_{m\in\mathcal{M}(j,l)} g_{j,m,l}\xi_{j,m} \tag{7.13}$$

for $j = 1, \dots, l-1$. In these formulas, the sets used for summation are given by

$$\begin{aligned}\mathcal{M}(j,l) &= \{m \in \mathcal{M}_j \mid g_{j,m,l} \neq 0\}\\ \mathcal{L}(j,m) &= \{l \in \mathcal{K}_{j+1} \mid m \in \mathcal{M}(j,l)\}\\ \mathcal{K}(j,l) &= \{k \in \mathcal{K}_j \mid h_{j,k,l} \neq 0\},\end{aligned}$$

with analogous definitions of $\tilde{\mathcal{L}}$ etc., see [122]. If the size of each of the sets $\mathcal{K}, \mathcal{L}, \tilde{\mathcal{K}}, \tilde{\mathcal{L}}$ is uniformly bounded for all j, k, l, transform and inverse transform have linear complexity.

7.2. Construction of Wavelet Bases

After the general discussion above, we turn now to the actual construction of suitable wavelet bases satisfying stability and vanishing moment conditions. After a short general introduction of the lifting scheme, we are concerned with wavelets based on finite elements.

For the wavelet decomposition of a finite element function $y_h \in S_l$ let

$$S_0 \subset S_1 \subset \cdots \subset S_l \subset H$$

be a sequence of linear finite element spaces, constructed over uniformly refined simplicial triangulations

$$\mathcal{T}_0 \subset \mathcal{T}_1 \subset \cdots \subset \mathcal{T}_l$$

of a polyhedral domain $\Omega \subset \mathbb{R}^d$. We get

$$y_h(x) = \sum_{j=0}^{l} \sum_{k \in \mathcal{K}_j} \xi_{j,k} \psi_{j,k}(x), \tag{7.14}$$

with $\mathcal{K}_j = \mathcal{N}_j$, the number of nodes on grid level j. Relation (7.3) yields the equivalence of the ℓ^2-norm of the coefficients and the norm of the function y_h for suitably chosen wavelets $\psi_{j,k}$.

7.2.1. Lifting

The lifting scheme due to Sweldens [122] is an easily implementable method to construct wavelet bases satisfying vanishing moment conditions, starting from some initial multiresolution analysis. It is a special case of the more general stable completion approach presented in Carnicer, Dahmen, and Peña [12].

Theorem 7.2.1. *[122] Let an initial set of scaling functions $\varphi^0_{j,k}, \tilde{\varphi}^0_{j,k}$ and wavelets $\psi^0_{j,k}, \tilde{\psi}^0_{j,k}$ be given, together with coefficients $h^0_{j,k,l}, \tilde{h}^0_{j,k,l}, g^0_{j,m,l}, \tilde{g}^0_{j,m,l}$ from the refinement relations. Then, for arbitrary lifting coefficients $s_{j,k,m}$, the scaling functions and wavelets defined by*

$$\begin{aligned}
\varphi_{j,k} &= \varphi^0_{j,k} \\
\tilde{\varphi}_{j,k} &= \sum_l \tilde{h}^0_{j,k,l} \tilde{\varphi}_{j+1,l} + \sum_m s_{j,k,m} \tilde{\psi}_{j,m} \\
\psi_{j,m} &= \psi^0_{j,m} - \sum_k s_{j,k,m} \varphi^0_{j,k} \\
\tilde{\psi}_{j,m} &= \sum_l \tilde{g}^0_{j,k,m} \tilde{\varphi}_{j+1,l}
\end{aligned} \tag{7.15}$$

are a collection of biorthogonal primal and dual scaling functions and wavelets. They satisfy refinement relations with coefficients given by

$$\begin{aligned}
h_{j,k,l} &= h^0_{j,k,l} \\
\tilde{h}_{j,k,l} &= \tilde{h}^0_{j,k,l} + \sum_m s_{j,k,m} \tilde{g}^0_{j,m,l} \\
g_{j,m,l} &= g^0_{j,m,l} - \sum_k s_{j,k,m} h^0_{j,k,l} \\
\tilde{g}_{j,m,l} &= \tilde{g}^0_{j,m,l}.
\end{aligned} \tag{7.16}$$

The freedom in the choice of the coefficients $s_{j,k,m}$ can be used to impose conditions on the wavelets, like vanishing moments. The lifting procedure can be included into the fast wavelet and inverse wavelet transforms in a straightforward way, see Algorithm 8.

Algorithm 8 Fast lifted wavelet transform

$$\begin{aligned}
&\text{Stage 1:} && y_{j,k} \leftarrow \sum_l \tilde{h}^0_{j,k,l}\ y_{j+1,l} && \forall k \in \mathcal{N}_j \\
&&& \xi_{j,m} \leftarrow \sum_l \tilde{g}^0_{j,m,l}\ y_{j+1,l} && \forall m \in \mathcal{N}_{j+1} \setminus \mathcal{N}_j \\
&\text{Stage 2:} && y_{j,k} \leftarrow y_{j,k} + \sum_m s_{j,k,m}\ \xi_{j,m} && \forall k \in \mathcal{N}_j
\end{aligned}$$

The lifting construction requires an initial set of scaling functions and wavelets. In the linear finite element setting at hand, the hierarchical basis provides a simple initial setting. There, the scaling functions $\varphi^0_{j,k}$ are modified finite element basis functions on level j,

$$\varphi_{j+1,k} = \begin{cases} \varphi_{j,k} - \frac{1}{2}\sum_m \psi_{j,m}, & k \in \mathcal{N}_j \\ \psi_{j,k}, & k \in \mathcal{N}_{j+1} \setminus \mathcal{N}_j, \end{cases}$$

while the finite element basis functions corresponding to the vertices on level $j+1$ define the wavelets $\psi^0_{j,m} = \varphi^0_{j+1,m}$. The formal dual of these interpolating scaling functions are Dirac functions $\tilde{\varphi}^0_{j,k}(x) = \delta(x - x_k)$, where x_k as usual denotes the coordinate of vertex $k \in \mathcal{N}_j$. Similarly, the dual wavelets are given by

$$\tilde{\psi}_{j,k} = \delta(x - x_k) - \frac{1}{2} \sum_{m \in \mathcal{N}_{j+1} \setminus \mathcal{N}_j} \delta(x - x_m).$$

As the hierarchical basis has no dual in L^2, and thus is not a Riesz basis for L^2, it can not be used for constructing compression algorithms bounding the L^2-error, but serves as a starting point for the construction of suitable wavelet bases.

7.2.2. Finite Element Wavelets

In this section we present two lifting-based constructions, starting from the hierarchical basis decomposition. These constructions have the important advantage that they can be easily integrated into existing finite element codes. Further, the created dual wavelets have small support, allowing an efficient implementation and

thus induce only a minor computational overhead. On the downside, these constructions do not yield H^{-1}-stable bases. Additionally, two other approaches are shortly summarized, which yield H^{-1} equivalence, but drastically increase the computational complexity and implementation effort. All constructions are described for two space-dimensions, but can readily be extended to 3D.

7.2.2.1. Linear interpolatory vertex bases

The first construction, described by Schröder and Sweldens [109], as well as Amaratunga and Castrillón-Candás [4] adds one vanishing moment,

$$\int_\Omega \psi_{j,k}(x)\ dx = 0,$$

to the wavelet construction. This is enforced using only the two parent nodes of a vertex $m \in \mathcal{N}_{j+1} \setminus \mathcal{N}_j$, such that the resulting wavelets and dual wavelets have local support.

The scaling functions are defined by the refinement relation

$$\varphi_{j,k} = \varphi_{j+1,k} + \frac{1}{2} \sum_{m \in n(j,k)} \varphi_{j+1,m}, \tag{7.17}$$

with $n(j,k) = \{m \in \mathcal{N}_{j+1} \setminus \mathcal{N}_j \mid k \in \mathcal{N}_j \text{ is a parent of } m\}$. Defining

$$I_{j,k} = \int_\Omega \varphi_{j,k}(x)\ dx,$$

the wavelets are constructed using the lifting coefficients $s_{j,k,m} = I_{j+1,m}/(2I_{j,k})$. This gives the refinement relation

$$\psi_{j,m} = \varphi_{j+1,m} - \sum_{k \in A(j,m)} s_{j,k,m} \varphi_{j,k}. \tag{7.18}$$

There, $A(j,m) = \{k \in \mathcal{N}_j \mid m \in \mathcal{N}_{j+1} \setminus \mathcal{N}_j \text{ is a child of } k\}$. For illustration, a resulting wavelet is depicted in Figure 7.2 (top).

In his thesis [14], Castrillón-Candás shows that the multiresolution analysis resulting from these relations yields, for functions $f \in H_0^3(\Omega)$, the norm equivalence (7.3) with $H = L^2(\Omega)$ and weights $w_{j,k} = 1/h_j$, where h_j is the characteristic size of the level j mesh, i.e. for uniform refinement $h_j \sim 2^{-j}$.

7.2.2.2. Construction due to Cohen et al.

Again starting with the hierarchical basis, Cohen et al. [16] develop a lifting-based construction of biorthogonal wavelets on polygonal domains. To take care of irregular grids and boundary effects, they divide the nodes of the triangulation into three classes, and choose the lifting coefficients adapted to each class.

This classification is created as follows. From the initial triangulation, all common sides of two triangles such that these two triangles form a parallelogram are deleted. The resulting object is called *frame*. A vertex is called *exceptional*, if it lies on the intersection of different line segments of the frame. The remaining vertices on the frame form the second class, the frame nodes. All remaining vertices are *inner nodes*, see Figure 7.1. For their construction they require that at most one node in a triangle is an exceptional node, which is fulfilled if the initial triangulation is sufficiently fine. In the following $A(j,m)$ and $I_{j,k}$ are defined as in the previous section.

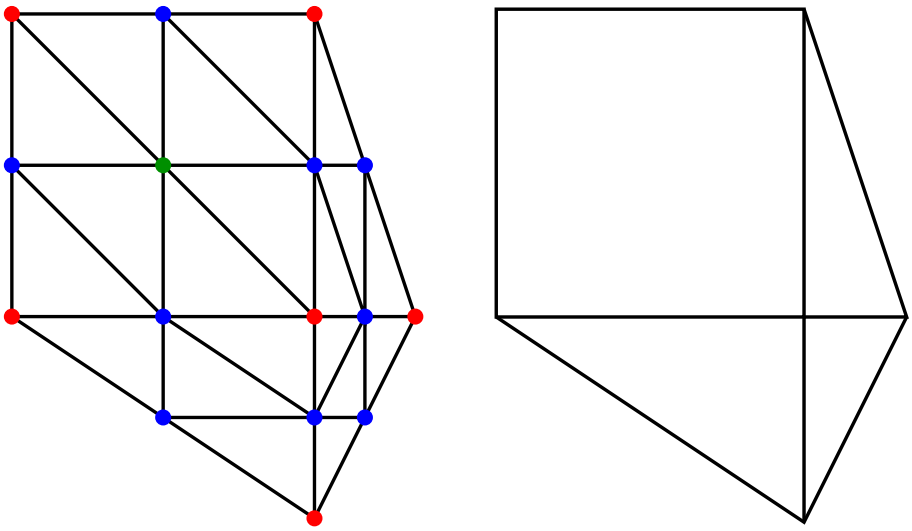

Figure 7.1.: Classification of vertices (adapted from [16]). Left: initial triangulation of a domain Ω. Right: frame. On the left, exceptional nodes are marked red, frame nodes blue, the remaining inner node green.

Exceptional nodes k are lifted using

$$s_{j,k,m} = \begin{cases} I_{j+1,m}/I_{j,k}, & k \text{ is exceptional and a neighbor of } m \\ 0, & \text{otherwise} \end{cases}. \tag{7.19}$$

For frame nodes two cases need to be distinguished. If both parents of m are on the frame, the lifting coefficients are given by

$$s_{j,k,m} = \begin{cases} \frac{1}{2} I_{j+1,m}/I_{j,k}, & k \in A(j,m) \\ 0, & \text{otherwise} \end{cases}. \tag{7.20}$$

If only one parent $k_f \in A(j,m)$ is on the frame, the lifting coefficient is nonzero only for this vertex:

$$s_{j,k,m} = \begin{cases} I_{j+1,m}/I_{j,k}, & k = k_f \\ 0, & \text{otherwise} \end{cases}. \tag{7.21}$$

Finally, for the inner nodes, the lifting coefficients are given by

$$s_{j,k,m} = \begin{cases} \frac{3}{4} I_{j+1,m}/I_{j,k}, & k \in A(j,m) \\ -\frac{1}{4} I_{j+1,m}/I_{j,k}, & k \in T(j,m) \end{cases}. \tag{7.22}$$

Here, $T(j,m) \subset \mathcal{N}_j \setminus A(j,m)$ denotes the vertices of the triangles of level j which have $m \in \mathcal{N}_{j+1} \setminus \mathcal{N}_j$ as midpoint of an edge and are not the parents of m. A wavelet for inner nodes is depicted in Figure 7.2 (bottom), where it can be compared to the wavelet resulting from the construction of the previous section.

In [16] it is shown that this construction yields dual scaling functions in $H^{\tilde{s}}$ with $\tilde{s} < 0.114$, and thus norm equivalences for H^s, $-0.114 < s < 3/2$. Moreover, for $0 \le s < 3/2$ and $0 \le \tilde{s} < 0.114$ the estimates

$$\|\varphi_{j,k}\|_{H^s} \le C 2^{j(s-1)} \qquad \text{and} \qquad \|\tilde{\varphi}_{j,k}\|_{H^{\tilde{s}}} \le C 2^{j(\tilde{s}+1)} \tag{7.23}$$

hold, with C independent of j and k. By the refinement relation (7.11), these estimates also hold for the wavelets $\psi_{j,k}$

7.2.2.3. Other approaches

To conclude the section, we present two additional wavelet constructions, which yield the required norm equivalences.

In [20], Dahmen and Stevenson construct finite element wavelet bases, achieving $H^s(\Omega)$-stability for $-3/2 < s < 3/2$. Nguyen and Stevenson [95] modify this construction to improve the conditioning of the wavelets, i.e. the ratio c_2/c_1 of the constants in the norm equivalence (7.3). Their construction is split in two parts. First, wavelets on a reference element are computed, depending on the space dimension and the requested number of primal and dual vanishing moments. In the second step, adaptation of these wavelets to the actual mesh and boundary conditions are

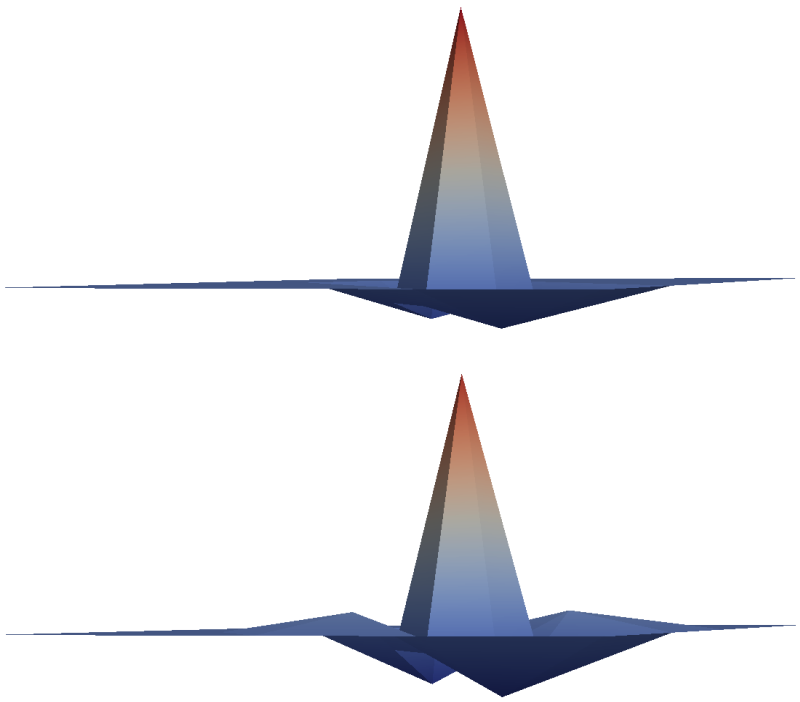

Figure 7.2.: Comparison of lifted wavelets constructed due to Section 7.2.2.1 (top) and Section 7.2.2.2 (bottom).

given. This construction can be implemented efficiently, and allows a fast inverse wavelet transform, i.e. the transformation from wavelet to nodal basis. However, as the construction does not yield localized dual wavelets it is not suitable for use in compression algorithms, as the computational cost for the transformation from nodal to wavelet basis is prohibitive.

As a continuation of that work, Stevenson [118] gives a construction for biorthogonal wavelets on nonuniform meshes, which also give H^s-stability for $|s| < 3/2$, but in addition yield locally supported dual wavelets. The construction is based on standard Lagrange finite elements on meshes created by uniform dyadic refinement of an arbitrary initial mesh. Concrete realizations are given for examples in 1D and 2D. On the downside, already in 2D the constructions are rather complex, and computationally more expensive than the approaches presented above.

There are many other approaches available in the literature. Besides numerous wavelet constructions in special settings, different ideas for stabilizing multiresolution decompositions can be found. One example, using wavelet-like basis func-

tions to modify the classical hierarchical basis, was derived by Vassilevski and Wang [132, 133]. They achieve L^2-stability by subtracting from the hierarchical basis functions their approximately computed L^2-projection on coarser grid levels. While their intention is the construction of an efficient multilevel preconditioner, it can be used for compression as well. As a complete overview is beyond the scope of this chapter, we restrict ourselves to the constructions given in Sections 7.2.2.1, 7.2.2.2 for the numerical experiments.

7.3. Numerical Results

As a first step, we consider the two test functions f_1, f_3 of Section 6.1,

$$f_1(x) = \sin(12(x_0-0.5)(x_1-0.5)), \quad f_3(x) = \sin(50(x_0-0.5)(x_1-0.5)), \quad x \in [0,1]^2.$$

For comparison, the hierarchical basis transform together with L^∞-quantization was performed with a tolerance δ to achieve interpolation error accuracy. In a second step, the two wavelet transforms described in the previous section were used to compress these functions with tolerances to keep the same L^2, resp. H^{-1}, accuracy as the hierarchical basis compression.

For determination of the quantization tolerance δ for the wavelet compression, we combine (7.6) with the decay estimates (7.23) yielding weights $w_{j,k} = 2^{-j}$ for controlling the L^2-norm. For H^{-1} we use $w_{j,k} = 2^{-2j}$, i.e. (7.23) with $s = -1$. Although the regularity proofs do not give H^{-1} equivalence, this heuristic choice works very well in our examples. In Figure 7.3 the behavior of the H-norm of the wavelets with regard to the grid levels is shown; a good agreement with the assumed decay can be observed. Note that in the finite element setting at hand, in equation (7.6) we have $\sum_j |\mathcal{K}_j| = |\mathcal{N}_l \setminus \mathcal{N}_0|$, i.e. the number of vertices is the finest mesh without the coarse grid nodes.

The results for encoding f_1, f_3 are depicted in Figure 7.4. For keeping the L^2 error bound, no notable increase of the compression factor is achieved. In contrast, the H^{-1} error bound allows for a significantly larger quantization tolerance compared to L^∞ quantization, and thus yields a compression factor up to four times higher than before. For the latter, Figure 7.5 shows the quantization errors for f_1 on a uniform mesh with 16 641 nodes on 7 levels.

As a second example, we use the construction from Section 7.2.2.2 for solving the Kolmogorov optimal control problem from Section 6.3 with the BFGS-quasi-Newton method. We discretize by finite elements on a fixed mesh with 8192 elements/4225 vertices and use the linearly implicit Euler scheme with timestep size $dt = 0.05$.

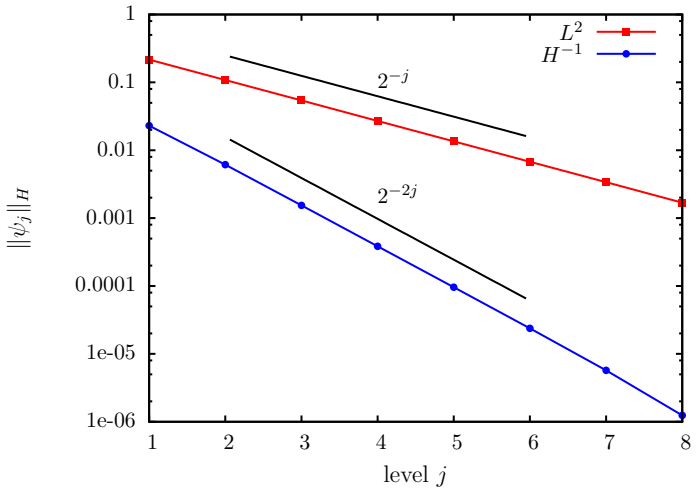

Figure 7.3.: Decay of the H-norm of the wavelets with regard to the grid levels.

For the hierarchical basis transform, we fix a quantization tolerance $\delta^{\mathrm{HB}} = 10^{-5}$, controlling the L^∞-norm of the reconstruction error. For the wavelet transform, prescribing $\epsilon = 10^{-6}$ for controlling the H^{-1}-norm of the error yields the quantization tolerance $\delta^{\mathrm{WLT}} = 1.54 \cdot 10^{-8}$, leading to a similar optimization progress (Figure 7.6). While the average compression factor for the hierarchical basis compression is about 3.5, wavelet-based compression gives a factor 22.7, thus increasing the compression factor by nearly 650%.

Remark. These first results indicate that wavelet-based compression schemes can further improve the compression factors. In the example, we were able to control the H^{-1}-error of the reconstruction, although theoretical results on norm equivalences for H^{-1} are not available. This short concluding chapter should serve as a starting point for future research.

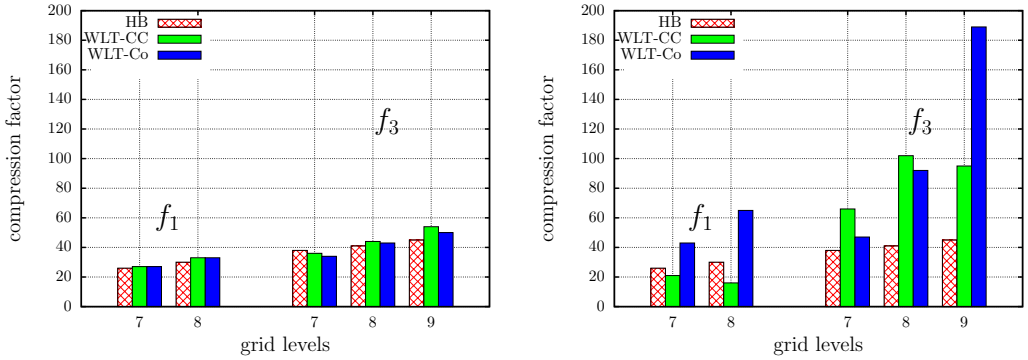

Figure 7.4.: Comparison of different transforms. Left: compression factors for L^2 error bounds. Right: compression factors for H^{-1} error bounds. For the wavelets, WLT-CC refers to the construction of Section 7.2.2.1, WLT-CO to Section 7.2.2.2.

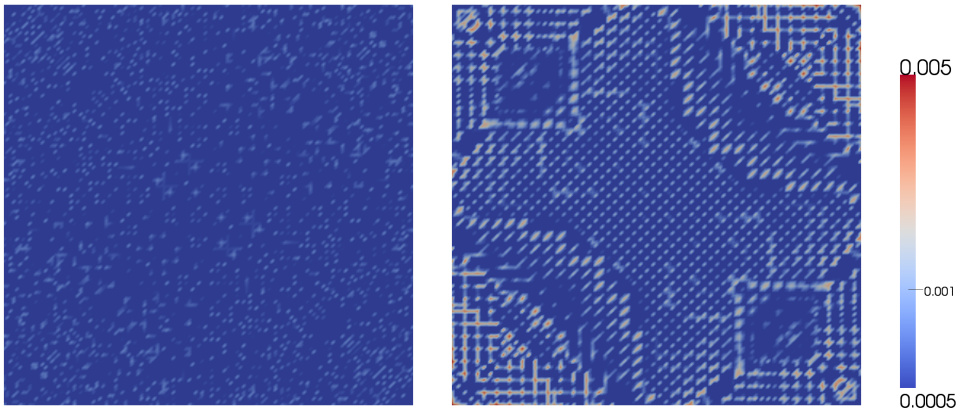

Figure 7.5.: Comparison of quantization errors yielding the same H^{-1} error. Left: hierarchical basis. Right: wavelets.

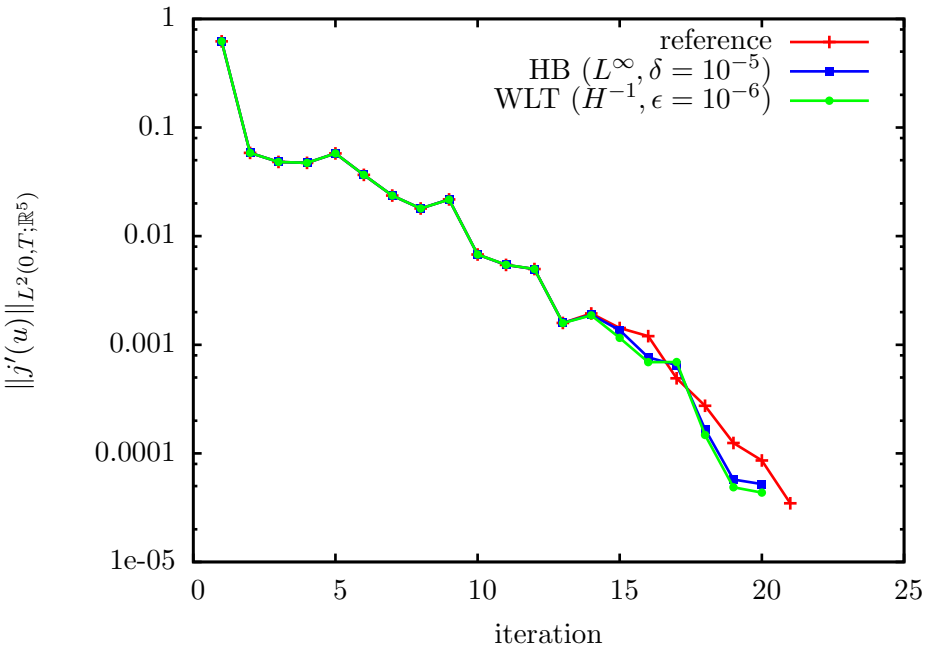

Figure 7.6.: Optimization progress of BFGS for the Kolmogorov example using wavelet and hierarchical basis compression.

8. Conclusion

In this thesis, a computationally inexpensive lossy compression method adapted to the specific needs of parabolic optimal control problems has been introduced and analyzed. While lossy compression techniques are common tools in, e.g., computer graphics and geometry processing, they are, up to now, rarely encountered in PDE-constrained optimization.

All algorithms developed here work on unstructured, adaptively refined grids in two and three space dimensions, and can be efficiently implemented. For keeping a pointwise bound on the quantization error, a-priori estimates for the achievable compression factors have been derived. Due to the inexact reconstruction of the state trajectories, and thus inexact data for the adjoint equation, the error induced in the reduced gradient, and reduced Hessian, has to be controlled, to not impede convergence of the optimization. In this work, accuracy requirements of three exemplary optimization methods have been analyzed. Derivation of error representations and computable error estimates for the influence of lossy trajectory storage allow to control the accuracy of the compressed data adaptively during the progress of the optimization. Going beyond pointwise error control, wavelet-based compression has been presented, allowing to control the quantization error in norms other than L^∞.

The efficiency of the algorithms has been demonstrated on several numerical examples, ranging from a simple linear, scalar equation to a semi-linear system of reaction-diffusion equations, modeling cardiac defibrillation. In all these examples, significant reductions of storage space and memory bandwidth were achieved.

The tools and analysis presented in this dissertation can serve as one ingredient for the adaptive solution of real-world application problems, allowing adaptive control of storage requirements, in addition to more commonly used adaptive control of discretization- and iteration errors. While adaptive quantization has been developed for gradient computation, it can easily be extended to other post-processing applications, like visualization or data analysis.

A. A Comparison Theorem

In this section we briefly present a comparison principle for classical solutions to semi-linear systems of m reaction-diffusion equations

$$\begin{aligned} \frac{\partial y}{\partial t} - D\nabla \cdot (\sigma \nabla y) &= f(y) \quad \text{in } \Omega \times (0,T) \\ B\partial_\nu y + Cy &= 0 \quad \text{on } \partial\Omega \times (0,T) \\ y(\cdot,0) &= y_0 \quad \text{in } \Omega. \end{aligned} \tag{RDS'}$$

In the following, for vectors $y, z \in \mathbb{R}^m$, $y \geq z$ is defined as $y_i \geq z_i \ \forall i = 1, \ldots, m$. Other relations etc. are also defined component-wise. For better readability, often we do not state dependence of functions on (x,t), e.g. $f(x,t,y)$ is abbreviated by $f(y)$.

Definition A.1.

1. A function $\underline{y}$ is a *sub-solution* to (RDS'), if in the differential equations, initial- and boundary conditions "$\leq$" holds instead of "$=$". $\overline{y}$ is a *super-solution*, if "$\geq$" holds instead of "$=$".
2. A function $f : \mathbb{R}^m \to \mathbb{R}^m$ is called *quasi-monotone non-decreasing*, if each component $f_i(y)$ is non-decreasing in y_j for each $i \neq j$.
3. A function $\overline{f} : \mathbb{R}^m \to \mathbb{R}^m$ is called *super-reaction function*, if the inequality $\overline{f}(y) \geq f(y) \ \forall y \in \mathbb{R}^m$ holds.

If the reaction term $f(y)$ in (RDS') is not quasi-monotone non-decreasing, for example in the monodomain equations (6.5), there is no comparison principle available for error estimation. A remedy is the construction of a super-reaction function for the error equations, and using the fact that the solution of the original equation is a sub-solution for the modified system.

For use in Chapter 5, we state the following theorem, lemma and corollary, see, e.g., Fife and Tang [34], and Britton [10].

Theorem A.2. *Let $\underline{y}, \overline{y}$ be a sub- respectively super-solution to* (RDS'). *Assume f is uniformly Lipschitz continuous in y and is quasi-monotone non-decreasing. Then $\underline{y} \leq \overline{y}$ in $\overline{\Omega} \times [0,T]$.*

Lemma A.3. *Let $\overline{f}$ be defined by*

$$\overline{f}_i(y) = \sup_{\{z | \underline{y} \leq z \leq y,\ z_i = y_i\}} f_i(z)$$

for some sub-solution $\underline{y}$. Then $\overline{f}(y) \geq g(y)\ \forall y$, $\overline{f}$ is uniformly Lipschitz continuous, provided f is and is quasi-monotone non-decreasing.

Corollary A.4. *Let $\overline{f}$ be a quasi-monotone non-decreasing, uniformly Lipschitz continuous super-reaction function. Let $\overline{y}$ be a super-solution of the problem* (RDS) *with f replaced by $\overline{f}$, and y a sub-solution of* (RDS'). *Then $y \leq \overline{y}$ in $\overline{\Omega} \times [0, T]$.*

Remark A.5. In the scalar case $m = 1$ the required quasi-monotonicity is trivially fulfilled.

Bibliography

[1] JavaView homepage. `www.javaview.de`.

[2] R. A. Adams. *Sobolev Spaces*. Cambridge University Press, 2001.

[3] P. Alliez and C. Gotsman. Recent advances in compression of 3D meshes. In N. Dodgson, M. Floater, and M. Sabin, editors, *Advances in Multiresolution for Geometric Modelling*, Mathematics and Visualization, pages 3–26. Springer Berlin Heidelberg, 2005.

[4] K. Amaratunga and J. E. Castrillón-Candás. Surface wavelets: a multiresolution signal processing tool for 3D computational modelling. *Int. J. Numer. Meth. Engng.*, 52(3):239–271, 2001.

[5] R. Becker, D. Meidner, and B. Vexler. Efficient numerical solution of parabolic optimization problems by finite element methods. *Optim. Methods Softw.*, 22(5):813–833, 2007.

[6] R. Becker and R. Rannacher. An optimal control approach to a posteriori error estimation in finite element methods. *Acta Numer.*, 10(1):1–102, 2001.

[7] W. R. Bennett. Spectra of quantized signals. *Bell Sys. Tech. J.*, 27(3):446–472, 1948.

[8] A. Borzí and V. Schulz. *Computational Optimization of Systems Governed by Partial Differential Equations*. Computational Science and Engineering. SIAM, Philadelphia, 2012.

[9] D. Braess. *Finite elements: Theory, Fast Solvers, and Applications in Solid Mechanics*. Cambridge University Press, 2001.

[10] N. F. Britton. *Reaction-Diffusion Equations and Their Application to Biology*. Academic Press, 1986.

[11] M. Burtscher and P. Ratanaworabhan. FPC: A high-speed compressor for double-precision floating-point data. *IEEE Trans. Comput.*, 58(1):18–31, 2009.

[12] J. M. Carnicer, W. Dahmen, and J. M. Peña. Local decomposition of refinable spaces and wavelets. *Appl. Comput. Harmon. Anal.*, 3(2):127–153, 1996.

[13] T. Carraro, M. Geiger, and R. Rannacher. Indirect multiple shooting for nonlinear parabolic optimal control problems with control constraints. *SIAM J. Sci. Comput.*, 36(2):A452–A481, 2014.

[14] J. E. Castrillón-Candás. *Spatially adaptive multiwavelet representations on unstructured grids with applications to multidimensional computational modeling.* PhD thesis, Massachusetts Institute of Technology, 2001.

[15] P. G. Ciarlet. *The finite element method for elliptic problems.* North-Holland, Amsterdam, 1978.

[16] A. Cohen, L. M. Echeverry, and Q. Sun. Finite element wavelets. Technical report, Université Pierre et Marie Curie, Paris, 2000.

[17] A. Comas. *Time–Domain Decomposition Preconditioners for the Solution of Discretized Parabolic Optimal Control Problems.* PhD thesis, Rice University, 2005.

[18] W. Dahmen. Wavelet and multiscale methods for operator equations. *Acta Numer.*, 6:55–228, 1997.

[19] W. Dahmen. Wavelet methods for PDEs – some recent developments. *J. Comput. Appl. Math.*, 128(1):133–185, 2001.

[20] W. Dahmen and R. Stevenson. Element-by-element construction of wavelets satisfying stability and moment conditions. *SIAM J. Numer. Anal.*, 37(1):319–352, 1999.

[21] J. E. Dennis, Jr. and J. J. Moré. Quasi-Newton methods, motivation and theory. *SIAM Rev.*, 19(1):46–89, 1977.

[22] P. Deuflhard. Recent progress in extrapolation methods for ordinary differential equations. *SIAM Rev.*, 27(4):505–535, 1985.

[23] P. Deuflhard. *Newton Methods for Nonlinear Problems: Affine Invariance and Adaptive Algorithms.* Springer, 2nd edition, 2006.

[24] P. Deuflhard and F. Bornemann. *Scientific computing with ordinary differential equations*, volume 42. Springer, 2002.

[25] P. Deuflhard, E. Hairer, and J. Zugck. One-step and extrapolation methods for differential-algebraic systems. *Numer. Math.*, 51(5):501–516, 1987.

[26] P. Deuflhard, P. Leinen, and H. Yserentant. Concepts of an adaptive hierarchical finite element code. *IMPACT Comp. Sci. Eng.*, 1(1):3–35, 1989.

[27] P. Deuflhard and U. Nowak. Extrapolation integrators for quasilinear implicit ODEs. In P. Deuflhard and B. Engquist, editors, *Large Scale Scientific Computing*, volume 7 of *Progress in Scientific Computing*, pages 37–50. Birkhäuser, 1987.

[28] P. Deuflhard and M. Weiser. Local inexact Newton multilevel FEM for nonlinear elliptic problems. In M.-O. Bristeau, G. Etgen, W. Fitzgibbon, J.-L. Lions, J. Periaux, and M. Wheeler, editors, *Computational science for the 21st century*, pages 129–138. Wiley, 1997.

[29] P. Deuflhard and M. Weiser. *Adaptive numerical solution of PDEs*. de Gruyter, 2012.

[30] R. A. DeVore. Adaptive wavelet bases for image compression. In P. J. Laurent, A. L. Méhauté, and L. L. Schumaker, editors, *Wavelets, Images, and Surface Fitting*, pages 197–219. A K Peters, 1994.

[31] R. Ehrig, U. Nowak, L. Oeverdieck, and P. Deuflhard. Advanced extrapolation methods for large scale differential algebraic problems. In H.-J. Bungartz, F. Durst, and C. Zenger, editors, *High Performance Scientific and Engineering Computing*, volume 8 of *Lecture Notes in Computational Science and Engineering*, pages 233–241. Springer Berlin Heidelberg, 1999.

[32] A. Ern and J.-L. Guermond. *Theory and Practice of Finite Elements*. Springer, New York, 2004.

[33] U. Felgenhauer. On the stable global convergence of particular quasi-Newton-methods. *Optimization*, 26:97–113, 1992.

[34] P. C. Fife and M. M. Tang. Comparison principles for reaction-diffusion systems: Irregular comparison functions and applications to questions of stability and speed of propagation of disturbances. *J. Differential Equations*, 40(2):168–185, 1981.

[35] P. C. Franzone, P. Deuflhard, B. Erdmann, J. Lang, and L. F. Pavarino. Adaptivity in space and time for reaction-diffusion systems in electrocardiology. *SIAM J. Numer. Anal.*, 28(3):942–962, 2006.

[36] B. Goeman, H. Vandierendonck, and K. De Bosschere. Differential FCM: Increasing value prediction accuracy by improving table usage efficiency. In *High-Performance Computer Architecture, 2001. HPCA. The Seventh International Symposium on*, pages 207–216. IEEE, 2001.

[37] S. Götschel, N. Chamakuri, K. Kunisch, and M. Weiser. Lossy compression in optimal control of cardiac defibrillation. *J. Sci. Comput.*, 60(1):35–59, 2014.

[38] S. Götschel, C. von Tycowicz, K. Polthier, and M. Weiser. Reducing memory requirements in scientific computing and optimal control. In T. Carraro, M. Geiger, S. Körkel, and R. Rannacher, editors, *Multiple Shooting and Time Domain Decomposition Methods*, Contributions in Mathematical and Computational Sciences. Springer, 2015. to appear.

[39] S. Götschel and M. Weiser. Lossy compression for PDE-constrained optimization: Adaptive error control. *Comput. Optim. Appl.*, 2014. online first.

[40] S. Götschel, M. Weiser, and A. Schiela. Solving optimal control problems with the Kaskade 7 finite element toolbox. In A. Dedner, B. Flemisch, and R. Klöfkorn, editors, *Advances in DUNE*, pages 101–112. Springer, 2012.

[41] S. Gratton, P. L. Toint, and A. Tröltzsch. How much gradient noise does a gradient-based linesearch method tolerate? NAXYS Technical Report 04-2012, Department of Mathematics, Namur Center for Complex Systems, CERFACS, ENSEEIHT-IRIT, 2012.

[42] R. M. Gray and D. L. Neuhoff. Quantization. *IEEE Trans. Inform. Theory*, 44(6):2325–2383, 1998.

[43] A. Greenbaum. Estimating the attainable accuracy of recursively computed residual methods. *SIAM J. Matrix Anal. Appl.*, 18(3):535–551, 1997.

[44] A. Griewank. Rates of convergence for secant methods on nonlinear problems in Hilbert space. In J.-P. Hennart, editor, *Numerical Analysis*, volume 1230 of *Lecture Notes in Mathematics*, pages 138–157. Springer Berlin Heidelberg, 1986.

[45] A. Griewank. Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation. *Optim. Methods Softw.*, 1(1):35–54, 1992.

[46] A. Griewank and A. Walther. Algorithm 799: revolve: an implementation of checkpointing for the reverse or adjoint mode of computational differentiation. *ACM Trans. Math. Software*, 26(1):19–45, 2000.

[47] A. Griewank and A. Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, Philadelphia, 2008.

[48] M. H. Gutknecht and Z. Strakoš. Accuracy of two three-term and three two-term recurrences for Krylov space solvers. *SIAM J. Matrix Anal. Appl.*, 22(1):213–229, 2000.

[49] M. Heinkenschloss. A time-domain decomposition iterative method for the solution of distributed linear quadratic optimal control problems. *J. Comput. Appl. Math.*, 173(1):169–198, 2005.

[50] M. Heinkenschloss and L. Vicente. Analysis of inexact trust-region SQP algorithms. *SIAM J. Optim.*, 12(2):283–302, 2002.

[51] H. K. Hesse. *Multiple Shooting and Mesh Adaptation for PDE Constrained Optimization Problems.* PhD thesis, University Heidelberg, 2008.

[52] H. K. Hesse and G. Kanschat. Mesh adaptive multiple shooting for partial differential equations. part I: linear quadratic optimal control problems. *J. Numer. Math.*, 17(3):195–217, 2009.

[53] V. Heuveline and A. Walther. Online checkpointing for parallel adjoint computation in PDEs: Application to goal-oriented adaptivity and flow control. In *Euro-Par 2006 Parallel Processing*, pages 689–699. Springer, 2006.

[54] M. Hinze and K. Kunisch. Second order methods for optimal control of time-dependent fluid flow. *SIAM J. Control Optim.*, 40(3):925–946, 2001.

[55] M. Hinze, R. Pinnau, M. Ulbrich, and S. Ulbrich. *Optimization with PDE constraints.* Springer, Berlin, 2009.

[56] M. Hinze and J. Sternberg. A-revolve: an adaptive memory-reduced procedure for calculating adjoints; with an application to computing adjoints of the instationary Navier-Stokes system. *Optim. Methods Softw.*, 20(6):645–663, 2005.

[57] M. Hinze and S. Volkwein. Error estimates for abstract linear–quadratic optimal control problems using proper orthogonal decomposition. *Comput. Optim. Appl.*, 39:319–345, 2008.

[58] P. K. Hoh Phua. Eigenvalues and switching algorithms for quasi-Newton updates. *Optimization*, 42(3):185–217, 1997.

[59] D. A. Hooks, M. L. Trew, B. J. Caldwell, G. B. Sands, I. J. LeGrice, and B. H. Smaill. Laminar arrangement of ventricular myocytes influences electrical behavior of the heart. *Circ. Res.*, 101(10):e103–12, 2007.

[60] L. Ibarria, P. Lindstrom, J. Rossignac, and A. Szymczak. Out-of-core compression and decompression of large n-dimensional scalar fields. In *Computer Graphics Forum*, volume 22, pages 343–348. Wiley Online Library, 2003.

[61] L. Ibarria and J. Rossignac. Dynapack: space-time compression of the 3D animations of triangle meshes with fixed connectivity. *ACM SIGGRAPH/Eurographics Symposium on Computer animation*, pages 126–135, 2003.

[62] M. Isenburg and J. Snoeyink. Mesh collapse compression. In *In Proceedings of SIBGRAPI'99*, pages 27–28, 1999.

[63] M. Isenburg and J. Snoeyink. Early-split coding of triangle mesh connectivity. In *Graphics Interface Conference Proceedings*, pages 89–97, Toronto, Ont., Canada, Canada, 2006. Canadian Information Processing Society.

[64] K. Ito and K. Kunisch. Receding horizon optimal control for infinite dimensional systems. *ESAIM Control Optim. Calc. Var.*, 8(1):741–760, 2002.

[65] K. Ito and K. Kunisch. *Lagrange Multiplier Approach to Variational Problems and Applications*. Advances in Design and Control. SIAM, 2008.

[66] J. Iverson, C. Kamath, and G. Karypis. Fast and effective lossy compression algorithms for scientific datasets. In *Euro-Par 2012 Parallel Processing*, pages 843–856. Springer, 2012.

[67] C. Jörres, G. Vossen, and M. Herty. On an inexact gradient method using proper orthogonal decomposition for parabolic optimal control problems. *Comput. Optim. Appl.*, 55(2):459–468, 2013.

[68] F. Kälberer, K. Polthier, U. Reitebuch, and M. Wardetzky. Freelence – coding with free valences. *Computer Graphics Forum*, 24(3):469–478, 2005.

[69] F. Kälberer, K. Polthier, and C. von Tycowicz. Lossless compression of adaptive multiresolution meshes. In *Proc. Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI)*, volume 22, 2009.

[70] E. Kammann, F. Tröltzsch, and S. Volkwein. A method of a-posteriori error estimation with application to proper orthogonal decomposition. *ESAIM: M2AN*, 47:555–581, 2013.

[71] C. T. Kelley and E. W. Sachs. Quasi-Newton methods and unconstrained optimal control problems. *SIAM J. Control Optim.*, 25(6):1503–1516, 1987.

[72] C. T. Kelley and E. W. Sachs. Approximate quasi-Newton methods. *Math. Program.*, 48(1-3):41–70, 1990.

[73] A. Khodakovsky, P. Schröder, and W. Sweldens. Progressive geometry compression. In *SIGGRAPH '00 Proceedings*, pages 271–278, 2000.

[74] K. Kunisch, C. Nagaiah, and M. Wagner. A parallel Newton-Krylov method for optimal control of the monodomain model in cardiac electrophysiology. *Comput. Vis. Sci.*, 14(6):257–269, 2011.

[75] K. Kunisch and S. Volkwein. Galerkin proper orthogonal decomposition methods for parabolic problems. *Numer. Math.*, 90:117–148, 2001.

[76] K. Kunisch and M. Wagner. Optimal control of the bidomain system (I): The monodomain approximation with the Rogers–McCulloch model. *Nonlinear Anal. Real World Appl.*, 13(4):1525–1550, 2012.

[77] F.-S. Kupfer. An infinite-dimensional convergence theory for reduced SQP methods in Hilbert space. *SIAM J. Optim.*, 6(1):126–163, 1996.

[78] S. Lakshminarasimhan, N. Shah, S. Ethier, S. Klasky, R. Latham, R. Ross, and N. F. Samatova. Compressing the incompressible with ISABELA: In-situ reduction of spatio-temporal data. In *Euro-Par 2011 Parallel Processing*, pages 366–379. Springer, 2011.

[79] S. Lakshminarasimhan, N. Shah, S. Ethier, S.-H. Ku, C.-S. Chang, S. Klasky, R. Latham, R. Ross, and N. F. Samatova. ISABELA for effective in situ compression of scientific data. *Concurrency Computat.: Pract. Exper.*, 25:524–540, 2013.

[80] H. Lee, P. Alliez, and M. Desbrun. Angle-Analyzer: A triangle-quad mesh codec. *Computer Graphics Forum*, 21:383–392, 2002.

[81] J. E. Lengyel. Compression of time-dependent geometry. *ACM Symposium on Interactive 3D Graphics*, pages 89–95, 1999.

[82] P. Lindstrom and M. Isenburg. Fast and efficient compression of floating-point data. *IEEE Trans. Visual. Comput. Graphics*, 12(5):1245–1250, 2006.

[83] T.-W. Liu. A regularized limited memory BFGS method for nonconvex unconstrained minimization. *Numer. Algorithms*, 65(2):305–323, 2014.

[84] M. Lounsbery, T. D. DeRose, and J. Warren. Multiresolution analysis for surfaces of arbitrary topological type. *ACM Trans. Graph.*, 16:34–73, 1997.

[85] C. Lubich and M. Roche. Rosenbrock methods for differential-algebraic systems with solution-dependent singular matrix multiplying the derivative. *Computing*, 43(4):325–342, 1990.

[86] K. Mamou, T. Zaharia, and F. Prêteux. FAMC: The MPEG-4 standard for animated mesh compression. *IEEE International Conference on Image Processing*, pages 2676–2679, 2008.

[87] G. N. N. Martin. Range encoding: an algorithm for removing redundancy from a digitised message. Presented at Video & Data Recording Conference, Southampton, 1979.

[88] H. J. Martínez, Z. Parada, and R. A.Tapia. On the characterization of q-superlinear convergence of quasi-Newton interior-point methods for nonlinear programming. *Bol. Soc. Mat. Mexicana (3)*, 1, 1995.

[89] H. Maurer and J. Zowe. First and second-order necessary and sufficient optimality conditions for infinite-dimensional programming problems. *Math. Program.*, 16(1):98–110, 1979.

[90] R. V. Mayorga and V. H. Quintana. A family of variable metric methods in function space, without exact line searches. *J. Optim. Theory Appl.*, 31(3):303–329, 1980.

[91] D. Meidner and B. Vexler. Adaptive space-time finite element methods for parabolic optimization problems. *SIAM J. Control Optim.*, 46:116–142, March 2007.

[92] J. L. Mitchell, W. B. Pennebaker, C. E. Fogg, and D. J. LeGall, editors. *MPEG video compression standard*. Chapmann & Hall, New York, 1997.

[93] C. Nagaiah and K. Kunisch. Higher order optimization and adaptive numerical solution for optimal control of monodomain equations in cardiac electrophysiology. *Appl. Numer. Math.*, 61:53–65, 2011.

[94] C. Nagaiah, K. Kunisch, and G. Plank. Numerical solution for optimal control of the reaction-diffusion equations in cardiac electrophysiology. *Comput. Optim. Appl.*, 49:149–178, 2011. 10.1007/s10589-009-9280-3.

[95] H. Nguyen and R. Stevenson. Finite element wavelets with improved quantitative properties. *J. Comput. Appl. Math.*, 230(2):706–727, 2009.

[96] B. F. Nielsen, T. S. Ruud, G. T. Lines, and A. Tveito. Optimal monodomain approximations of the bidomain equations. *Appl. Math. Comput.*, 184(2):276–290, 2007.

[97] J. Nocedal. Updating quasi-Newton matrices with limited storage. *Math. Comp.*, 35(151):773–782, 1980.

[98] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, 2006.

[99] U. Nowak. A fully adaptive MOL-treatment of parabolic 1-D problems with extrapolation techniques. *Appl. Numer. Math.*, 20:129–141, 1996.

[100] M. H. Protter and H. F. Weinberger. *Maximum principles in differential equations*. Springer, New York, corr. reprint of the second edition, 1999.

[101] M. Rettenmeier. *Data compression for computational fluid dynamics on irregular grids*. PhD thesis, Universität zu Köln, 2012.

[102] I. E. G. Richardson. *Video codec design*. Wiley, Chichester, 2002.

[103] J. Rissanen and G. G. Langdon Jr. Arithmetic coding. *IBM J. Res. Dev.*, 23(2):149–162, 1979.

[104] J. M. Rogers and A. D. McCulloch. A collocation-Galerkin finite element model of cardiac action potential propagation. *IEEE Trans. Biomed. Eng.*, 41:743–757, 1994.

[105] J. Rossignac. Edgebreaker: Connectivity compression for triangle meshes. *IEEE Trans. Visual. Comput. Graphics*, 5(1):47–61, 1999.

[106] E. W. Sachs. Broyden's method in Hilbert space. *Math. Program.*, 35(1):71–82, 1986.

[107] Y. Sazeides and J. E. Smith. The predictability of data values. In *Microarchitecture, 1997. Proceedings., Thirtieth Annual IEEE/ACM International Symposium on*, pages 248–258. IEEE, 1997.

[108] A. Schiela and A. Günther. An interior point algorithm with inexact step computation in function space for state constrained optimal control. *Numer. Math.*, 119(2):373–407, 2011.

[109] P. Schröder and W. Sweldens. Spherical wavelets: Efficiently representing functions on the sphere. In *SIGGRAPH '95 Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 161–172. ACM, 1995.

[110] F. Schröder-Pander, T. Sonar, and O. Friedrich. Generalized multiresolution analysis on unstructured grids. *Numer. Math.*, 86(4):685–715, 2000.

[111] T. M. Shafaat and S. B. Baden. A method of adaptive coarsening for compressing scientific datasets. In B. Kågström, E. Elmroth, J. Dongarra, and J. Wasniewski, editors, *Applied Parallel Computing. State of the Art in Scientific Computing. 8th International Workshop, PARA 2006, Umeå, Sweden, June 18-21, 2006, Revised Selected Papers*, volume 4699 of *Lecture Notes in Computer Science*, pages 774–780. Springer, 2007.

[112] C. E. Shannon. A mathematical theory of communication. *Bell Sys. Techn. J.*, 27, 1948.

[113] J. M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Trans. Signal Process.*, 41:3445–3462, 1993.

[114] V. Simoncini and D. B. Szyld. Theory of inexact Krylov subspace methods and applications to scientific computing. *SIAM J. Sci. Comput.*, 25(2):454–477, 2003.

[115] G. L. G. Sleijpen and H. A. van der Vorst. Reliable updated residuals in hybrid Bi-CG methods. *Computing*, 56(2):141–163, 1996.

[116] N. Stefanoski and J. Ostermann. Spatially and temporally scalable compression of animated 3D meshes with MPEG-4/FAMC. *IEEE International Conference on Image Processing*, pages 2696–2699, 2008.

[117] J. Sternberg and M. Hinze. A memory-reduced implementation of the Newton-CG method in optimal control of nonlinear time-dependent PDEs. *Optim. Methods Softw.*, 25(4):553–571, 2010.

[118] R. Stevenson. Locally supported, piecewise polynomial biorthogonal wavelets on nonuniform meshes. *Constr. Approx.*, 19(4):477–508, 2003.

[119] P. Stumm and A. Walther. Multi-stage approaches for optimal offline checkpointing. *SIAM J. Sci. Comput.*, 31(3):1946–1967, 2009.

[120] P. Stumm and A. Walther. New algorithms for optimal online checkpointing. *SIAM J. Sci. Comput.*, 32(1):836–854, 2010.

[121] G. J. Sullivan and T. Wiegand. Video compression – from concepts to the H.264/AVC standard. *Proceedings of the IEEE*, 93(1):18–31, 2005.

[122] W. Sweldens. The lifting scheme: A construction of second generation wavelets. *SIAM J. Math. Anal.*, 29(2):511–546, 1998.

[123] A. Szymczak. Optimized Edgebreaker encoding for large and regular triangle meshes. In *DCC '02 Proceedings*, page 472, Washington, DC, USA, 2002. IEEE Computer Society.

[124] R. I. Teran, C.-A. Thole, and R. Lorentz. New developments in the compression of LS-DYNA simulation results using FEMZIP. 6th European LS-DYNA Users' Conference, 2007.

[125] C.-A. Thole. Compression of LS-DYNA3D^{TM} simulation results using FEMZIP©. 3. LS-DYNA Anwenderforum, 2004.

[126] C. Touma and C. Gotsman. Triangle mesh compression. In *Graphics Interface Conference Proceedings*, pages 26–34, 1998.

[127] F. Tröltzsch. *Optimal control of partial differential equations: Theory, methods and applications*, volume 112 of *Graduate Studies in Mathematics*. AMS, 2010.

[128] F. Tröltzsch and S. Volkwein. POD a-posteriori error estimates for linear-quadratic optimal control problems. *Comput. Optim. Appl.*, 44:83–115, 2009.

[129] D. Unat, T. Hromadka, and S. B. Baden. An adaptive sub-sampling method for in-memory compression of scientific data. In *Data Compression Conference, 2009. DCC '09*, pages 262–271. IEEE, 2009.

[130] J. van den Eshof and G. L. G. Sleijpen. Inexact Krylov subspace methods for linear systems. *SIAM J. Matrix Anal. Appl.*, 26(1):125–153, 2004.

[131] H. A. van der Vorst and Q. Ye. Residual replacement strategies for Krylov subspace iterative methods for the convergence of true residuals. *SIAM J. Sci. Comput.*, 22(3):835–852, 2000.

[132] P. S. Vassilevski and J. Wang. Stabilizing the hierarchical basis by approximate wavelets, I: Theory. *Numer. Linear Algebra Appl.*, 4(2):103–126, 1997.

[133] P. S. Vassilevski and J. Wang. Stabilizing the hierarchical basis by approximate wavelets II: Implementation and numerical results. *SIAM J. Sci. Comput.*, 20(2):490–514, 1998.

[134] L. Váša. Optimised mesh traversal for dynamic mesh compression. *Graphical Models*, 73:218–230, 2011.

[135] Y. M. Volin and G. M. Ostrovskii. Automatic computation of derivatives with the use of the multilevel differentiating techniques—1. algorithmic basis. *Comput. Math. Appl.*, 11(11):1099–1114, 1985.

[136] C. von Tycowicz, F. Kälberer, and K. Polthier. Context-based coding of adaptive multiresolution meshes. *Computer Graphics Forum*, 30(8):2231–2245, 2011.

[137] G. von Winckel and A. Borzì. Computational techniques for a quantum control problem with H^1-cost. *Inverse Problems*, 24(3):034007, 2008.

[138] A. Walther. *Program reversal schedules for single-and multi-processor machines*. PhD thesis, Institute of Scientific Computing, Technical University Dresden, Germany, 1999.

[139] Q. Wang, P. Moin, and G. Iaccarino. Minimal repetition dynamic checkpointing algorithm for unsteady adjoint calculation. *SIAM J. Sci. Comput.*, 31(4):2549–2567, 2009.

[140] M. Weiser. On goal-oriented adaptivity for elliptic optimal control problems. *Optim. Methods Softw.*, 28(5):969–992, 2013.

[141] M. Weiser and S. Götschel. State trajectory compression for optimal control with parabolic PDEs. *SIAM J. Sci. Comput.*, 34(1):A161–A184, 2012.

[142] H. Yserentant. On the multi-level splitting of finite element spaces. *Numer. Math.*, 49(4):379–412, 1986.

[143] E. Zeidler. *Nonlinear Functional Analysis and its Applications I: Fixed-Point Theorems*. Springer, New York, 1986.

[144] E. Zeidler. *Nonlinear Functional Analysis and its Applications II/A: Linear Monotone Operators*. Springer, New York, 1990.

[145] J. C. Ziems. Adaptive multilevel inexact SQP-methods for PDE-constrained optimization with control constraints. *SIAM J. Optim.*, 23(2):1257–1283, 2013.

[146] J. C. Ziems and S. Ulbrich. Adaptive multilevel inexact SQP methods for PDE-constrained optimization. *SIAM J. Optim.*, 21(1):1–40, 2011.

Acknowledgments

First of all, I wish to thank Dr. Martin Weiser, head of the Computational Medicine group and of the Numerical Analysis and Modelling department at the Zuse Institute Berlin (ZIB), for introducing me to the field of data compression for optimal control problems, and for many constructive discussions. Without his constant support and guidance this work would not have been possible.

I am grateful to my advisor Prof. Dr. Dr. h.c. Peter Deuflhard, FU Berlin and former president of ZIB, for giving me the opportunity to work in an inspiring environment and his continued interest in my work.

Also, I want to thank Prof. Dr. Matthias Heinkenschloss, Rice University, for the readiness to act as the second referee for this thesis. I am grateful to Dr. Chamakuri Nagaiah, RICAM Linz, and Prof. Dr. Karl Kunisch, University of Graz, for their interest in my work and the fruitful collaboration.

I acknowledge, with thanks, partial funding by the DFG research center MATHEON "Mathematics for key technologies", projects C25/F9. Thanks also to Prof. Dr. Konrad Polthier and Dr. Christoph von Tycowicz, FU Berlin, for helpful discussions and the collaboration in these projects.

Special thanks go to all my former and present colleagues at ZIB for their company, many interesting discussions, and for creating a pleasant work environment.

Last, but not least, a huge thanks to my friends and family, especially to my wife Martina, for their everlasting support, encouragement, and patience.

Zusammenfassung

Optimalsteuerungsprobleme mit parabolischen partiellen Differentialgleichungen als Nebenbedingung werden häufig in ein unrestringiertes Optimierungsproblem mit reduziertem Zielfunktional überführt. Zur Berechnung des reduzierten Gradienten muss eine adjungierte Gleichung gelöst werden. Diese ist eine Rückwärtsgleichung, für die die zuvor berechnete Lösung der Zustandsgleichung benötigt wird. Bei hohen Anforderungen an die Diskretisierungsgenauigkeit fällt dafür ein hoher Speicherbedarf an. Die vorliegende Arbeit befasst sich mit der Entwicklung und Analyse von Verfahren zur verlustbehafteten Kompression solcher Finite-Elemente-Lösungen.

Die entwickelten Methoden verwenden einen Basiswechsel, um Korrelationen in den zu speichernden Daten zu reduzieren, sowie Quantisierung, welche die Genauigkeit der Daten verringert. Für den grundlegenden Algorithmus wird die Transformation von Knoten- zu Hierarchischer Basis verwendet, und anschließend die Koeffizienten auf die gewünschte Präzision gerundet.

Ein Schwerpunkt der Arbeit liegt auf der adaptiven Wahl der erforderlichen Genauigkeit, um den Verlauf der Optimierung nicht zu beeinträchtigen. Dafür werden berechenbare Fehlerabschätzungen sowie Kriterien zur Wahl der Quantisierungstoleranz für verschiedene Optimierungsverfahren hergeleitet. Während für Gradienten- und Quasi-Newton-Verfahren nur der Fehler im reduzierten Gradienten von Bedeutung ist, muss bei Newton-CG-Verfahren berücksichtigt werden, dass Matrix-Vektor-Produkte während des CG-Verfahrens nur inexakt berechnet werden können. Mittels Fehlerverfolgung und rechtzeitiger Neuberechnung des Residuums kann verhindert werden dass der Algorithmus vorzeitig abbricht.

Die entwickelten Verfahren werden an verschiedenen Bespielen getestet. In allen numerischen Experimenten kann durch adaptive Wahl der Genauigkeit erreicht werden, dass trotz verlustbehafteter Kompression keine signifikante Abweichung im Konvergenzverhalten der Optimierungsverfahren zu beobachten ist.

Um über punktweise Fehlerkontrolle hinausgehen zu können, wird die Transformation auf die Hierarchische Basis durch eine Wavelet-Transformation ersetzt. Hierfür werden effizient implementierbare Wavelet-Konstruktionen vorgestellt. Numerische Experimente belegen, dass durch Fehlerkontrolle in der passenden Norm deutlich verbesserte Kompressionsfaktoren erreicht werden können.