SEBASTIAN GÖTSCHEL, CHRISTOPH VON TYCOWICZ,
KONRAD POLTHIER, MARTIN WEISER

Reducing Memory Requirements in Scientific Computing and Optimal Control

ZIB-Report 13-64 (October 2013)

Herausgegeben vom
Konrad-Zuse-Zentrum für Informationstechnik Berlin
Takustraße 7
D-14195 Berlin-Dahlem

Telefon: 030-84185-0
Telefax: 030-84185-125

e-mail: `bibliothek@zib.de`
URL: `http://www.zib.de`

ZIB-Report (Print) ISSN 1438-0064
ZIB-Report (Internet) ISSN 2192-7782

Reducing Memory Requirements in Scientific Computing and Optimal Control

Sebastian Götschel[*] Christoph von Tycowicz[†]

Konrad Polthier[†] Martin Weiser[*]

Abstract

In high accuracy numerical simulations and optimal control of time-dependent processes, often both many time steps and fine spatial discretizations are needed. Adjoint gradient computation, or post-processing of simulation results, requires the storage of the solution trajectories over the whole time, if necessary together with the adaptively refined spatial grids. In this paper we discuss various techniques to reduce the memory requirements, focusing first on the storage of the solution data, which typically are double precision floating point values. We highlight advantages and disadvantages of the different approaches. Moreover, we present an algorithm for the efficient storage of adaptively refined, hierarchic grids, and the integration with the compressed storage of solution data.

1 Introduction

The numerical solution and optimal control of time-dependent, nonlinear PDEs often requires fine discretization both of the time interval $[0,T]$ and the — typically three-dimensional — spatial domain Ω to achieve accurate results. For optimization, adjoint gradient computation is often used, see e.g. [28]. There, the solution trajectory over the whole time interval needs to be stored, together with the adaptively refined spatial grids. To be more precise, consider the abstract optimal control problem

$$\min_{y,u} J(y,u) \text{ s.t. } c(y,u) = 0,$$

where the relation between the state y and the control u is governed by the equality constraint $c : Y \times U \to Z^\star$, which, for example, may be a parabolic PDE on Hilbert spaces Y,U,Z. Under suitable assumptions, and with $y = y(u)$ the unique solution of the state equation $c(y,u) = 0$, we arrive at the reduced formulation

$$\min_{u} j(u) := J(y(u),u).$$

The reduced gradient, required for the optimization, is then given by

$$j'(u) = J_u(y,u) - c_u(y,u)^\star \lambda,$$

[*]Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany (goetschel@zib.de, weiser@zib.de)

[†]Freie Universität Berlin, Arnimallee 6, 14195 Berlin, Germany (christoph.vontycowicz@fu-berlin.de, konrad.polthier@fu-berlin.de)

where the adjoint variable λ fulfills $c_y(y,u)^\star\lambda = J_y(y,u)$. The notations $J_y, J_u, \dots$ denote the partial derivatives with respect to the variables y, u. The adjoint equation is backwards-in-time, and — depending on the objective functional and the state equation — the solution of the forward state equation is needed for the adjoint equation, so the state has to be stored at every timestep.

Of course the storage of simulation results is not only important for optimization, but also for other post-processing algorithms, visualization, archiving of results, and more.

Not only the mere storage size is important, with the ever-growing speed of CPUs, memory access time is more and more becoming a bottleneck for large-scale simulation and optimization. To reduce the amount of data, compression methods are required to be able to tackle real-world applications. In this paper, we discuss various techniques to reduce the memory requirements, both bandwidth and size. An important 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. 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.

This paper is organized as follows. In Sec. 2, we discuss approaches for general floating point compression, before we come to methods specialized for optimal control in Sec. 3. In both sections we mainly focus on the compression of the solution data. In Sec. 4 we describe an algorithm for the efficient storage of the adaptively refined spatial grids, and its integration with the lossy compression approach discussed in Sec. 3.2.

2 General floating point compression

In this section, we discuss approaches for general pupose floating point compression, both lossless and lossy.

2.1 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 ratios are good indicators for the quality and applicability of the algorithms to our problem at hand.

FPC. In [8], 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, fcm [57] and dfcm [17] are used, 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 bit specifying the chosen predictor and the remaining non-zero bytes of the prediction error. The reported compression ratios range between

1.02 and up to 15.05 (for one special test data set), the geometric mean compression ratio 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 [48], traverses the data in some coherent order, and uses the Lorenzo predictor [31] 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 [49], a variant of arithmetic coding. Lossless compression ratios of 1.4 – 2.7 for a double precision test data set are reported in [48], with a average ratio of approximately 1.6.

2.2 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 ratio 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, makes it difficult to give a quantitative comparison of the algorithms described below.

Adaptive coarsening/sub-sampling. This method, presented in [59, 73], 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 [73] 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 [36] propose a compression algorithm based on the decomposition of the compuational grid into so-called ε-bounded sets. The method works on structured and unstructured meshes, which are modelled 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. [45, 46] 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 reconstrued data is reported by two quantities, the normalized root mean squared error (NRMSE), and Pearson's correlation coefficient defined by

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

where D denotes the original data, $\tilde{D}$ the de-compressed data, and σ the standard deviation. In [46] they report compression factors between 3.8 and 5.6 for error bounds $\rho > 0.99$ and NRMSE < 0.01 and five different data sets.

FEMZIP. FEMZIP [68, 67] is a commercial tool for the compression of finite element solutions created by certain FE-programs. After a quantization step with prescribed relative or absolute tolerance, a prediction step follows. In space, a hierarchic approximation of the FE functions is performed, using coarsening of the computational grid by algebraic multigrid techniques [68]. 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 [67], but without quantitative specification of the accuracy.

2.3 Geometry compression

Compression of geometric data is a vital factor in many computer graphics and visualization applications. Here we will briefly discuss techniques developed for the compression of polygonal surface meshes. For a more detailed overview and comparisons of various schemes we refer to the excellent surveys [3] and [54].

A wealth of compression schemes have been developed for single-rate coding (compressing the whole mesh in a region-growing fashion) as well as progressive coding (encoding the model from coarse to fine). For triangle meshes, the most prominent single-rate coders are the Edgebreaker [56] and the method of Touma and Gotsman [69] which both spawned numerous descendants. In particular, the early-split coder of Isenburg and Snoeyink [34] and the optimized Edgebreaker encoding of Szymczak [66] are among the best-performing variants and are able to achieve bit rates well below the Tutte limit [71] of roughly 3.24 bits per vertex. In addition, many triangle mesh compression schemes have been generalized to polygonal meshes, such as Martin Isenburg's method [32] which extends the Touma-Gotsman coder.

Bits rates can be improved even further by accessing already encoded geometry data when encoding connectivity and vice versa, hence exploiting the correlation between connectivity and geometry. Based on this approach, FreeLence [38] is especially performant in the triangular case, while Angle Analyzer [47] is optimized for quadrilateral meshes.

Progressive coders follow a different approach: the coder starts from a coarse mesh and then successively encodes refinement data for finer representations of the model. This approach allows the application of multiresolution analysis to decorrelate high- and low-frequency components of the geometry and/or attribute data such as colors and texture coordinates. Details in the data are thus represented as wavelet-coefficients which typically feature a smaller entropy than the original representation.

Wavelet transforms have been presented for both (unstructured) hierarchical and irregular grids. The latter group employs mesh coarsening methods that progressively remove vertices causing the smallest distortion. Prominent coders in this category are [33, 2, 74]. The best results for geometry compression however have been achieved for hierarchical meshes where efficient wavelet transforms have been derived based on the notion of subdivison. The best known scheme in this group is the progressive geometry compression (PGC) codec by Khodakovsky et al. [42] adapting the established zerotree coding scheme [61] from image compression. Numerous variants have been proposed extending PGC to different types of meshes [41], resolution scalability[4] and efficient embedded quantization [53]. Using context-based entropy coding to account for intraband correlations of the wavelet coefficients, Denis et al. [13] and von Tycowicz et al. [72] are able to further improve the compression performance. In addition, [72] incorporates strategies to encode adaptively refined hierarchies independent from the geometry or attribute data which we utilize in our coding technique presented in Sec. 4.

In the field of geometry compression the accuracy is typically measured in terms of the *root mean square error* as reported by METRO [10] which is based on a point-to-surface distance and thus neglects tangential errors. For an accuracy of orders 10^{-4} to 10^{-5} w.r.t. the bounding box diameter, FreeLence reports average compression factors of 21 for irregular triangle meshes whereas [72] achieves average factors of 29 and 122 for adaptive and uniform hierarchical grids, respectively.

3 Specialized methods for optimization with differential equations

In the remainder of this paper, we focus on methods tailored to the needs of optimal control problems governed by time-dependent differential equations.

3.1 Checkpointing

So-called "checkpointing methods" are a tool for the computation of the reduced gradient using the adjoint equation. Instead of keeping track of the whole forward trajectory, only the solution at some intermediate time steps 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, such that fixed spatial grids are considered.

3.1.1 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*, e.g. [6]. Recursive application of this strategy to each group of timesteps between two checkpoints results in a *multilevel checkpointing* strategy [6, 23]. Both techniques do not yield optimal distributions, i.e. distributions leading to a minimal amount of re-computations. *Binomial checkpointing* [21, 22] 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 e.g. [22, 23]. An implementation called revolve by Griewank and Walther [22] is available.

The achieved compression factor for storage space is given by n_t/c. However, due multiple read- and write-accesses of checkpoints during the re-computations for the adjoint equation, the reduction in memory bandwidth is significantly smaller. An evaluation of the number of times a snapshot is written or read can be found in [63]. There Stumm and Walther analyse a multistage approach, where some checkpoints are kept in RAM, others written to a hard disk or tape. 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 [78]. In this example we get $r = 2$, and the computational overhead amounts to $1,948$ additional forward steps.

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 [76], or generated using heuristics [62].

3.1.2 Adaptive timesteps

If the number of time steps 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 in [29, 62], 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 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 in [27], with extensions and theoretical foundations in [64]. 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 in [77]. 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 can be achieved at all.

3.1.3 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 timesteps [23]. 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.

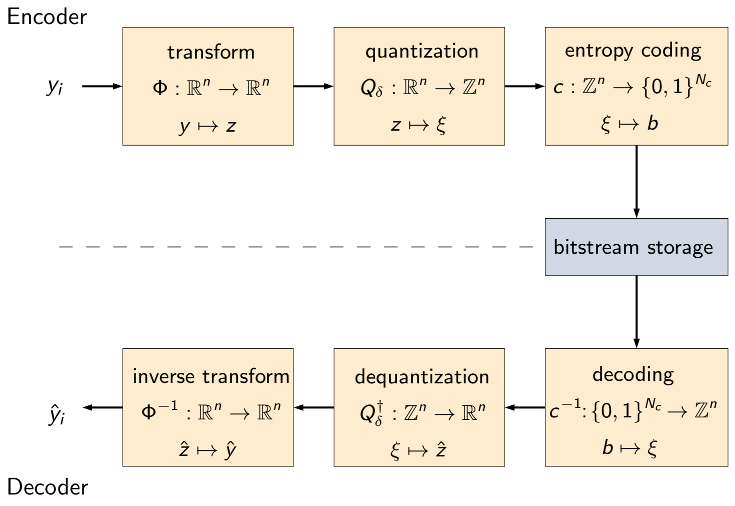
For non-uniform timestep cost which is not known a-priori, checkpoint distributions have to be chosen heuristically. With adaptive mesh refinement, also the sizes of the snapshots are unknown a priori. For this case, no optimal checkpoint distributions are known, not even heuristics.

3.2 Lossy compression

Checkpointing methods pay for the storage reduction with an increase in runtime, but reconstruct the solution data exactly. However, due to discretization of the state equation by finite elements, quadrature, and iterative solution of the resulting linear equation systems, the solution is inherently inexact. Thus a trade-off between storage demand and accuracy is natural.

3.2.1 Point-wise error bounds

In [78] we propose using the general principle of transform coding for the compression of finite element solution trajectories. It consists of a prediction step, quantization, and entropy coding of the prediction errors, see Fig. 1. To fix the setting we consider spatial discretization by a nested family $\mathscr{T}_0 \subset \cdots \subset \mathscr{T}_l$ of simplicial triangulations, constructed from an initial triangulation $\mathscr{T}_0$ of a polygonal domain $\Omega \subset \mathbb{R}^d$. This grid hierarchy can be created either by uniform or adaptive refinement. The set of vertices on level j is denoted by $\mathscr{N}_j$. The time grid for the time interval $[0,T]$ is given by $0 = t_0 < \cdots < t_f = T$. For brevity, we restrict the attention to piecewise linear finite elements.



Encoder
y_i
transform
$\Phi : \mathbb{R}^n \to \mathbb{R}^n$
$y \mapsto z$
quantization
$Q_\delta : \mathbb{R}^n \to \mathbb{Z}^n$
$z \mapsto \xi$
entropy coding
$c : \mathbb{Z}^n \to \{0,1\}^{N_c}$
$\xi \mapsto b$
bitstream storage
decoding
$c^{-1}: \{0,1\}^{N_c} \to \mathbb{Z}^n$
$b \mapsto \xi$
dequantization
$Q_\delta^\dagger : \mathbb{Z}^n \to \mathbb{R}^n$
$\xi \mapsto \hat{z}$
inverse transform
$\Phi^{-1} : \mathbb{R}^n \to \mathbb{R}^n$
$\hat{z} \mapsto \hat{y}$
$\hat{y}_i$
Decoder

Figure 1: Principle of transform coding.

Quantization. For a given accuracy $\delta > 0$ 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* $Q_\delta^\dagger : \mathbb{Z} \to \mathbb{R}$ then is given by $Q_\delta^\dagger(\xi) := 2\delta\xi$. This guarantees the quantization error bound

$$|y - Q_\delta^\dagger(Q_\delta(y))| \le \delta.$$

This implies an ℓ^∞ error bound of δ on the coefficient vector, and hence an L^∞ bound on the FE function.

Prediction in space. Values y_k of coarse level vertices are quantized directly to $\xi_k = Q_\delta(y_k)$, yielding a reconstructed value $\hat{y}_k := Q_\delta^\dagger(\xi_k)$. For new vertices $x_k \in \mathscr{N}_j \setminus \mathscr{N}_{j-1}$

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 vertices. There are several algorithmic choices for the predictor. One possibility is a change of basis from the nodal basis to the hierarchic basis [79]. This is easily implemented, as it essentially is the application of prolongation matrices between grid levels, which is easily accessible in most FE codes.

A priori estimates for the compression factors were derived in [78]. To achieve L^∞-interpolation error accuracy for the reconstructed FE function, 2.9 bits/vertex in 2D and 2.5 bits/vertex in 3D are sufficient. This amounts to compression factors of 22.1 and 25.6, respectively, compared to storing double precision floating point values at 64 bits/vertex.

Prediction in time. Additionally temporal correlations can be used to further reduce the entropy of the data. 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. We thus can use delta-encoding, and store for a timestep $t_n < T$ only the difference

$$\xi_k^\Delta(t_n) = \begin{cases} \xi_k(t_n) - \xi_k(t_{n+1}), & k \in \big(\mathcal{N}_j(t_n) \setminus \mathcal{N}_{j-1}(t_n)\big) \cap \big(\mathcal{N}_j(t_{n+1}) \setminus \mathcal{N}_{j-1}(t_n)\big) \\ \xi_k(t_n), & \text{otherwise} \end{cases}.$$

At final time T,

$$\xi_k^\Delta(t_f) = \xi_k(t_f) \quad \forall k \in \mathcal{N}_j(t_f) \setminus \mathcal{N}_{j-1}(t_f).$$

Delta-encoding the quantized coefficients avoids error accumulation. Of course higher order prediction can be used instead of the constant predictor described above, at the expense of keeping more timesteps in RAM.

Entropy coding. The quantized and possibly delta-encoded coefficients ξ_k^Δ are written to disk using range coding [49]. They are encoded with variable-size symbols, where fewer bits are assigned to the more frequent coefficients.

More details and numerical examples can be found in [78].

3.2.2 Adaptive error control

A crucial algorithmic choice is the quantization tolerance δ. To choose the error bound as large as possible without impeding the convergence of the optimization algorithm, we need to estimate the induced error in the reduced gradient $j'(u) = J_u(y,u) - c_u(y,u)^\star \lambda$. Typically, J_u and c_u are independent of y, such that the error is determined by the error of the adjoint. If additionally c_y does not depend on the state, e.g. for linear equations, the error in the adjoint e_λ satisfies the equation $c_y(y,u)^\star e_\lambda = -J_{yy}(y,u)\varepsilon_y$, where ε_y denotes the error in the reconstructed state solution. For nonlinear equations, the error additionally depends on $c_{yy}(y,u)\varepsilon_y$ and the solution of the adjoint equation with inexact data. Computationally available estimates of the gradient error can be used to determine the quantization tolerance according to the progress of the optimization procedure. A detailed discussion can be found in [19].

For second order methods, errors in the reduced Hessian have to be considered as well. A derivation of error estimates and the influence on a Newton-CG method can be found in [18] specialized to the application in optimal control of cardiac defibrillation, and more general in [19].

3.2.3 H^{-1} error bounds

While bounding the pointwise ℓ^∞ error in the coefficients of the reconstructed FE solution trajectory is easily implemented and yields good compression factors, considering other error measures is reasonable in the optimal control setting. In particular, the reconstructed solution enters into the right-hand side of the adjoint equation. Due to smoothing properties of parabolic equations, a quantization error with high spatial frequency is preferable, such that the H^{-1}-norm is more appropriate.

Controlling the H^{-1} error can be achieved by using a wavelet transform to represent the finite element solution $y(x,t_n)$ at some fixed timestep t_n as

$$y(x,t_n) = \sum_{k\in\mathcal{N}_0} y_{0,k}\phi_{0,k}(x) + \sum_{j=0}^{l-1} \sum_{m\in\mathcal{N}_{j+1}\setminus\mathcal{N}_j} z_{j,m}\psi_{j,m}(x).$$

For this we assume again a level partitioning of the grid vertices $\mathcal{N} = \mathcal{N}_0 \cup \cdots \cup \mathcal{N}_l$, and denote by the subscript j,k values belonging to vertex k on grid level j. We use the abbreviations $n(j,k) = \{m \in \mathcal{N}_{j+1}\setminus\mathcal{N}_j \mid m \text{ is a child of } k\}$ and $N(j,m) = \{k \in \mathcal{N}_j \mid k \text{ is a parent of } m\}$. Here, a vertex $m \in \mathcal{N}_{j+1}$ is a child of $k \in \mathcal{N}_j$, if m was created by splitting an edge $[k,l]$. The *scaling functions* $\phi_{j,k}$ satisfy the refinement relation

$$\phi_{j,k} = \phi_{j+1,k} + \sum_{m\in n(j,k)} \frac{1}{2}\phi_{j+1,m},$$

the *wavelets* are of the form

$$\psi_{j,m} = \phi_{j+1,m} - \sum_{k\in N(j,m)} s_{j,k,m}\phi_{j,k}.$$

The lifting coefficients $s_{j,k,m}$ are determined to impose vanishing moments on the wavelets, see e.g. [65, 9]. In particular, one vanishing moment is easily obtained on unstructured grids if the mass matrix is available.

The modified coarse grid values $y_{0,k}$ and wavelet coefficients $z_{j,m}$ are computed using the fast wavelet transform with lifting [58, 65], for grid levels $l-1,\dots,0$:

$$\begin{aligned} z_{j,m} &= y_{j+1,m} - \frac{1}{2}\sum_{k\in N(j,m)} y_{j,k} && \forall m \in \mathcal{N}_{j+1}\setminus\mathcal{N}_j \\ y_{j,k} &= y_{j+1,k} + \sum_{m\in n(j,k)} s_{j,k,m} z_{j,m} && \forall k \in \mathcal{N}_j. \end{aligned}$$

Norm equivalences, e.g. [12], suggest that the error bound $\|y-\hat{y}\|_{H^{-1}} < \varepsilon$ holds, if the wavelet coefficients $z_{j,k}$ are quantized using a level-dependent tolerance $\delta_j \sim 2^{j(d/2+1)}\varepsilon$.

To compare a first, simple implementation of this approach with the hierarchical basis (HB) prediction of Sec. 3.2.1, we use the three functions

$$\begin{aligned} f_1(x) &= \sin(12(x_0-0.5)(x_1-0.5)), \quad x \in [0,1]^2 \\ f_2(x) &= \sin(50(x_0-0.5)(x_1-0.5)), \quad x \in [0,1]^2 \\ f_3(x) &= \|x\|^2 + \sin(12(x_0-0.5)(x_1-0.5)), \quad x \in [0,1]^3. \end{aligned}$$

For the HB approach, quantization tolerances were chosen to achieve L^∞-interpolation error accuracy. For the wavelet approach the tolerance was set to achieve the same H^{-1} error as the corresponding HB result. The resulting compression factors shown in Fig. 2 indicate that on average a wavelet approach might indeed give better compression factors when H^{-1} reconstruction errors are used..

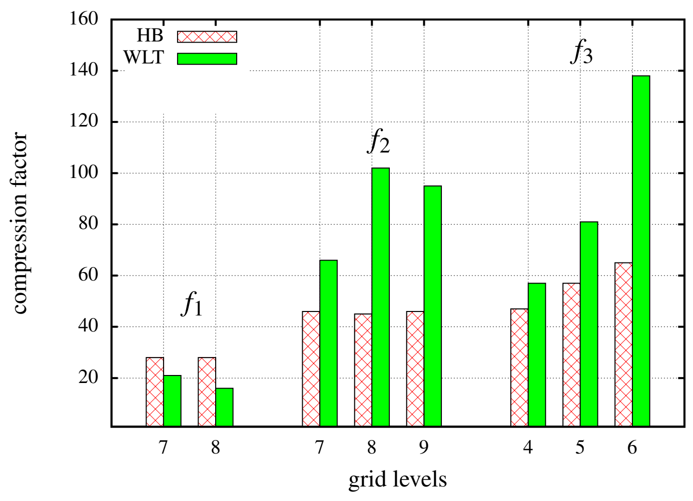


Figure 2: Compression factors for hierachical basis prediction/point-wise error bounds (HB) and wavelet-based compression (WLT), for three test functions and different mesh sizes.

3.2.4 Discussion

The lossy compression technique sketched in this section offers significant reduction of storage space as well as memory bandwidth, as only the compressed data is transferred to storage media. The computational cost of the basic method is negligible: Quantization, delta-encoding in time, and entropy coding consist only of cheap elementary arithmetic operations; in space, the prediction step amounts to the computation of products between prolongation matrices and FE coefficient vectors. Prolongation matrices are often available from multigrid preconditioners, or can otherwise be computed inexpensive on the fly.

As a downside, information is discarded in the quantization step, and the FE solution can not be reconstructed exactly. If used in optimal control of differential equations, adaptive control of the quantization error ensures that the inexactness has no influence on the convergence of the optimization. For other post-processing tasks, like data analysis or visualization, the error norms and tolerances can be chosen according to the particular needs of the application, offering a flexible way to balance data size and accuracy.

3.3 Other techniques

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

3.3.1 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. For many problems, only few basis vectors are necessary to get sufficiently accurate approximations. A detailed analysis of POD methods for parabolic PDEs can be found in [43], see e.g. [30] 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 [70], such an estimator is derived for the linear-quadratic case, and extended to semilinear equations in [40]. 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 in [37]: 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.

3.3.2 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 [14], for instance. In the last years, this principle was applied to solve optimal control problems governed by time-dependent partial differential equations, e.g. [24, 11, 25, 26]. 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 the auxiliary variables [11]. 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. [35]). Combination with adaptive mesh refinement is discussed e.g. in [25, 26], where a dual weighted residual (DWR) method [7] 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 Compression of hierarchical, unstructured grids

For problems with spatially local solution features, it is beneficial to use adaptively refined spatial meshes to reduce the computational cost and memory demand of simulation and optimization. As a downside, in the context of trajectory storage discussed here, this incurs the need to store the mesh together with the trajectory data. For applying our lossy compression approach, we even need the complete hierarchy, not just the leaf level. In this section we discuss an efficient algorithm for mesh storage [39, 72], as well as the integration of this method with the lossy compression approach described above.

4.1 Connectivity compression

Numerous strategies for the adaptive refinement of grids have been presented in the literature. Exploiting the particular structure inherent to a given strategy is paramount in the construction of an efficient compression scheme. Here we present a method that is tailored to hierarchies based on split operations for which the resulting grid is independent of the order in which the operations are applied. In particular, we confine our attention to the well-known and established *red–green refinement* [5] on 2-dimensional grids. However, the ideas presented here can also be adapted to refinement schemes for 3-dimensional grids and/or other types of elements. For example, [72] additionally provides results for quadrilateral hierarchies.

Typically, the root grid is described by a small, carefully laid out mesh that can be compressed well using single-rate coders. Explicitly, our implementation uses Free-Lence [38] to losslessly encode the triangular base mesh. Starting from the root grid, it is sufficient to encode which elements (including those on finer levels) have been refined to reconstruct the adaptive hierarchy. Thus, the hierarchy can be represented as a forest where each node corresponds to an element in the grid and the parent-child relation reflects which triangles resulted through refinement of a particular coarse one.

We convert this representation into a bit stream by traversing the forest breadth-first and writing `true` only if the node has children, *i.e.*, was refined. If a geometric criterion is used to resolve non-conforming situations between elements of differing refinement grade, we can uniquely determine the connectivity of the grid from the root grid together with the bit stream. However, if the conformization is determined exclusively by local indexing, additional symbols have to be coded whenever there is freedom of choice, *e.g.* a coarse triangle with two refined neighbors (see Fig. 3 middle). We entropy code these symbols, but found that they where virtually incompressible without knowing the exact implementation of the grid manager.

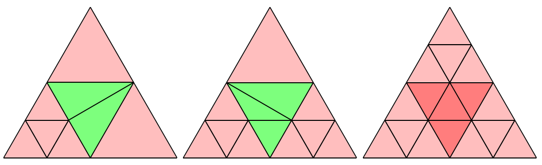


Figure 3: Red-green conformizations of non-conforming configurations due to adaptive refinement.

Before compressing the bit stream we remove nodes whose state can be implicitly reconstructed. In particular, no symbols are written in the following cases:

1-Regularity In balanced grids, neighboring triangles must not differ in more than one level of refinement to ensure a certain level of shape regularity. Thus, elements adjacent to coarse green triangles cannot be refined and can therefore be culled from the bit stream.

Stream Truncation Due to breadth-first traversal, nodes at the finest level are visited last. The corresponding `false` symbols can be left from the stream since they cause no further refinements. In fact, we discard all trailing `false` entries.

Uniform Refinement We store a separate byte that encodes the degree of uniform refinement, allowing the coder to skip all nodes on coarser levels.

Overall, for the test set of adaptive hierarchies used in [72], the above steps allow to nearly halve the number of bits in the binary representation, without even looking at the characteristics of the particular input grid. However, grids do show certain characteristics in practice and we use *context groups* as a simple measure to account for the conditional entropy (see [60]) of the bit stream. Just like two adjacent pixels in a digital photograph are likely to be similar, the refinement grades in hierarchical meshes typically tend to be locally similar. We call two nodes within one level of the hierarchy adjacent, if their corresponding triangles share an edge. This notion of locality allows us to define the context of a node based on the refinement status of its neighbors. Naturally, we may only assume knowledge of neighbors whose status is also available during decoding. Thus, we define the context of a node by the number of refined, not refined, and unknown neighbors. The latter category is made up of nodes whose status can not be implied and have not been traversed so far. We write (x, y, z) to denote the context with x refined, y not refined, and z unknown adjacent triangles. The symbols of different context groups are kept in separate arrays, which are entropy coded independently. With arithmetic coding, each context group will then compress to its conditional entropy in the limit, allowing us to exploit the correlation in the refinement status of adjacent triangles.

However, the mutual information inherent to each context group varies drastically. For example, context (0,0,3) with all neighbors unknown is virtually incompressible as no advantage can be taken of mutual information. The same holds for context groups where the extra information is rather ambiguous, for example (1,1,1), (1,2,0), and (2,1,0). At the contrary, the other context groups perform very well in the experiments. These observations motivate an optimization of the traversal scheme used within each level since the iteration of nodes can be arbitrary as long as encoder and decoder agree on a common one. Instead of iterating each node using a standard breadth-first in-order traversal of the forest, we determine an ordering that attempts to maximize the mutual information. The idea is to prioritize each node by the entropy of its current context. Learning these entropies, however, is expensive in terms of compression performance as well as computational cost. As shown in [72], this approach typically leads to a fixed prioritization of context groups once the learning phase is settled. Therefore, the effects of the learning process of the contexts' entropies can be remedied by using fixed priorities. Explicitly, context groups are assigned higher priorities with decreasing number of unknown neighbors, where ties are resolved by prioritizing contexts with a higher number of known refined neighbors. While the (culled) binary representation of the hierarchy is almost incompressible when entropy coded directly, the proposed context groups together with the improved traversal reduced the code length by more than 50% for the test data in [72].

Furthermore, the proposed context-based coding can easily be extended to time-varying series. When coding the status of a node in a sequence of frames we can extend the context groups to account for its status in a previous frame. The previous state of a node can either be refined, not refined, or it did not exist, hence we triple each context. If the refinements between frames does not vary much, the contexts corresponding to previously refined nodes will mainly comprise `true` symbols whereas the other contexts will primarily contain `false`. Therefore, grids which equal their preceding frame can be stored at no cost (except for a small overhead due to the increased number of context groups). On the contrary, if there is no correlation between the frames, the compression will be as good as in the static case since the entropy of the individual context groups can not increase.

4.2 Numerical example

The lossy compression method discussed in Sec. 3.2 is implemented in the C++ finite element toolbox Kaskade 7 [20]. An implementation of the algorithm for connectivity compression is available in JavaView [1], a toolkit for mathematical geometry processing and visualization. Both packages have been combined using the Java Native Interface to allow a fully adaptive solution of optimal control problems with compression of both meshes and solution data.

As an illustrative example we use an optimal control problem for the monodomain equations describing the electrical activity in the heart (see e.g. [51, 44]) on a simple 2D unit square domain $\Omega = (0,1)^2$. As membran model, we use the Rogers-McCulloch variant of the Fitzhugh-Nagumo model [55]. The state equations for the *transmembrane voltage v* and the *gating variable w* are given by

$$\begin{aligned} v_t &= \nabla\cdot\sigma\nabla v - gv\big(1-\frac{v}{v_{th}}\big)\big(1-\frac{v}{v_p}\big) + \eta_1 vw + \chi_{\Omega_c}u(t) \\ w_t &= \eta_2\big(\frac{v}{v_p} - \eta_3 w\big), \end{aligned}$$

together with homogeneous Neumann boundary conditions, and initial values

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

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. [50]. 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 objective is to dampen out the excitation wave front induced by the initial values,

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

where $\Omega_{\text{obs}} = \Omega \setminus \big([0.35, 0.42] \times [0.43, 0.57] \cup [0.58, 0.65] \times [0.43, 0.57]\big)$, and $\alpha = 3\times 10^{-6}$. Optimality conditions and more details can be found in [18]. We use adjoint gradient computation and the BFGS-Quasi-Newton method [52] for optimization. Spatial adaptivity is performed individually for state and adjoint using the hierarchical DLY error estimator [15]. For time stepping, a linearly implicit extrapolated Euler method [16] is used, with fixed timestep sizes for ease of implementation.

First, we consider just one iteration, i.e. one state and adjoint solve, on the time interval $[0,6]$ with timestep size 0.04. In space, we restrict the number of vertices to be less then $60,000$. We choose a fixed quantization tolerance $\delta = 10^{-2}$, yielding a relative absolute error bound of 10^{-4} for v. In Fig. 4 we show the v component of the state variable at selected times. Compression factors for the state values, and the number of bits/vertex necessary for connectivity encoding is shown in Fig. 5. 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. Detailed CPU times are shown in Table 1.

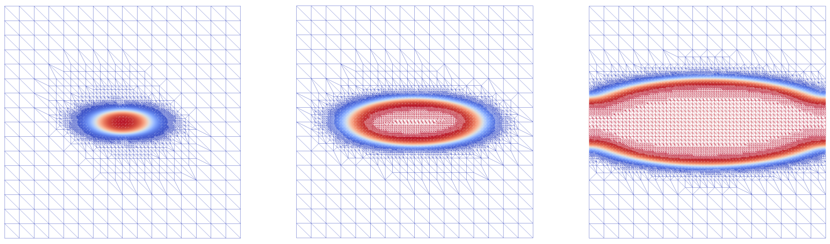


Figure 4: Uncontrolled solution v at $1ms$, $3ms$ and $6ms$. The adaptively refined meshes have $37,344$, $41,729$ and $38,346$ vertices.

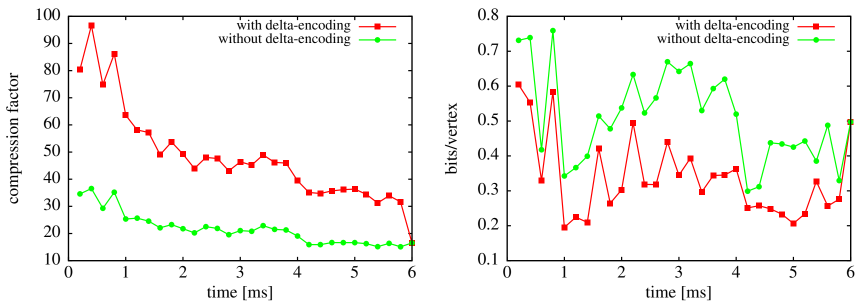


Figure 5: Compression factor of the state values using lossy compression with $\delta = 10^{-2}$ compared to double precision floating point values at 64 bits/vertex (left), bits/vertex for connectivity encoding (right), both with and without delta-encoding between timesteps.

		state values			grid		
	solve	setup	encode	decode	encode	decode	transfer
state	3026.2	31.4	11.3	–	90.6	–	–
adjoint	1473.1	31.4	–	3.7	–	59.3	183.6

Table 1: CPU times (in seconds) for one state and adjoint solve, averaged over 5 test runs. Times are measured without delta-encoding of trajectory and mesh.

solve consists of time for assembly, adaptivity, and solution of the linear systems using BiCGStab [75] with an ILU preconditioner. For state value compression, during *setup*, the prolongation matrices required for the spatial prediction are generated, which is more expensive than the actual encoding and decoding. The overall computational overhead for state value compression amounts to 1.4% in the state equation, and 2.4% in the adjoint. Encoding and decoding the mesh take up 3% and 4% in state and adjoint, respectively. As state and adjoint equations are solved on independently adapted grids, the de-compressed state trajectory has to be interpolated on the adjoint mesh (last column in Table 1) which takes up 12.5% CPU time of an adjoint solve; this overhead occurs also if the trajectory is stored uncompressed. In our current preliminary implementation, we have to re-create the mesh hierarchy in the Java code for encoding, and in the C++ code after decoding. Additionally, as different data structures are used in the two combined software toolboxes, re-assignement of the degrees of freedom is necessary after the encoding step. This significant overhead is not included in the CPU times reported here, as it can be avoided by improving the implementation.

Second, the complete optimization is performed on the time interval $[0,4]$, with timestep size 0.04, and a restriction to at most $25,000$ vertices in space. Fig. 6 shows the progress of the optimization method. For trajectory compression, different fixed quantization tolerances 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.

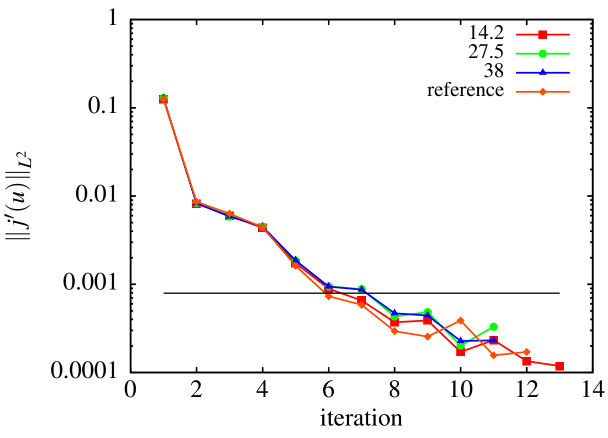


Figure 6: Optimization progress for different quantization tolerances for the state trajectory. No delta-encoding between timesteps was used.

5 Conclusion

To reduce the memory requirements of scientific data, essentially two classes of algorithms are available: Methods like checkpointing, which reduce storage space at the cost of computation time, and lossy compression techniques, where the trade-off is between memory requirements and accuracy. While general purpose floating point compression methods can be used for many different applications, good compression results can only achieved with structure-exploiting techniques, like checkpointing, FEMZIP, or our lossy compression approach.

Optimal control problems pose specific requirements for accuracy, which can be satisfied using quantitative error estimates to choose suitable quantization tolerances. The combination of lossy state vakzes compression and compressed storage of adaptively refined meshes yields significant reduction of storage space and memory bandwidth, at small computational cost.

Acknowledgment

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

References

[1] JavaView homepage. `www.javaview.de`

[2] Alliez, P., Desbrun, M.: Progressive compression for lossless transmission of triangle meshes. In: Proc. 28th Annual Conference on Computer Graphics and Interactive Techniques, pp. 195–202. ACM (2001)

[3] Alliez, P., Gotsman, C.: Recent advances in compression of 3d meshes. In: N. Dodgson, M. Floater, M. Sabin (eds.) Advances in Multiresolution for Geometric Modelling, Mathematics and Visualization, pp. 3–26. Springer Berlin Heidelberg (2005). DOI 10.1007/3-540-26808-1_1. URL `http://dx.doi.org/10.1007/3-540-26808-1_1`

[4] Avilés, M., Morán, F., García, N.: Progressive lower trees of wavelet coefficients: efficient spatial and SNR scalable coding of 3D models. Advances in Mulitmedia Information Processing-PCM 2005 pp. 61–72 (2005)

[5] Bank, R., Sherman, A., Weiser, A.: Some refinement algorithms and data structures for regular local mesh refinement. Scientific Computing, Applications of Mathematics and Computing to the Physical Sciences (1983)

[6] Becker, R., Meidner, D., Vexler, B.: Efficient numerical solution of parabolic optimization problems by finite element methods. Optim. Methods Softw. **22**(5), 813–833 (2007). DOI http://dx.doi.org/10.1080/10556780701228532

[7] Becker, R., Rannacher, R.: An optimal control approach to a posteriori error estimation in finite element methods. Acta numerica **10**(1), 1–102 (2001)

[8] Burtscher, M., Ratanaworabhan, P.: FPC: A high-speed compressor for double-precision floating-point data. IEEE Transactions on Computers **58**(1), 18–31 (2009)

[9] Castrillón-Candás, J.E., Amaratunga, K.: Spatially adapted multiwavelets and sparse representation of integral equations on general geometries. SIAM Journal on Scientific Computing **24**(5), 1530–1566 (2003)

[10] Cignoni, P., Rocchini, C., Scopigno, R.: Metro: measuring error on simplified surfaces. Tech. rep., Paris, France (1996)

[11] Comas, A.: Time–domain decomposition preconditioners for the solution of discretized parabolic optimal control problems. Ph.D. thesis, Rice University (2005)

[12] Dahmen, W.: Wavelet methods for PDEs – some recent developments. Journal of Computational and Applied Mathematics **128**(1), 133–185 (2001)

[13] Denis, L., Satti, S., Munteanu, A., Cornelis, J., Schelkens, P.: Scalable Intraband and Composite Wavelet-Based Coding of Semiregular Meshes. IEEE Transactions on Multimedia **12**(8), 773–789 (2010)

[14] Deuflhard, P., Bornemann, F.: Scientific computing with ordinary differential equations, vol. 42. Springer (2002)

[15] Deuflhard, P., Leinen, P., Yserentant, H.: Concepts of an adaptive hierarchical finite element code. IMPACT Comp. Sci. Eng. **1**(1), 3–35 (1989)

[16] Deuflhard, P., Nowak, U.: Extrapolation integrators for quasilinear implicit ODEs. In: P. Deuflhard, B. Engquist (eds.) Large Scale Scientific Computing, *Progress in Scientific Computing*, vol. 7, pp. 37–50. Birkhäuser (1987)

[17] Goeman, B., Vandierendonck, H., De Bosschere, K.: Differential FCM: Increasing value prediction accuracy by improving table usage efficiency. In: High-Performance Computer Architecture, 2001. HPCA. The Seventh International Symposium on, pp. 207–216. IEEE (2001)

[18] Götschel, S., Nagaiah, C., Kunisch, K., Weiser, M.: Lossy compression in optimal control of cardiac defibrillation. J. Sci. Comput., to appear (2013)

[19] Götschel, S., Weiser, M.: Lossy compression for PDE-constrained optimization: Adaptive error control. ZIB Report 13-27 (2013)

[20] Götschel, S., Weiser, M., Schiela, A.: Solving optimal control problems with the Kaskade 7 finite element toolbox pp. 101–112 (2012)

[21] Griewank, A.: Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation. Optimization Methods and software **1**(1), 35–54 (1992)

[22] Griewank, A., Walther, A.: 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)

[23] Griewank, A., Walther, A.: Evaluating derivatives: principles and techniques of algorithmic differentiation. SIAM, Philadelphia (2008)

[24] Heinkenschloss, M.: A time-domain decomposition iterative method for the solution of distributed linear quadratic optimal control problems. Journal of Computational and Applied Mathematics **173**(1), 169–198 (2005)

[25] Hesse, H.K.: Multiple shooting and mesh adaptation for PDE constrained optimization problems. Ph.D. thesis, University Heidelberg (2008)

[26] Hesse, H.K., Kanschat, G.: Mesh adaptive multiple shooting for partial differential equations. part I: linear quadratic optimal control problems. Journal of Numerical Mathematics **17**(3), 195–217 (2009)

[27] Heuveline, V., Walther, A.: Online checkpointing for parallel adjoint computation in PDEs: Application to goal-oriented adaptivity and flow control. In: Euro-Par 2006 Parallel Processing, pp. 689–699. Springer (2006)

[28] Hinze, M., Pinnau, R., Ulbrich, M., Ulbrich, S.: Optimization with PDE constraints. Springer, Berlin (2009)

[29] Hinze, M., Sternberg, J.: 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)

[30] Hinze, M., Volkwein, S.: Error estimates for abstract linearquadratic optimal control problems using proper orthogonal decomposition. Comput. Optim. Appl. **39**, 319–345 (2008)

[31] Ibarria, L., Lindstrom, P., Rossignac, J., Szymczak, A.: Out-of-core compression and decompression of large n-dimensional scalar fields. In: Computer Graphics Forum, vol. 22, pp. 343–348. Wiley Online Library (2003)

[32] Isenburg, M.: Compressing polygon mesh connectivity with degree duality prediction. In: Graphics Interface Conference Proceedings, pp. 161–170 (2002)

[33] Isenburg, M., Snoeyink, J.: Mesh collapse compression. In: In Proceedings of SIBGRAPI'99, pp. 27–28 (1999)

[34] Isenburg, M., Snoeyink, J.: Early-split coding of triangle mesh connectivity. In: Graphics Interface Proceedings, pp. 89–97. Canadian Information Processing Society, Toronto, Ont., Canada, Canada (2006)

[35] Ito, K., Kunisch, K.: Receding horizon optimal control for infinite dimensional systems. ESAIM: control, optimisation and calculus of variations **8**(1), 741–760 (2002)

[36] Iverson, J., Kamath, C., Karypis, G.: Fast and effective lossy compression algorithms for scientific datasets. In: Euro-Par 2012 Parallel Processing, pp. 843–856. Springer (2012)

[37] Jörres, C., Vossen, G., Herty, M.: On an inexact gradient method using proper orthogonal decomposition for parabolic optimal control problems. Computational Optimization and Applications pp. 1–10 (2013)

[38] Kälberer, F., Polthier, K., Reitebuch, U., Wardetzky, M.: Freelence - coding with free valences. Computer Graphics Forum **24**(3), 469–478 (2005)

[39] Kälberer, F., Polthier, K., von Tycowicz, C.: Lossless compression of adaptive multiresolution meshes. In: Proc. Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI), vol. 22 (2009)

[40] Kammann, E., Tröltzsch, F., Volkwein, S.: A method of a-posteriori error estimation with application to proper orthogonal decomposition. Tech. rep. (2011)

[41] Khodakovsky, A., Guskov, I.: Compression of normal meshes. In: In Geometric Modeling for Scientific Visualization, pp. 189–206. Springer-Verlag (2003)

[42] Khodakovsky, A., Schröder, P., Sweldens, W.: Progressive geometry compression. In: SIGGRAPH '00 Proceedings, pp. 271–278 (2000)

[43] Kunisch, K., Volkwein, S.: Galerkin proper orthogonal decomposition methods for parabolic problems. Numer. Math. **90**, 117–148 (2001)

[44] Kunisch, K., Wagner, M.: Optimal control of the bidomain system (I): The monodomain approximation with the RogersMcCulloch model. Nonlinear Analysis: Real World Applications **13**(4), 1525–1550 (2012). DOI 10.1016/j.nonrwa.2011.11.003. URL `http://www.sciencedirect.com/science/article/pii/S1468121811003099`

[45] Lakshminarasimhan, S., Shah, N., Ethier, S., Klasky, S., Latham, R., Ross, R., Samatova, N.F.: Compressing the incompressible with ISABELA: In-situ reduction of spatio-temporal data. In: Euro-Par 2011 Parallel Processing, pp. 366–379. Springer (2011)

[46] Lakshminarasimhan, S., Shah, N., Ethier, S., Ku, S.H., Chang, C.S., Klasky, S., Latham, R., Ross, R., Samatova, N.F.: ISABELA for effective in situ compression of scientific data. Concurrency and Computation: Practice and Experience **25**, 524–540 (2013)

[47] Lee, H., Alliez, P., Desbrun, M.: Angle-analyzer: A triangle-quad mesh codec. pp. 383–392 (2002)

[48] Lindstrom, P., Isenburg, M.: Fast and efficient compression of floating-point data. IEEE Transactions on Visualization and Computer Graphics **12**(5), 1245–1250 (2006). DOI http://dx.doi.org/10.1109/TVCG.2006.143

[49] Martin, G.: Range encoding: an algorithm for removing redundancy from a digitised message. Presented at Video & Data Recording Conference, Southampton (1979)

[50] Nagaiah, C., Kunisch, K., Plank, G.: Numerical solution for optimal control of the reaction-diffusion equations in cardiac electrophysiology. Computational Optimization and Applications **49**, 149–178 (2011). URL `http://dx.doi.org/10.1007/s10589-009-9280-3`. 10.1007/s10589-009-9280-3

[51] Nielsen, B.F., Ruud, T.S., Lines, G.T., Tveito, A.: Optimal monodomain approximations of the bidomain equations. Applied Mathematics and Computation **184**(2), 276–290 (2007)

[52] Nocedal, J., Wright, S.J.: Numerical Optimization. Springer, New York (2006)

[53] Payan, F., Antonini, M.: An efficient bit allocation for compressing normal meshes with an error-driven quantization. CAGD **22**(5), 466–486 (2005)

[54] Peng, J., Kim, C.S., Jay Kuo, C.C.: Technologies for 3d mesh compression: A survey. J. Vis. Comun. Image Represent. **16**(6), 688–733 (2005). DOI 10.1016/j.jvcir.2005.03.001. URL `http://dx.doi.org/10.1016/j.jvcir.2005.03.001`

[55] Rogers, J.M., McCulloch, A.D.: A collocation-Galerkin finite element model of cardiac action potential propagation. IEEE Trans. Biomed. Eng. **41**, 743–757 (1994)

[56] Rossignac, J.: Edgebreaker: Connectivity compression for triangle meshes. IEEE Transactions on Visualization and Computer Graphics pp. 47–61 (1999)

[57] Sazeides, Y., Smith, J.E.: The predictability of data values. In: Microarchitecture, 1997. Proceedings., Thirtieth Annual IEEE/ACM International Symposium on, pp. 248–258. IEEE (1997)

[58] Schröder, P., Sweldens, W.: Spherical wavelets: Efficiently representing functions on the sphere. In: SIGGRAPH '95 Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, pp. 161–172. ACM (1995)

[59] Shafaat, T.M., Baden, S.B.: A method of adaptive coarsening for compressing scientific datasets. In: B. Kågström, E. Elmroth, J. Dongarra, J. Wasniewski (eds.) Applied Parallel Computing. State of the Art in Scientific Computing. 8th International Workshop, PARA 2006, Umeå, Sweden, June 18-21, 2006, Revised Selected Papers, *Lecture Notes in Computer Science*, vol. 4699, pp. 774–780. Springer (2007)

[60] Shannon, C.E.: A mathematical theory of communication. Bell System Technical Journal **27** (1948)

[61] Shapiro, J.M.: Embedded image coding using zerotrees of wavelet coefficients. In: IEEE Transactions of Signal Processing, vol. 41, pp. 3445–3462 (1993)

[62] Sternberg, J., Hinze, M.: A memory-reduced implementation of the Newton-CG method in optimal control of nonlinear time-dependent PDEs. Optimization Methods & Software **25**(4), 553–571 (2010)

[63] Stumm, P., Walther, A.: Multi-stage approaches for optimal offline checkpointing. SIAM J. Sci. Comput. **31**(3), 1946–1967 (2009)

[64] Stumm, P., Walther, A.: New algorithms for optimal online checkpointing. SIAM J. Sci. Comput. **32**(1), 836–854 (2010)

[65] Sweldens, W.: The lifting scheme: A construction of second generation wavelets. SIAM Journal on Mathematical Analysis **29**(2), 511–546 (1998)

[66] Szymczak, A.: Optimized edgebreaker encoding for large and regular triangle meshes. In: DCC '02 Proceedings, p. 472. IEEE Computer Society, Washington, DC, USA (2002)

[67] Teran, R.I., Thole, C.A., Lorentz, R.: New developments in the compression of LS-DYNA simulation results using FEMZIP. 6th European LS-DYNA Users' Conference (2007). URL `https://www.dynalook.com/european-conf-2007/new-developments-in-the-compression-of-ls-dyna.pdf`

[68] Thole, C.A.: Compression of LS-DYNA3D$^{\text{TM}}$ simulation results using FEMZIP©. 3. LS-DYNA Anwenderforum (2004)

[69] Touma, C., Gotsman, C.: Triangle mesh compression. In: Graphics Interface Conference Proceedings, pp. 26–34 (1998)

[70] Tröltzsch, F., Volkwein, S.: POD a-posteriori error estimates for linear-quadratic optimal control problems. Comput. Optim. Appl. **44**, 83–115 (2009)

[71] Tutte, W.: A census of planar triangulations. Canadian Journal of Mathematics **14**, 21–38 (1962)

[72] von Tycowicz, C., Kälberer, F., Polthier, K.: Context-based coding of adaptive multiresolution meshes. Computer Graphics Forum **30**(8), 2231–2245 (2011). DOI 10.1111/j.1467-8659.2011.01972.x. URL `http://dx.doi.org/10.1111/j.1467-8659.2011.01972.x`

[73] Unat, D., Hromadka, T., Baden, S.: An adaptive sub-sampling method for in-memory compression of scientific data. In: Data Compression Conference, 2009. DCC '09, pp. 262–271. IEEE (2009)

[74] Valette, S., Prost, R.: Wavelet-based progressive compression scheme for triangle meshes: Wavemesh. IEEE Transactions on Visualization and Computer Graphics **10**(2) (2004)

[75] van der Vorst, H.A.: Bi-CGSTAB: A fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems. SIAM J. Sci. Stat. Comput. **13**, 631–644 (1994)

[76] Walther, A.: Program reversal schedules for single-and multi-processor machines. Ph.D. thesis, Institute of Scientific Computing, Technical University Dresden, Germany (1999)

[77] Wang, Q., Moin, P., Iaccarino, G.: Minimal repetition dynamic checkpointing algorithm for unsteady adjoint calculation. SIAM Journal on Scientific Computing **31**(4), 2549–2567 (2009)

[78] Weiser, M., Götschel, S.: State trajectory compression for optimal control with parabolic PDEs. SIAM J. Scientific Computing **34**(1), A161–A184 (2012)

[79] Yserentant, H.: On the multi-level splitting of finite element spaces. Numer. Math. **49**(4), 379–412 (1986)