Lossy Compression for PDE-constrained Optimization: Adaptive Error Control

Sebastian Götschel* Martin Weiser*

December 3, 2013

Abstract

For the solution of optimal control problems governed by nonlinear parabolic PDEs, methods working on the reduced objective functional are often employed to avoid a full spatio-temporal discretization of the problem. 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 which are accurate enough, in many case very fine discretizations in time and space are necessary, which leads to a significant amount of data to be stored and transmitted to mass storage. Lossy compression methods were developed to overcome the storage problem by reducing the accuracy of the stored trajectories. The inexact data induces errors in the reduced gradient and reduced Hessian. In this paper, we analyze the influence of such a lossy trajectory compression method on Newton-CG methods for optimal control of parabolic PDEs and design an adaptive strategy for choosing appropriate quantization tolerances.

Keywords: optimal control, semilinear parabolic PDEs, Newton-CG, trajectory storage, lossy compression

MSC2010: 35K58, 49M15, 65M60, 68P30, 94A29

1 Introduction

Optimal control problems governed by nonlinear, time-dependent PDEs on 3D spatial domains form an important problem-class in many fields, ranging from engineering applications to medicine. For their solution, methods working on the reduced objective functional are often employed to avoid a full spatio-temporal discretization of the problem. 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 which are accurate

*Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany, {goetschel,weiser}@zib.de

enough, in many cases very fine discretizations in time and space are necessary, which leads to a significant amount of data to be stored and transmitted to mass storage. A detailed overview of compression methods can be found in [7].

Checkpointing methods, see e.g. [10, 11, 12] are a popular technique to reduce the storage requirements of such trajectories at the cost of multiple PDE solves. In [27], a computationally inexpensive method for lossy compression has been developed, where the trade-off is between accuracy and storage requirements. There, for linear problems and steepest-descent methods, a technique for adaptively controlling the error introduced by the inexact storage during the optimization iterations was derived. For a medical application, cardiac defibrillation, this method was extended in [6] to a Newton-CG method for solving the optimal control problem governed by the monodomain equations. In that work, for simplicity, the influence of inexact storage of state and adjoint values on the matrix-vector products during CG was neglected.

In this paper, we extend adaptive error control to general semi-linear systems of reaction-diffusion equations, and analyze Newton-CG methods in a rigorous way, providing adaptive accuracy requirements and error control for state, adjoint and linearized-state solutions. The behavior of conjugate gradient methods in finite precision arithmetic is well-studied, e.g. [9, 13, 23, 16, 15]. While the errors induced by lossy compression are significantly larger, the ideas developed there can be re-used. Especially, inexact storage of solution trajectories leads to inexact matrix-vector products, with the error varying in each CG iteration. Krylov methods with inexact matrix-vector products were investigated in detail for example in [21, 24, 1, 4].

This paper is organized as follows. In Sec. 2 we describe the problem setting, introduce the basic ingredients of our lossy compression approach, and state comparison principles used later in the derivation of error bounds. In Sec. 3, we derive error bounds and computationally availably estimates. In order not to impede the convergence of the optimization, accuracy requirements are given in Sec. 4. In Sec. 5, we give numerical examples.

2 Preliminaries

In this section we fix the problem setting and standing assumptions. We briefly introduce the lossy compression technique. For error estimation, we require a comparison theorem for the governing PDEs.

2.1 Problem Setting

We consider the abstract optimal control problem

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

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

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

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. 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 $\Omega_c \subset \Omega$, $g_\Omega(u) = \chi_{\Omega_c}(x)u(t)$.

For application of comparison theorems, we require that the spatial domain $\Omega \subset \mathbb{R}^d$ has a sufficiently regular boundary Γ (C^2, or at least satisfying the interior sphere property, see e.g. [5, 2]). Further, we assume that the PDEs possess an at least locally unique solution $y(u)$ for every control $u \in U$. For error estimation we require the functions g_Ω, g_Γ to be monotonely increasing in the control u.

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

$$J(y,u) = J_1(y) + J_2(u).$$

By j we denote the reduced functional, $j(u) = J(y(u),u)$. Further, we assume that J, c are sufficiently smooth. The partial derivatives of the operator c are given 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

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

2.2 Lossy Compression

The lossy compression algorithm consists of two main ingredients: quantization and prediction. For discretization of the PDEs, we consider a time-stepping scheme with, for simplicity of presentation, uniform time steps. At each time step a nested family $\mathcal{T}_0 \subset \cdots \subset \mathcal{T}_l$ of triangulations is used for spatial discretization by linear finite elements, constructed from an initial triangulation $\mathcal{T}_0$ of the domain $\Omega \subset \mathbb{R}^d$. The set of nodes on level j is denoted by $\mathcal{N}_j$.

Quantization. For a given $\delta > 0$, we define the *quantization* $Q_\delta : \mathbb{R} \to \mathbb{Z}$ as

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

the *reconstruction* $Q_\delta^\dagger : \mathbb{Z} \to \mathbb{R}$ is given by

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

This yields for the *quantization error* $|y - Q_\delta^\dagger(Q_\delta(y))| \le \delta$.

Prediction. Values y_k of coarse level nodes $x_k \in \mathcal{N}_0$ are quantized directly to $i_k = Q_\delta(y_k)$, giving a reconstructed value $\hat{y}_k := Q_\delta^\dagger(i_k)$. For new nodes $x_k \in \mathcal{N}_j \backslash \mathcal{N}_{j-1}$ on level $j > 0$, we make use of the grid hierarchy and quantize and store only the deviation of y_k from a prediction $P_k(\hat{y}_l : l \in \mathcal{N}_{j-1})$ obtained from reconstructed values $\hat{y}_l$ of lower level nodes. For simplicity, here we use the usual multigrid prolongation operator, which is just linear interpolation between coarser grid nodes adjacent to x_k. The prediction increases the frequency of small numbers in the coefficients to be stored, such that subsequent entropy coding reduces the amount of data to be stored effectively. More details on the algorithmic procedure can be found in [27].

2.3 A Comparison Theorem

In this section we briefly present a comparison principle for semi-linear systems of reaction-diffusion equations (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 2.1. 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.

With these definitions, we state the following theorem and corollary, see e.g. [5, 2].

Theorem 2.2. *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 $\overline{\Omega} \times [0, T]$.*

Corollary 2.3. *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 2.4. In the scalar case $m = 1$ the required quasi-monotonicity is trivially fulfilled.

3 Error Estimates for Lossy Compression

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. We use the following notations 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. $\tilde{\lambda} = \lambda + e_\lambda$.

3.1 Reduced Gradient

The reduced gradient can be computed via the implicit function theorem:

$$j'(u) = J_u(y,u) + c_u(y,u)^\star \lambda, \tag{1}$$

where λ solves the adjoint equation

$$c_y(y,u)^\star \lambda = -J_y(y,u). \tag{2}$$

Due to compression, only an inexact reduced gradient $\tilde{j'}$ can be computed, with y replaced by its reconstruction $\hat{y}$ in (1) and (2). The inexactly computed adjoint state is denoted by $\tilde{\lambda}$.

Theorem 3.1. *The error in the reduced gradient $e_{j'} = \tilde{j'} - j'$ is given by*

$$e_{j'} = c_u(\hat{y},u)^\star e_\lambda, \tag{3}$$

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

$$\left(c_y(\hat{y},u)^\star - (c_{yy}(\hat{y},u)\varepsilon_y)^\star\right) e_\lambda = -J_{yy}(\hat{y},u)\varepsilon_y - (c_{yy}(\hat{y},u)\varepsilon_y)^\star \tilde{\lambda} \tag{4}$$

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

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

$$c_y(\hat{y},u)^\star \tilde{\lambda} - c_y(y,u)^\star \lambda = -J_y(\hat{y},u) + J_y(y,u). \tag{5}$$

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) 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 (4). As by our general assumptions $J_u(\hat{y},u) = J_u(y,u)$ and $c_u(\hat{y},u) = c_u(y,u)$ the claim follows. □

3.2 Reduced Hessian-Vector Products

To evaluate the action of the reduced Hessian $j''(u)$ on a given 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 adjoint-for-Hessian equation $c_y(y,u)^\star w = z$ for $w \in Z$
4. set $j''(u)\delta u := J_{uu}(y,u)\delta u + c_u(y,u)^\star w$.

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

$$\big(c_y(\hat{y}, u) - c_{yy}(\hat{y}, u)\varepsilon_y\big)e_v = -c_{yy}(\hat{y}, u)\varepsilon_y\tilde{v} \tag{6}$$

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

Proof. Subtracting exact and inexact equation, and using Taylor expansion as in the proof of Thm. 3.1 we get

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

as by assumption 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 3.3. *The error $e_z = \tilde{z} - z$ is given by*

$$\begin{aligned} e_z = {} & \big(J_{yy}(\hat{y}, u) - J_{yyy}(\hat{y}, u)\varepsilon_y\big)(e_v + \varepsilon_v) + J_{yyy}(\hat{y}, u)\varepsilon_y\hat{\tilde{v}} \\ & + \langle\big(c_{yy}(\hat{y}, u) - c_{yyy}(\hat{y}, u)\varepsilon_y\big)(e_v + \varepsilon_v), \hat{\tilde{\lambda}}\rangle + \langle c_{yyy}(\hat{y}, u)\varepsilon_y\hat{\tilde{v}}, \hat{\tilde{\lambda}}\rangle \\ & + \langle\big(c_{yy}(\hat{y}, u) - c_{yyy}(\hat{y}, u)\varepsilon_y\big)\hat{\tilde{v}}, e_\lambda + \varepsilon_\lambda\rangle \\ & - \langle\big(c_{yy}(\hat{y}, u) - c_{yyy}(\hat{y}, u)\varepsilon_y\big)(e_v + \varepsilon_v), e_\lambda + \varepsilon_\lambda\rangle \end{aligned} \tag{7}$$

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

$$\Big(c_y(\hat{y},u)^\star - \big(c_{yy}(\hat{y},u)\varepsilon_y\big)^\star\Big)\, e_w = e_z - \big(c_{yy}(\hat{y},u)\varepsilon_y\big)^\star \tilde{w} \tag{8}$$

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

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

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

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

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

Lemma 3.5. *The error $e_{mv} = \tilde{j''}(u)\delta u - j''(u)\delta u$ in the matrix-vector product is given by*

$$e_{mv} = c_u(\hat{y},u)^\star e_w. \tag{9}$$

Proof. By assumptions, J_{uu}, c_u are independent of y. Subtracting exact and inexact equation shows the lemma. □

3.3 Worst Case 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 (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

$$\begin{aligned}
\|e_z\| \le{}& \|J_{yy}(\hat{y},u) - J_{yyy}(\hat{y},u)\varepsilon_y\|\,\|e_v+\varepsilon_v\| + \|J_{yyy}(\hat{y},u)\varepsilon_y\|\,\big\|\hat{\tilde{v}}\big\| \\
&+ \|c_{yy}(\hat{y},u) - c_{yyy}(\hat{y},u)\varepsilon_y\|\,\|e_v+\varepsilon_v\|\,\big\|\hat{\tilde{\lambda}}\big\| + \|c_{yyy}(\hat{y},u)\varepsilon_y\|\,\big\|\hat{\tilde{v}}\big\|\big\|\hat{\tilde{\lambda}}\big\| \\
&+ \|c_{yy}(\hat{y},u) - c_{yyy}(\hat{y},u)\varepsilon_y\|\,\big\|\hat{\tilde{v}}\big\|\,\|e_\lambda+\varepsilon_\lambda\| \\
&+ \|c_{yy}(\hat{y},u) - c_{yyy}(\hat{y},u)\varepsilon_y\|\,\|e_v+\varepsilon_v\|\,\|e_\lambda+\varepsilon_\lambda\|\,.
\end{aligned} \tag{10}$$

We now turn to evaluating the errors e_v, e_λ, and e_w. The errors for adjoint and adjoint-for-Hessian, given by (4) and (8), are governed by similar equations with different right-hand-sides. We make use of Thm. 2.2 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

$$f(x,t,e_\iota) = \big(c_{yy}(\hat{y},u)\varepsilon_y\big)^\star e_\iota + \varphi - \big(c_{yy}(\hat{y},u)\varepsilon_y\big)^\star \psi, \tag{11}$$

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 Eq. (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 $\varphi = 0, \psi = \tilde{v}$.

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

Scalar equations. In the scalar case, we have the following error bound:

Theorem 3.6. *Let $m = 1$, and $\overline{e_\iota}$ be the solution of*

$$c_y(\hat{y},u)^\star \overline{e_\iota} = \overline{f}(x,t,\overline{e_\iota}), \quad \iota = \{\lambda, w\} \tag{12}$$

with

$$\overline{f}(x,t,e) = |c_{yy}(\hat{y},u)^\star|\varepsilon_y^{max} e + |\varphi| + |c_{yy}(\hat{y},u)^\star \psi|\varepsilon_y^{max}, \tag{13}$$

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 Eq. (12), we have

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

and $\overline{e_\iota}$ is a super-solution to Eq. (4) or (8), respectively. With the standard time transformation $\tau = T - t$ the backward-in-time equations (4) or (8) and (12) are transformed to forward equations. Then by Thm. 2.2 in combination with Rem. 2.4 the claim follows. □

Remark 3.7. 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 Sec. 4 we give more details on the actual realization and sketch an algorithm in Sec. 4.3.

Reaction-diffusion systems. In the case of reaction-diffusion systems, we can construct a super-reaction function by following [2], and apply the comparison theorem Thm. 2.2.

Theorem 3.8. *Let $\underline{e}$ be a sub-solution to the adjoint error equation* (4) *or* (8) *for $\iota = \lambda, w$, respectively. Define $\overline{f}$ by*

$$\overline{f}_i(x,t,e) = \sup_{\{\eta | \underline{e} \leq \eta \leq e,\ \eta_i = e_i\}} f_i(x,t,\eta),\ i = 1,\ldots,m. \tag{14}$$

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

$$c_y(\hat{y}, u)^\star \overline{e_\iota} = \overline{f}(x, t, \overline{e_\iota}), \tag{15}$$

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

Proof. The function $\overline{f}$ constructed by (14) is uniformly Lipschitz continuous in e. It satisfies $\overline{f}(x, t, e) \geq f(x, t, e)\ \forall e \in \mathbb{R}^m$ and is quasi-monotone non-decreasing, see [2]. Thus $e_\iota \leq \overline{e_\iota}$ by Thm. 2.2. □

Remark 3.9. The construction of super-reaction functions by (14) 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 [6], see also Sec. 5.2.

4 Adaptive Quantization for Newton-CG

In this section we analyze the quantization accuracy required for the convergence of Newton-CG methods in detail. Adaptive choice of quantization tolerances for a steepest-descent method were already discussed in [27]. Specific to an optimal control problem in cardiac defibrillation, adaptive quantization for the Newton-CG method was introduced in [6]. Here, we generalize and extend these results, based on the error equations of the previous section.

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

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

with $\rho \to 0$ for super-linear convergence [3, 19]. As $\|r\| \leq \|\tilde{r}\| + \|\tilde{r} - r\|$, we need to control three error contributions. Thus we have to ensure

$$\|e_{j'}\| + \|\tilde{r}\| + \|\tilde{r} - r\| \leq \rho \|j'(u)\|. \tag{16}$$

As $\|j'(u)\| = \left\|\tilde{j}'(u) - e_{j'}\right\| \geq \left\|\tilde{j}'(u)\right\| - \|e_{j'}\|$, Eq. (16) is replaced by

$$(1 + \rho) \|e_{j'}\| + \|\tilde{r}\| + \|\tilde{r} - r\| \leq \rho \left\|\tilde{j}'(u)\right\|. \tag{17}$$

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

$$\|e_{j'}\| \leq \zeta_1 \rho \left\|\tilde{j}'(u)\right\| / (1+\rho),\ \ \|\tilde{r}\| \leq \zeta_2 \rho \left\|\tilde{j}'(u)\right\|,\ \ \|\tilde{r} - r\| \leq (1-\zeta_1-\zeta_2)\rho \left\|\tilde{j}'(u)\right\| \tag{18}$$

hold.

We discuss these three accuracy conditions in the following.

4.1 Adaptive Quantization for the 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 4.1. *In iteration i of the Newton method, define*

$$\mu = \|c_u(\hat{y}, u)^\star \overline{e_\lambda}\| \tag{19}$$

with the error estimate $\overline{e_\lambda}$ from Thm. 3.6 or Thm. 3.8 for $\varepsilon_y^{max} = \mathbf{1}$. 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

$$\delta_{i+1}^y \leq \frac{\theta \zeta_1 \rho_{i+1}}{(1+\rho_{i+1})\mu}, \tag{20}$$

$\|e_{j',i+1}\| \leq \zeta_1 \rho_{i+1} \left\|\tilde{j}'(u_{i+1})\right\|$ 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 $\|c_u(\hat{y}, u)^\star e_\lambda\| \leq \delta \|c_u(\hat{y}, u)^\star \overline{e_\lambda}\|$ for $\varepsilon_y^{\max} = \delta$. This yields

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

□

For a computationally available approximation of $\overline{e_\lambda}$, we refer to Sec. 4.3, see especially Eq. (31).

Remark 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 e. g. [19]).

2. As we only *approximate* the worst-case error $\overline{e_\lambda}$ and the gradient norm of the next iteration, we can not guarantee to keep the error bound $\|e_{j'}\| \leq \zeta_1 \rho_{i+1} \|j'(u_{i+1})\|$. 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.

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.

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. in [21, 24]. In [6], we adapted their work to our setting, proposing a quantization tolerance

$$\delta_k^v \leq \frac{l_m}{\mu} \frac{(1-\zeta_1-\zeta_2)\rho_i \left\|\tilde{j}'(u_i)\right\|}{\|\tilde{r}^k\|}. \tag{21}$$

for the linearized-state trajectory in iteration k of the CG. Similar to Thm. 4.1, in Eq. (21) $\mu = \|c_u(\hat{y},u)^\star \overline{e_w}\|$ with a worst case error bound $\overline{e_w}$ for the error in the adjoint-for-Hessian solution.

As discussed in [6], we used $\lambda_{\min}/m$ as a heuristic for the unavailable, problem-dependent value l_m. There, $\lambda_{\min}$ is an estimate for the smallest eigenvalue of the reduced Hessian, and m the maximal allowed number of CG iterations. $\lambda_{\min}$ can be estimated during the CG method using an inexact Rayleigh quotient. 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. Such a restart approach was necessary, as due to the unknown 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 [9, 13].

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

$$\delta u^{k+1} = \delta u^k + \alpha^k p^k \tag{22}$$

$$\tilde{r}^{k+1} = \tilde{r}^k - \alpha^k j''(u)p^k + \xi^{k+1} \tag{23}$$

with direction p^k and $\alpha^k = (\tilde{r}^k, \tilde{r}^k)/(j''(u)p^k + e^k_{\mathrm{mv},v}, p^k)$. Here $\xi^{k+1} = -\alpha^k e^k_{\mathrm{mv},v}$ with $e^k_{\mathrm{mv},v}$ denoting the error in the computed product $j''(u)p^k$ due to compressed storage of the linearized-state v. By Eq. (22) 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 (23) as

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

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

$$\left\|r^{k+1} - \tilde{r}^{k+1}\right\| \leq \sum_{j=0}^{k} |\alpha^j| \left\|e^j_{\mathrm{mv},v}\right\| =: E^{k+1}. \tag{24}$$

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. in [22, 25]. Analogously to the latter, we trigger the restart in iteration k, when the estimated, accumulated residual error E fulfills

$$E^k > \epsilon \left\| \tilde{r}^k \right\|, \quad E^k > 1.1 E^{\text{init}}, \tag{25}$$

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 $j''(u)\delta u^k + \tilde{j}'(u)$. For the evaluation of the Hessian-vector product, δ^v is multiplied by some factor $s^v \leq 1$, such that the linearized state is stored more accurately.

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 (8). Considering the estimate (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

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

by choosing δ^y as

$$\delta^y_{i+1} \leq \frac{\text{TOL}_\lambda}{\|c_{yy}(\hat{y}, u) - c_{yyy}(\hat{y}, u)\varepsilon_y\| \left\|\hat{\tilde{v}}\right\|} \frac{1}{\|\overline{e_\lambda}\|}. \tag{26}$$

As before, $\overline{e_\lambda}$ is the worst case error in the adjoint given by Thm. 3.6 or Thm. 3.8, respectively.

For the evaluation of Eq. (27), an estimate for $\left\|\hat{\tilde{v}}\right\|$ has to be provided. Apart from restarts, $\tilde{v}$ is determined by the linear parabolic equation

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

in CG iteration k. As $p^0 = r^0 = -\tilde{j}'(u)$ we estimate $\left\|\hat{\tilde{v}}\right\| \leq c\left\|\tilde{j}'(u)\right\|$, 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. $\text{TOL}_\lambda = \zeta_1 \rho \left\|\tilde{j}'(u)\right\| / (1+\rho)$. Combined, Eq. (26) becomes

$$\delta^y_{i+1} \leq \frac{\zeta_1 \rho_{i+1}}{\tilde{c}(1+\rho_{i+1}) \|c_{yy}(\hat{y}, u) - c_{yyy}(\hat{y}, u)\varepsilon_y\|} \frac{1}{\|\overline{e_\lambda}\|}. \tag{27}$$

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

$$\delta^\lambda \leq s^\lambda \frac{\text{TOL}_\lambda}{\|c_{yy}(\hat{y}, u) - c_{yyy}(\hat{y}, u)\varepsilon_y\| \left\|\hat{\tilde{v}}\right\|} \tag{28}$$

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

4.3 Realization

To fix the details we present an algorithm for a Newton-CG method using lossy compression with adaptive quantization tolerances. 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 Thm. 3.8.

Algorithm 1 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 $\delta u^0 = 0, p^0 = r^0 = -\tilde{j}', E = 0, s^v = 1$, 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 (21)
9: solve adjoint-for-Hessian equation $c_y(\hat{y}, u_i)^\star \tilde{w} = \tilde{z}$
10: compute α^k, update $\delta u^{k+1}, r^{k+1}, p^{k+1}$
11: estimate error of Hessian-vector product $\left\|e^k_{\mathrm{mv},v}\right\|$
12: update $E = E + |\alpha^k| \left\|e^k_{\mathrm{mv},v}\right\|$
13: **if** E satisfies the restart conditions (25) **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: **end while**
18: estimate new values for $\delta_{i+1}^y, \delta_{i+1}^\lambda$
19: compute suitable step size s_i and update $u_{i+1} = u_i + s_i \delta u$
20: **end for**

In line 5 of Alg. 1 we solve the error equation

$$c_y(\hat{y}, u_i)^\star e = |c_{yy}(\hat{y}, u_i)^\star| \delta_i^y e + \mathbf{1}. \tag{29}$$

Compared to Thm. 3.6, the terms $|\phi|$ and $|\psi|$ are replaced by the constant $\mathbf{1}$-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 [6], 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

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

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 Eq. (21).

Further, μ 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 $\mu \approx \|c_u(\hat{y}, u_i)^\star e\|$ 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 Eq. (29) by $\left\|c_{yy}(\hat{y}, u_i)\hat{\tilde{v}}\right\|_{L^\infty}\delta^y_i$.

Additionally, we can evaluate $\overline{e_z}$ by Eq. (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 Eq. 29) 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

$$\overline{e_{\mathrm{mv}}} = \|c_u(\hat{y}, u_i)^\star e\|\left(\left\|\overline{e_z}\right\| + \left\|c_{yy}(\hat{y}, u_i)^\star\tilde{w}\right\|_{L^\infty}\delta^y_i\right). \tag{30}$$

Remark 4.3. 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 18. The state quantization tolerance is computed by using the minimum of the values given by Eqs. (27), (20). In the latter,

$$\mu = \left\|c_u(\hat{y}, u_i)^\star e\right\|\left\|-J_{yy}(\hat{y}, u_i) - c_{yy}(\hat{y}, u_i)^\star\tilde{\lambda}\right\|_{L^\infty}, \tag{31}$$

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^λ_{i+1} is determined by Eq. (28).

Remark 4.4. 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 Numerical Examples

For the numerical examples, we restrict ourselves to 2D problems. The adaptive lossy compression method has been implemented in the C++ finite element toolbox Kaskade 7 [8]. The *compression factor* is used to measure the performance of the compression method; it is defined as the ratio between uncompressed and compressed data. All computations were carried out on a Dual-Core AMD Opteron 8220 CPU with 2.8 GHz, without using parallelization.

5.1 Kolmogorov Equation

As a first example we consider an optimization problem from [27], 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

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

subject to

$$\begin{aligned} y_t - \sigma^2 \Delta y &= f(y) + \chi_{\Omega_c} u(t) && \text{in } \Omega \times (0,T) \\ \partial_\nu y &= 0 && \text{on } \partial\Omega \times (0,T) \\ y(\cdot,0) &= y_0 && \text{in } \Omega \end{aligned}$$

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

Following the theory presented above, 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

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

We use the algorithm sketched in Sec. 4.3, without any additional a-priori information, on a fixed FE grid with 32768 elements and 16641 vertices. Fig. 1 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 Fig. 2, where $s^\lambda = 1$ for the determination of δ^λ by Eq. (28). Before iteration 5, the state quantization tolerance δ^y is determined by Eq. (27), afterwards the tolerance given by Eq. (20) 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 problem is to use more problem-dependent information for the error estimation, to achieve tighter error bounds. To some extend 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).

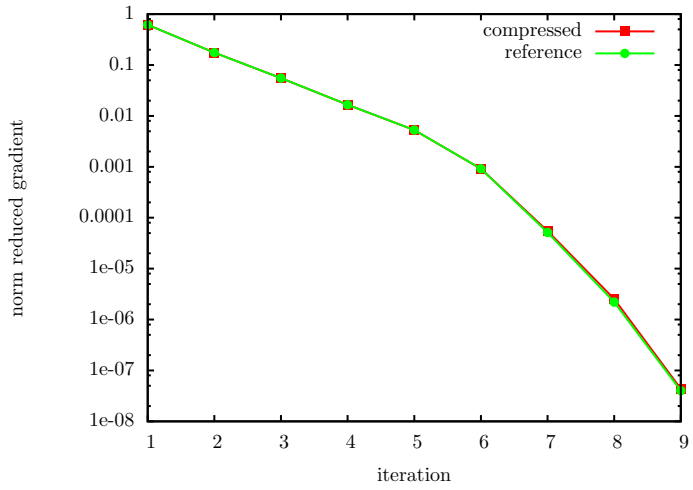

Figure 1: L^2-norm of the reduced gradient with and without compression. No significant difference is notable.

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 Eq. (25)). The safety factor s^v is set to one in the beginning, and multiplied by 0.1 on each re-computation of the residual. Fig. 3 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

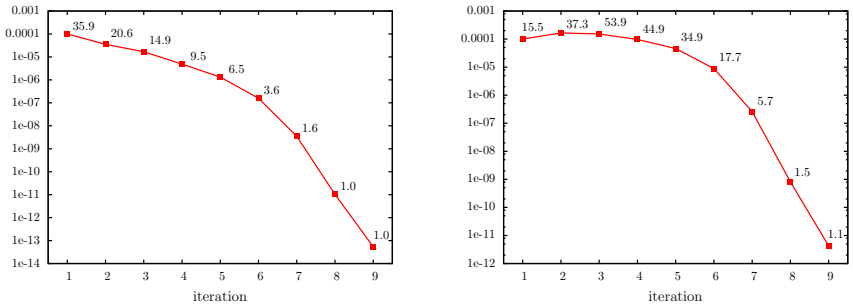

Figure 2: Adaptively chosen quantization tolerances and corresponding compression factors for state (left) and adjoint (right).

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.

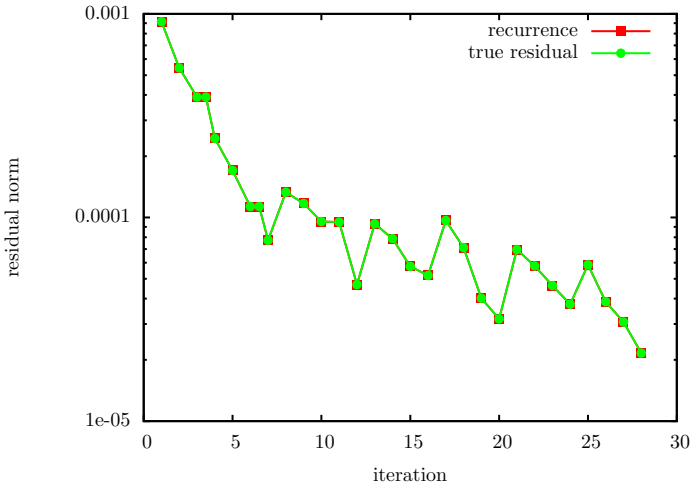

Figure 3: Behavior of CG in Newton iteration 6. The *true residual* denotes the residual without quantization of the linearized state. In iterations 3 and 6 the residual was re-computed.

With our previous restart strategy (see [6], and the discussion in Sec. 4.2.1), 9 Newton iterations with 189 CG iterations and 33 restarts were required for convergence. The behavior of both strategies is compared in Fig. 4. With the new residual replacement, re-computations were required in CG iterations 3 and 4, whereas the previous approach leads to restarts in CG iteration 2,3,4, and 6. Overall, using the old 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 newly proposed residual replacement this perturbation is minimal, as all other quantities except

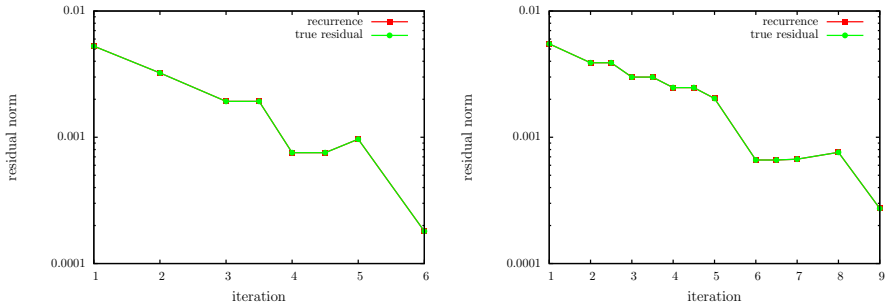

Figure 4: Behavior of CG in Newton iteration 5. The *true residual* denotes the residual without quantization of the linearized state. Left: new residual replacement, right: old restart strategy.

the residual itself are kept.

5.2 Monodomain Equations

As an example for reaction-diffusion systems, we consider an optimal control problem for the monodomain equations (see e.g. [18, 14]) 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 [20]. The state system is given by

$$\begin{aligned} v_t &= \nabla \cdot \sigma \nabla v - I_{\text{ion}}(v,w) + \chi_{\Omega_c} u(t) \\ w_t &= \eta_2\big(\frac{v}{v_p} - \eta_3 w\big), \end{aligned} \tag{32}$$

with

$$I_{\text{ion}}(v,w) = gv\big(1 - \frac{v}{v_{th}}\big)\big(1 - \frac{v}{v_p}\big) + \eta_1 vw.$$

Homogeneous Neumann boundary conditions and initial values

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

are prescribed. Here, Ω_{exi} is a circle with radius 0.04 and midpoint $(0.5, 0.5)$. The state variable is $y = (v,w)$; $\sigma : \mathbb{R}^2 \to \mathbb{R}^{2\times 2}$ and $g, \eta_i, v_p, v_{th} \in \mathbb{R}_+$ are given parameters. For details, see e.g. [17]. The control u is spatially constant on the control domain $\Omega_c = [0.37, 0.4] \times [0.45, 0.55] \cup [0.6, 0.63] \times [0.45, 0.55]$, modeling an extracellular current stimulus.

For the objective functional we choose

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

i.e. we aim at damping out an excitation wave. We set $\Omega_{\text{obs}} = \Omega \setminus \big([0.35, 0.42] \times [0.43, 0.57] \cup [0.58, 0.65] \times [0.43, 0.57]\big)$, and $\alpha = 3 \times 10^{-6}$. A detailed discussion of this optimal control problem, including optimality conditions, can be found in [6].

To apply the adaptive error control techniques presented in Sec. 4, we need to derive a super-reaction function for the system (32). The adjoint variables are denoted by (p, q), the solution of the linearized-state equation by $(\delta v, \delta w)$, and the adjoint-for-Hessian solution by $(\delta p, \delta q)$. Adapting the approach presented in [6] to the present setting, we get the following equations for the error in the adjoint variables:

$$\begin{aligned} -[e_p]_t - \nabla \cdot \sigma \nabla e_p - a e_p + \frac{\eta_2}{v_{\text{p}}} \underline{e}_q &= c \\ -[e_q]_t - d e_p + \eta_2 \eta_3 e_q &= f \end{aligned} \tag{33}$$

where

$$\begin{aligned} a &= -[I_{\text{ion}}]_v(\hat{v}, \hat{w}) + [I_{\text{ion}}]_{vv}(\hat{v}, \hat{w})\varepsilon_v + [I_{\text{ion}}]_{vw}(\hat{v}, \hat{w})\varepsilon_w \\ c &= |\chi_{\Omega_{\text{obs}}}\varepsilon_v - [I_{\text{ion}}]_{vv}(\hat{v}, \hat{w})\varepsilon_v \hat{\tilde{p}} - [I_{\text{ion}}]_{vw}(\hat{v}, \hat{w})\varepsilon_w \hat{\tilde{p}}| \\ d &= |[I_{\text{ion}}]_w(\hat{v}, \hat{w}) + [I_{\text{ion}}]_{wv}(\hat{v}, \hat{w})\varepsilon_v| \\ f &= |[I_{\text{ion}}]_{wv}(\hat{v}, \hat{w})\varepsilon_v \hat{\tilde{p}}|. \end{aligned}$$

The sub-solution $\underline{e}_q$ required for Thm. 3.8 is given by

$$\underline{e}_q = \min\left\{\underline{c} v_{\text{p}}/\eta_2,\ \underline{f}/(\eta_2 \eta_2)\right\},$$

with

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

We refer to [6, Lem. 4.8] for the derivation.

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,

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

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. For the residual replacement, we choose the same parameters ϵ, s as in the scalar example above (Sec. 5.1).

As before, the optimization progress is not affected by lossy compression of the trajectories (Fig. 5). We show the corresponding compression factors for state and adjoint in Fig. 6. As in the previous example, the quantization tolerances decrease during the Newton iterations. In contrast to the previous work [6], here the quantization tolerance for the adjoint is chosen adaptively as well.

For one solution of the state equation, 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.

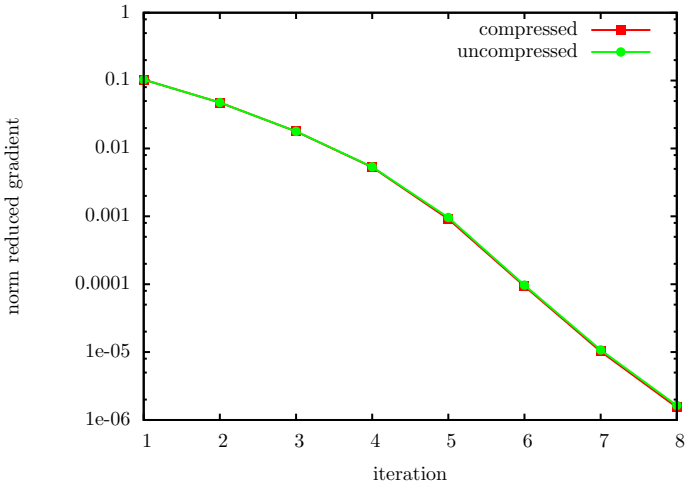

Figure 5: L^2-norm of the reduced gradient with and without compression.

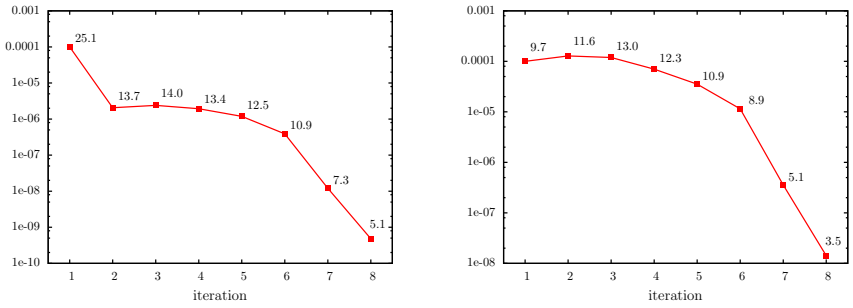

Figure 6: Adaptively chosen quantization tolerances $\delta^{\cdot}$ and corresponding compression factors for state (left) and adjoint (right).

We show the behavior of the CG method during Newton iteration 7 in Fig. 7. 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.

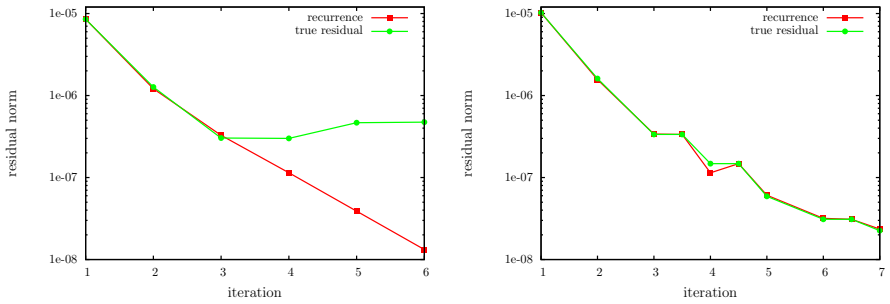

Figure 7: Behavior of CG in the final Newton iteration. The *true residual* denotes the residual without quantization of the linearized state. Left: behavior without residual replacement; Right: with residual replacement.

Remark 5.1. In this paper, we only considered fixed, uniform meshes. The lossy compression technique can be applied without modifications to adaptively refined meshes as well, see e.g. [6]. In this case, the connectivity of the grids has to be stored as well. We refer to [26] for an efficient algorithm. Combination of mesh storage and lossy compression for the FE solutions is presented in [7].

Conclusion

In this paper, we analyzed the impact of lossy trajectory compression on Newton-CG methods for optimal control of parabolic PDEs. We extended and generalized previous results, and presented an algorithm for adaptively choosing quantization tolerances. While the compression itself, as well as the error estimation, incur a negligible overhead in computation time, the overhead for re-computation of residuals is slightly more significant. The numerical examples show the expected qualitative behavior with respect to quantization tolerances and compression factors, but it is notable that considering the worst-case error leads to a severe overestimation of the error. This decreases the efficiency of the compression, as the chosen quantization tolerances are too small. A possible remedy is using problem-specific information to derive tighter error estimates. As typically the quantization error is oscillatory, future work will be concerned with the derivation of alternative error estimates, taking that oscillatory nature into account.

Acknowledgment

The authors gratefully acknowledge support by the DFG Research Center MATHEON, project F9.

References

[1] Amina Bouras and Valérie Frayssé. Inexact matrix-vector products in Krylov methods for solving linear systems: A relaxation strategy. *SIAM J.*

Matrix Anal. Appl., 26(3):660–678, 2005.

[2] Nicholas F. Britton. *Reaction-Diffusion Equations and Their Application to Biology*. Academic Press, 1986.

[3] Peter Deuflhard and Martin 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. 1997.

[4] Xiuhong Du, Eldad Haber, Maria Karampataki, and Daniel B. Szyld. Varying iteration accuracy using inexact conjugate gradients in control problems governed by pdes. In Raju Chen, editor, *Proc. 2nd Annual Conference on Computational Mathematics, Computational Geometry & Statistics (CMCGS 2013)*, pages 29–38, 2013.

[5] Paul C. Fife and Min Ming 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.

[6] Sebastian Götschel, Nagaiah Chamakuri, Karl Kunisch, and Martin Weiser. Lossy compression in optimal control of cardiac defibrillation. *J. Sci. Comput.*, 2013.

[7] Sebastian Götschel, Christoph von Tycowicz, Konrad Polthier, and Martin Weiser. Reducing memory requirements in scientific computing and optimal control. Technical Report 13-64, Zuse Institute Berlin, 2013.

[8] Sebastian Götschel, Martin Weiser, and Anton Schiela. Solving optimal control problems with the Kaskade 7 finite element toolbox. In Andreas Dedner, Bernd Flemisch, and Robert Klöfkorn, editors, *Advances in DUNE*, pages 101–112. Springer, 2012.

[9] Anne Greenbaum. Estimating the attainable accuracy of recursively computed residual methods. *SIAM J. Matrix Anal. Appl.*, 18(3):535–551, 1997.

[10] Andreas Griewank. Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation. *Optim. Methods Softw.*, 1(1):35–54, 1992.

[11] Andreas Griewank and Andrea Walther. Algorithm 799: revolve: an implementation of checkpointing for the reverse or adjoint mode of computational differentiation. *ACM Transactions on Mathematical Software (TOMS)*, 26(1):19–45, 2000.

[12] Andreas Griewank and Andrea Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, Philadelphia, 2008.

[13] Martin H. Gutknecht and Zdeněk 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.

[14] Karl Kunisch and Marcus 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.

[15] Gérard A. Meurant. *The Lanczos and conjugate gradient algorithms: from theory to finite precision computations*, volume 19. SIAM, 2006.

[16] Gérard A. Meurant and Zdeněk Strakoš. The Lanczos and conjugate gradient algorithms in finite precision arithmetic. *Acta Numerica*, 15(1):471–542, 2006.

[17] Chamakuri Nagaiah, Karl Kunisch, and Gernot 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.

[18] Bjørn Fredrik Nielsen, Tomas Syrstad Ruud, Glenn Terje Lines, and Aslak Tveito. Optimal monodomain approximations of the bidomain equations. *Appl. Math. Comput.*, 184(2):276–290, 2007.

[19] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Verlag, New York, second edition edition, 2006.

[20] Jack M. Rogers and Andrew D. McCulloch. A collocation-Galerkin finite element model of cardiac action potential propagation. *IEEE Trans. Biomed. Eng.*, 41:743–757, 1994.

[21] Valeria Simoncini and Daniel B. Szyld. Theory of inexact krylov subspace methods and applications to scientific computing. *SIAM J. Sci. Comput.*, 25(2):454–477, 2003.

[22] Gerard L. G. Sleijpen and Henk A. van der Vorst. Reliable updated residuals in hybrid Bi-CG methods. *Computing*, 56(2):141–163, 1996.

[23] Zdeněk Strakoš and Petr Tichý. On error estimation in the conjugate gradient method and why it works in finite precision computations. *Electron. Trans. Numer. Anal.*, 13:56–80, 2002.

[24] Jasper van den Eshof and Gerard L. G. Sleijpen. Inexact Krylov subspace methods for linear systems. *SIAM J. Matrix Anal. Appl.*, 26(1):125–153, 2004.

[25] Henk A. van der Vorst and Qiang Ye. Residual replacement strategies for Krylov subspace iterative methods for the convergence of true residuals. *SIAM J. Sci. Comput.*, 22(3):835–852, 2000.

[26] Christoph von Tycowicz, Felix Kälberer, and Konrad Polthier. Context-based coding of adaptive multiresolution meshes. *Computer Graphics Forum*, 30(8):2231–2245, 2011.

[27] Martin Weiser and Sebastian Götschel. State trajectory compression for optimal control with parabolic PDEs. *SIAM J. Sci. Comput.*, 34(1):A161–A184, 2012.