Hamburger Beiträge
zur Angewandten Mathematik

Construction of nested reversal schedules by dynamic programming for the optimal control of ODEs

Julia Sternberg and Andreas Griewank

Nr. 2007-05
March 2007

Construction of nested reversal schedules by dynamic programming for the optimal control of ODEs

Julia Sternberg
Department Mathematik,
University of Hamburg
D-20146 Hamburg, Germany,
email: sternberg@math.uni-hamburg.de

Andreas Griewank
Institut für Mathematik,
Humboldt-Universität zu Berlin
D-10099 Berlin, Germany,
griewank@mathematik.hu-berlin.de

Abstract

We consider the solution of optimal control problems for evolution equations by the method that linearizes the first-order necessary optimality conditions and solves the resulting linear problem exactly by factorizing and substitution. Some types of stable and efficient implementations of this method represent a triple sweep trough a discretized time interval. The straightforward implementation of such a triple sweep requires an enormous memory amount. The remedy for this problem is the application of nested reversal schedules. However, the computation of suitable nested reversal schedules may represent a bottleneck in itself. This paper discusses an efficient heuristic for solution of this combinatorial problem. The temporal and spatial complexity of this heuristic is linear w.r.t. given parameters. An optimal control problem of laser surface hardening of steel is solved using a so called Pantoja method with application of nested reversal schedules. This example can be seen as a some kind of motivating example for usefulness of nested reversal schedules. We report the memory savings that can be achieved by application of nested reversal schedules to the optimal control problem of laser surface hardening of steel.

Key words. Optimal control, Riccati/Pantoja method, nested checkpointing

1 Introduction

Consider the following unconstrained control problem

$$\min_{\mathbf{u}} \phi(\mathbf{x}(T)), \tag{1}$$

where the system evolution is described by

$$\dot{\mathbf{x}} = f(\mathbf{x}(t), \mathbf{u}(t), t), \quad \mathbf{x}(0) = \mathbf{x}_0. \tag{2}$$

Here, $\mathbf{x} : [0,T] \to \mathbb{R}^n$, $\mathbf{u} : [0,T] \to \mathbb{R}^m$, $f : \mathbb{R}^n \times \mathbb{R}^m \times [0,T] \to \mathbb{R}$, $\phi : \mathbb{R}^n \to \mathbb{R}$. In the present paper we consider the non-constrained case of the optimal control problem (1) - (2), i.e. there are no state or control constraints. The task is to find the function $\mathbf{u}(t)$ that minimizes (1). In order to characterize an optimal control function $\mathbf{u}(t)$ for the minimization problem (1) and (2) we consider the following adjoint state equation

$$\dot{\bar{\mathbf{x}}} = -H_{\mathbf{x}}^T = -f_{\mathbf{x}}^T\, \bar{\mathbf{x}}, \qquad \bar{\mathbf{x}}(T) = \phi_{\mathbf{x}}^T(\mathbf{x}(T)), \tag{3}$$

where the **Hamiltonian** function H, is given by

$$H(\mathbf{x}(t), \mathbf{u}(t), \bar{\mathbf{x}}(t), t) = \bar{\mathbf{x}}^T(t)\, f(\mathbf{x}(t), \mathbf{u}(t), t). \tag{4}$$

At each point along the solution path the Hamiltonian function must be minimal with respect to the control value $\mathbf{u}(t)$. Therefore, we have the **First Order Necessary Optimality Condition**

$$\left(\frac{\partial H(\mathbf{x}(t), \mathbf{u}(t), \bar{\mathbf{x}}(t), t)}{\partial \mathbf{u}} \right)^T = 0, \qquad 0 \le t \le T, \tag{5}$$

for the optimal control problem (1) - (2). Now, we need two assumptions in order to formulate the **Second Order Sufficient Conditions** for unconstrained control problems.

Assumption 1.1. Strict Legendre-Clebsch Condition
The relation

$$H_{\mathbf{uu}}(t) \succ 0 \tag{6}$$

is valid on the whole interval $[0,T]$ in a $W^{1,\infty}$ vicinity of the optimal trajectory, which means that the second derivative $H_{\mathbf{uu}}(t)$ of the Hamiltonian function $H(t)$ w.r.t. the control $\mathbf{u}$ is positive definite along the interval $[0,T]$.

Assumption 1.2. Coercivity Condition
There exists a solution of the Riccati equation

$$\dot{Q} = -Qf_{\mathbf{x}}(t) - f_{\mathbf{x}}^T(t)Q - H_{\mathbf{xx}}(t) + (H_{\mathbf{xu}}(t) + Qf_{\mathbf{u}}(t))\, H_{\mathbf{uu}}^{-1}(t)\, (H_{\mathbf{xu}}(t) + Qf_{\mathbf{u}}(t))^\top \tag{7}$$

that is bounded on $[0,T]$ and satisfies the boundary condition

$$Q(T) = \phi_{\mathbf{xx}}^\top(\mathbf{x}(T)). \tag{8}$$

The next theorem summarizes the SSC for a weak local minimum which are to be found in [9, 11, 14, 21].

Theorem 1.3. SSC for Control Problems
Let $(\mathbf{x}_0, \mathbf{u}_0)$ *be admissible for the problem (1) - (2). Suppose, there exist multipliers* $\bar{\mathbf{x}} \in W^{1,\infty}(0,T;\mathbb{R}^n)$*, so that the Euler-Lagrange equations (2) - (3) and Assumptions 1.1 and 1.2 are satisfied. Then, there exist* $c > 0$ *and* $\alpha > 0$*, so that*

$$\phi(\mathbf{x},\mathbf{u}) \geq \phi(\mathbf{x}_0,\mathbf{u}_0) + c\left\{\|\mathbf{x}-\mathbf{x}_0\|_{1,2}^2 + \|\mathbf{u}-\mathbf{u}_0\|_2^2\right\} \tag{9}$$

holds for all admissible $(\mathbf{x},\mathbf{u})$ *with* $\|\mathbf{x}-\mathbf{x}_0\|_{1,\infty} + \|\mathbf{u}-\mathbf{u}_0\|_\infty \leq \alpha$*. In particular,* $(\mathbf{x}_0,\mathbf{u}_0)$ *provides a strict weak local minimum for the problem (1) - (2).*

In this paper an indirect approach for the solution of optimal control problem (1) - (2) will be considered. The resulting BVP (2) - (3) coupled with the algebraic constraint (5) will be iteratively solved using the so called quasilinearization technique. The BVP (2) - (3) is in general non-linear with separated BC. Firstly, we linearize (2) - (3) and (5); after that we solve the resulting linear BVP using the Riccati approach. It is shown in [16] that the particular discretized quasilinearization scheme specified in this thesis is equivalent to the so called Pantoja approach [12, 13]. Pantoja approach describes an efficient stage-wise construction of the Newton step for a discrete optimal control problem. It was established in [18] how the quasilinearization scheme can be implemented using nested checkpointing techniques. The nested checkpointing techniques offer the possibility to reduce the storage requirement. In the present paper suitable nested reversal schedules will be constrained.

This paper is organized as follows. Section 2 introduces the quasilinearization scheme and Pantoja algorithm. Nested checkpointing techniques and their main properties are discusses in Section 3.1. In Section 3.2 and Section 3.3 an algorithm based on the dynamic programming is introduced. This algorithm can be applied for the construction of suitable nested reversal schedules. Section 4 gives a numerical example which demonstrates the operation mode of Pantoja approach and advantages of nested checkpointing techniques. Moreover, this example confirms the analytic estimation for the growth of the run-time complexity. Finally, in Section 5 we present some conclusions.

2 Quasilinearization Techniques

In the present section we introduce the quasilinearization techniques. It can be applied for a stable solution of the optimal control problem (1) - (2). The Newton-step $\delta\mathbf{u}$ is to determine as a solution of a linear-quadratic regulator problem, resulting as an approximation of the given nonlinear control problem about a currently available pair of primal and dual trajectories. The exact solution of this problem requires the integration of the matrix Riccati equation, which involves in particular first and second derivatives of the Hamiltonian function.

2.1 Quasilinearization/Pantoja Algorithm

We linearize (2), (3), (5) about a reference solution $\mathbf{x}(t)$, $\mathbf{u}(t)$, $\bar{\mathbf{x}}(t)$ and obtain the following equations for variations $\delta\mathbf{x}(t)$, $\delta\bar{\mathbf{x}}(t)$, and $\delta\mathbf{u}(t)$

$$\delta\dot{\mathbf{x}} - f_{\mathbf{x}}\delta\mathbf{x} - f_{\mathbf{u}}\delta\mathbf{u} = 0, \tag{10}$$

$$\delta\dot{\bar{\mathbf{x}}} + H^T_{\mathbf{xx}}\delta\mathbf{x} + H^T_{\mathbf{xu}}\delta\mathbf{u} + H^T_{\mathbf{x\bar{x}}}\delta\bar{\mathbf{x}} = 0, \tag{11}$$

$$H^T_{\mathbf{u}} + H^T_{\mathbf{ux}}\,\delta\mathbf{x} + H^T_{\mathbf{uu}}\,\delta\mathbf{u} + H^T_{\mathbf{u\bar{x}}}\,\delta\bar{\mathbf{x}} = 0, \tag{12}$$

with the linearized initial and terminal conditions

$$\begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} \delta\mathbf{x}(0) \\ \delta\bar{\mathbf{x}}(0) \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ -\phi_{\mathbf{xx}}(T) & I \end{pmatrix}\begin{pmatrix} \delta\mathbf{x}(T) \\ \delta\bar{\mathbf{x}}(T) \end{pmatrix} = -\begin{pmatrix} \mathbf{x}(0) - \mathbf{x}_0 \\ \bar{\mathbf{x}}(T) - \phi^T_{\mathbf{x}}(T) \end{pmatrix}. \tag{13}$$

Due to the Assumption 1.1 $H_{\mathbf{uu}} \succ 0$, i.e. the Hessian $H_{\mathbf{uu}}$ has a bounded inverse. Then, it is possible to express $\delta\mathbf{u}$ in terms of $\delta\mathbf{x}$ and $\delta\bar{\mathbf{x}}$ from the relation (12). Thus, we obtain

$$\delta\mathbf{u} = -H^{-1}_{\mathbf{uu}}\,\left(H^T_{\mathbf{u}} + H^T_{\mathbf{ux}}\,\delta\mathbf{x} + H^T_{\mathbf{u\bar{x}}}\,\delta\bar{\mathbf{x}}\right). \tag{14}$$

Substitution of this expression into the (10) - (11) yields the following linear BVP

$$\begin{pmatrix} \delta\dot{\mathbf{x}} \\ \delta\dot{\bar{\mathbf{x}}} \end{pmatrix} = S(t)\begin{pmatrix} \delta\mathbf{x} \\ \delta\bar{\mathbf{x}} \end{pmatrix} + q(t), \tag{15}$$

where

$$S(t) = \begin{pmatrix} S^{11} & S^{12} \\ S^{21} & S^{22} \end{pmatrix} = \left(\begin{array}{cc|c} f_{\mathbf{x}} - f_{\mathbf{u}}\,H^{-1}_{\mathbf{uu}}\,H_{\mathbf{xu}} & & -f_{\mathbf{u}}\,H^{-1}_{\mathbf{uu}}\,f^T_{\mathbf{u}} \\ -H_{\mathbf{xx}} + H_{\mathbf{ux}}\,H^{-1}_{\mathbf{uu}}\,H_{\mathbf{xu}} & & H_{\mathbf{ux}}\,H^{-1}_{\mathbf{uu}}\,f^T_{\mathbf{u}} - f^T_{\mathbf{x}} \end{array}\right) \tag{16}$$

is the system matrix and

$$q(t) = \begin{pmatrix} q^1 \\ q^2 \end{pmatrix} = \begin{pmatrix} -f_{\mathbf{u}}\,H^{-1}_{\mathbf{uu}}\,H^T_{\mathbf{u}} \\ H_{\mathbf{ux}}\,H^{-1}_{\mathbf{uu}}\,H^T_{\mathbf{u}} \end{pmatrix} \tag{17}$$

is the non-homogeneous part. The BC for this problem are given by the relation (13). Rather than solving this linear BVP using collocation or another 'global' discretization scheme we prefer the Riccati approach which allows the solution in a sequence of forward and backward sweeps through the time interval $[0, T]$. In order to achieve a suitable decoupling of the solution components we consider a linear transformation of the form

$$\begin{pmatrix} \delta\mathbf{x}(t) \\ \delta\bar{\mathbf{x}}(t) \end{pmatrix} = \begin{pmatrix} I & 0 \\ K(t) & I \end{pmatrix}\begin{pmatrix} \delta\mathbf{x}(t) \\ a(t) \end{pmatrix}. \tag{18}$$

In order to determine an appropriate $K(t)$ and the corresponding $a(t)$ we substitute the $\begin{pmatrix} \delta\mathbf{x} \\ \delta\bar{\mathbf{x}} \end{pmatrix}$ in terms of $\begin{pmatrix} \delta\mathbf{x} \\ a \end{pmatrix}$ into (15), which yields

$$\frac{d}{dt}\begin{pmatrix} I & 0 \\ K & I \end{pmatrix}\begin{pmatrix} \delta\mathbf{x} \\ a \end{pmatrix} = \begin{pmatrix} S^{11} & S^{12} \\ S^{21} & S^{22} \end{pmatrix}\begin{pmatrix} I & 0 \\ K & I \end{pmatrix}\begin{pmatrix} \delta\mathbf{x} \\ a \end{pmatrix} + \begin{pmatrix} q^1 \\ q^2 \end{pmatrix}. \tag{19}$$

Now, if we choose $K(t)$ as the solution of the Riccati equation

$$\dot{K}(t) = S^{21} - K(t)S^{11} + S^{22}K(t) - K(t)S^{12}K(t), \tag{20}$$

then it can be derived that the new variables $(\delta\mathbf{x}, a)$ satisfy the block-triangular system

$$\begin{pmatrix} \delta\dot{\mathbf{x}} \\ \dot{a} \end{pmatrix} = \left(\begin{array}{cc|c} & S^{11} + S^{12}K & S^{12} \\ & 0 & S^{22} - KS^{12} \end{array}\right)\begin{pmatrix} \delta\mathbf{x} \\ a \end{pmatrix} + \begin{pmatrix} q^1 \\ q^2 - Kq^1 \end{pmatrix}, \tag{21}$$

with the separated BC

$$\begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} \delta\mathbf{x}(0) \\ a(0) \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ -\phi_{\mathbf{xx}}(T) + K(T) & I \end{pmatrix}\begin{pmatrix} \delta\mathbf{x}(T) \\ a(T) \end{pmatrix} = -\begin{pmatrix} \mathbf{x}(0) - \mathbf{x}_0 \\ \bar{\mathbf{x}}(T) - \phi_{\mathbf{x}}^T(T) \end{pmatrix}. \tag{22}$$

This approach leads to a quasilinearization algorithm presented in [16, 18]. An explicit one-step method for the solution of ODEs, e.g. the Euler method, turns (20) and (21) into a discrete dynamical system of the form:

$$\left(S_i^{22} - K_{i+1}S_i^{12}\right)K_i = \left(-S_i^{21} + K_{i+1}S_i^{11}\right), \tag{23}$$

and

$$\begin{pmatrix} \delta\mathbf{x}_{i+1} \\ a_{i+1} \end{pmatrix} = \begin{pmatrix} S_i^{11} + S_i^{12}K_i & S_i^{12} \\ 0 & S_i^{22} - K_{i+1}S_i^{12} \end{pmatrix}\begin{pmatrix} \delta\mathbf{x}_i \\ a_i \end{pmatrix} + \begin{pmatrix} q_i^1 \\ q_i^2 - K_{i+1}q_i^1 \end{pmatrix}. \tag{24}$$

Applying the Sherman-Woodbury formula in (23) as it is shown in [16] we obtain the following Pantoja Algorithm 2.1. Pantoja algorithm represents a computationally efficient stage-wise construction of the Newton direction for the discrete-time optimal control problem. Moreover, this scheme can also be viewed as Newton's method applied to the solution of a discretized general non-linear BVP (2), (3), (5) using a particular LU-matrix factorization. These relations are discussed in detail in [16].

In the **Adjoint Step** of the Pantoja algorithm matrix multiplications are performed. These multiplications involve matrices which size is of order n^2. However, matrices $f_{\mathbf{x},i}$ and $f_{\mathbf{xx},i}$ resulting through the FE discretization of the state equations are generally very sparse. Therefore, its storage requirement can be reduced due to matrix compression. Moreover, run-time complexity needed to compute

Algorithm 2.1. Pantoja Algorithm
Choose initial control trajectory $\mathbf{u}_0^0$, ..., $\mathbf{u}_{l-1}^0$, $j = 0$.
Do:
Primal Initialization:
$\mathbf{x}_0^j = \mathbf{x}_0$.
For $i = 0, 1, ..., l-1$ **DO**
Primal Step:
$\mathbf{x}_{i+1}^j = f_i(\mathbf{x}_i^j, \mathbf{u}_i^j)$.
Adjoint Initialization:
Set $\bar{\mathbf{x}}_l^j = \eta_l = \phi_{\mathbf{x}}^\top(\mathbf{x}_l^j) \in \mathbb{R}^n$.
Set $K_l = \phi_{\mathbf{xx}}^\top(\mathbf{x}_l^j) \in \mathbb{R}^{n\times n}$.
For $i = l-1, l-2, ..., 0$ **DO**
Adjoint Step:
Compute $\bar{\mathbf{x}}_i = f_{\mathbf{x},i}^\top \bar{\mathbf{x}}_{i+1} \in \mathbb{R}^n$.
Compute $A_i = f_{\mathbf{x},i}^\top K_{i+1} f_{\mathbf{x},i} + f_{\mathbf{xx},i}^\top \bar{\mathbf{x}}_{i+1} \in \mathbb{R}^{n\times n}$.
Compute $B_i = f_{\mathbf{u},i}^\top K_{i+1} f_{\mathbf{x},i} + f_{\mathbf{ux},i}^\top \bar{\mathbf{x}}_{i+1} \in \mathbb{R}^{m\times n}$.
Compute $C_i = f_{\mathbf{u},i}^\top K_{i+1} f_{\mathbf{u},i} + f_{\mathbf{uu},i}^\top \bar{\mathbf{x}}_{i+1} \in \mathbb{R}^{m\times m}$.
If C_i is indefinite, terminate.
Else
Compute $K_i = A_i - B_i^\top C_i^{-1} B_i \in \mathbb{R}^{n\times n}$.
Compute $c_i = f_{\mathbf{u},i}^\top \eta_{i+1} \in \mathbb{R}^m$.
Compute $\eta_i = f_{\mathbf{x},i}^\top \eta_{i+1} - B_i^\top C_i^{-1} c_i \in \mathbb{R}^n$.
Final Initialization:
Set $\delta\mathbf{x}_0^j = 0$.
For $i = 0, 1, ..., l-1$ **DO**
Final Step:
Compute $\delta\mathbf{u}_i^j = -C_i^{-1}(B_i \delta\mathbf{x}_i^j + c_i) \in \mathbb{R}^m$.
Compute $\delta\mathbf{x}_{i+1}^j = f_{\mathbf{x},i}\delta\mathbf{x}_i^j + f_{\mathbf{u},i}\delta\mathbf{u}_i^j \in \mathbb{R}^n$.
$\mathbf{u}_i^{j+1} = \mathbf{u}_i^j + \delta\mathbf{u}_i^j$.
$j = j + 1$.
While: $\max_{0\le i<l} \|\delta\mathbf{u}_i^j\| \ge$ TOL and $j <$ MAX_ITER.

matrix products can be reduced from $O(n^3)$ to $O(np)$, where p is the number of non-vanishing elements in the matrix $f_{\mathbf{x},i}$ and usually $p \ll n^2$ is valid.

It is shown in [1] that the positive definiteness of C_i in the Pantoja Algorithm 2.1 is equivalent to $H_{\mathbf{uu}} \succ 0$, which means the strict Legendre-Clebsch conditions are satisfied. If C_i fails to be positive definite there are some modifications of the Algorithm 2.1 which ensure the generation of the descent direction. One of this approaches changes the recurrence relation for C_i to

$$C_i = f_{\mathbf{u},i}^{\top} K_{i+1} f_{\mathbf{u},i} + f_{\mathbf{uu},i}^{\top} \bar{\mathbf{x}}_{i+1} + \Lambda_i, \tag{25}$$

where $\Lambda_i \in \mathbb{R}^{m \times m}$ is symmetric and positive definite chosen so that $C_i \succ 0$. More globalization strategies are discussed in [2]. In the numerical example of laser surface hardening of steel used in Section 4 of this paper one has a situation $m = 1$. In the practical implementation of Pantoja Algorithm 2.1 applied to this example we replaced C_i with its absolute value.

In the **Final Step** of the Pantoja algorithm the product $C_i^{-1} B_i$ of two matrices is required. Due to the assumption $m \ll n$ the dimension of the matrices C_i is very small. Thus, matrix-vector and matrix-matrix products required in this step are inexpensive.

For the performance of the Pantoja algorithm 2.1 the positive definiteness of the Hessian $H_{\mathbf{uu}}$ has to be guaranteed, which is incorporated in the equivalent form $C_i \succ 0$ in the execution of this algorithm. Thus, the Assumption 1.1 from the Section 1 is satisfied. Moreover, the Assumption 1.2 is satisfied in that the bounded solution of the Riccati equation (7) is stepwise constructed during the execution of the Pantoja algorithm 2.1. The Theorem 1.3 then guarantees the existence of a strict local minimum for the optimal control problem (1) - (2). Thus, if a starting point $\mathbf{u}^0$ lies sufficiently close to a local minimum, the Algorithm 2.1 successfully terminates producing a local minimum $\mathbf{u}^\star$. The unmodified Pantoja algorithm is just an implementation of Newton's method and thus quadratically convergent.

2.2 Information Flow in Quasilinearization/Pantoja Algorithm

From the Algorithm 2.1 we can see, that each iteration of the Pantoja algorithm consists of three sweeps through the time window $[0, T]$, which are referred to as **primal**, **adjoint** and **final** sweep. In Figure 1 the dimensions of the data objects flowing between these three sweeps are shown.

The horizontal arrows represent informational flow between the three sweeps that are shown by slanted lines. Here $B(t)$ is a $n \times m$ matrix path that must be communicated from the adjoint to the final sweep. Two cameras pointing at the primal and adjoint sweeps represent the information which has to be stored, if the current composite state is saved as a checkpoint.

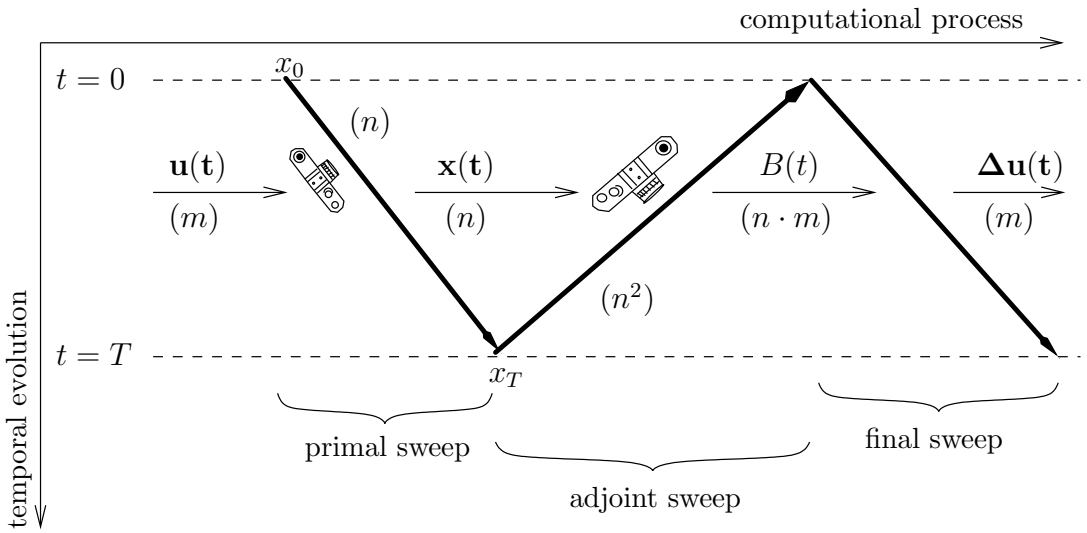

Figure 1: Information flow for Riccati/Pantoja Computation of Newton step

In any case the adjoint sweep causes much more computational effort than the primal and the final sweeps, because it involves matrix computations and factorizations. The final sweep proceeds forward in time and propagates vectors of dimension $(n + m)$. Computations on the final sweep proceed as soon as required information from the previous sweeps is available. The final sweep can be combined with the primal sweep of a subsequent Newton step.

The simplest strategy is to implement the Algorithm 2.1 straightforwardly storing all intermediate states of each sweep on a sequential data file and to restore them when they are needed. The memory requirement for the basic algorithm, where all intermediate values are stored, is of order $\mathcal{O}(l\, n^2)$, where l gives the number of time steps between 0 and T. But this approach can be realized only when there is a sufficiently large amount of memory available. If this is not the case then we can apply checkpointing techniques.

As developed in [3, 4, 15], checkpointing means that not all intermediate states are saved but only a small subset of them is stored as checkpoints. In previous work we have treated cases where checkpoints are stored only for a reversal consisting of a single forward and an adjoint, or reverse sweep. But because of the triple sweep within each Newton iteration (see Algorithm 2.1 and Figure 1) we are faced here with a new kind of checkpointing task, which will be discussed in the following section.

3 Nested Reversal Schedules

In the present section we introduce a formal concept for nested checkpointing. Then, some heuristics are discussed in order to construct admissible nested reversal sched-

ules. For the realization of these heuristics an efficient algorithm is developed, that is detailed explained in the following. Finally in this section, we investigate properties of by this algorithm constructed nested reversal schedules.

3.1 Formalism

Consider a multiple sweep evolution $\mathcal{E}_{3\times l}$ which contains three alternative sweeps. Each sweep consists of l consecutive time steps. An example of such an evolution is shown in Figure 2. Time steps are shown as horizontal arrows. Their directions denote the information flow. Nodes denote different intermediate states. Each

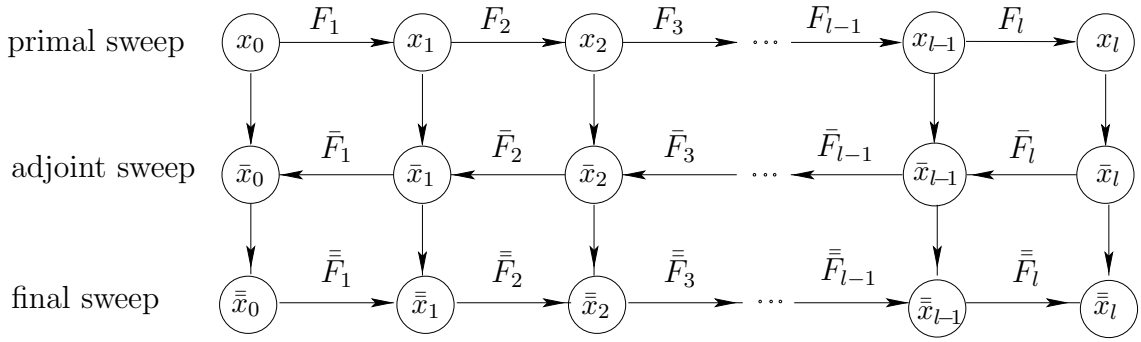

Figure 2: Multiple sweep evolution $\mathcal{E}_{3\times l}$

sweep is characterized by one specified direction, i.e. the direction of horizontal arrows within a single sweep. It shows the corresponding information flow between neighboring intermediate states of a single sweep. Note that the information flow within a single sweep has a constant direction. An additional information flow exists between nodes being intermediate states of different successive sweeps. This is shown by vertical lines in Figure 2.

We denote intermediate states of the primal, adjoint, and final sweeps by x_i, $\bar{x}_i$, and $\bar{\bar{x}}_i$, $0 \le i \le l$, respectively. In the same manner, we identify time steps of various sweeps with F_i, $\bar{F}_i$, and $\bar{\bar{F}}_i$, $1 \le i \le l$. Then, we obtain

$$x_i = F_i(x_{i-1}), \quad \bar{x}_{i-1} = \bar{F}_i(x_{i-1}, \bar{x}_i), \quad \bar{\bar{x}}_i = \bar{\bar{F}}_i(\bar{x}_i, \bar{\bar{x}}_{i-1}), \quad 1 \le i \le l. \tag{26}$$

We assume that sizes of intermediate states are constant within a single sweep. Therefore, we denote them

$$d \equiv \text{size}\,\{x_i\}, \quad \bar{d} \equiv \text{size}\,\{\bar{x}_i\}, \quad \bar{\bar{d}} \equiv \text{size}\,\{\bar{\bar{x}}_i\}. \tag{27}$$

Typically we have the situation:

$$\text{size}\,\{x_i\} = \mathcal{O}(n), \quad \text{size}\,\{\bar{x}_i\} = \mathcal{O}(n^2), \quad \text{size}\,\{\bar{\bar{x}}_i\} = \mathcal{O}(n). \tag{28}$$

Moreover, we introduce evaluation costs, i.e. the computational effort for time steps of different sweeps. We assume that within each single sweep we have uniform step costs, i.e. three constants t, $\bar{t}$, and $\bar{\bar{t}}$ exist there, so that

$$t \equiv TIME(F_i), \quad \bar{t} \equiv TIME(\bar{F}_i), \quad \bar{\bar{t}} \equiv TIME(\bar{\bar{F}}_i), \quad 1 \leq i \leq l. \tag{29}$$

Furthermore, we assume that

$$\bar{d} \gg d \qquad \text{and} \qquad \bar{t} \gg t. \tag{30}$$

Thus, the size $\bar{d}$ of the adjoint state is much larger than the size d of the primal state. Correspondingly, the evaluation of adjoint steps is much more extensive than the evaluation of primal steps. The relations (26) and the assumptions (30) agree with the scenario presented in Algorithm 2.1 and Figure 1.

The aim is to implement an evolution $\mathcal{E}_{3\times l}$ using nested checkpointing. The problem is how to place different checkpoints in order to implement the evolution $\mathcal{E}_{3\times l}$ most efficiently. We call each possible strategy a **nested reversal schedule** because checkpoints are set and released at two different levels. Since the information to be stored on the primal sweep differs from that needed on the adjoint sweep, we have two classes of checkpoints. Hence, we call the checkpoints **thin** on the primal sweep and **fat** on the adjoint sweep. Thin checkpoints typically have the size of the state space dimension n, and fat checkpoints the size of order n^2, i.e.

$$\text{size(thin)} \;=\; \text{size}(x_i) = d = \mathcal{O}(n), \qquad i = 0, ..., l, \tag{31}$$

$$\text{size(fat)} \;=\; \text{size}(\bar{x}_i) = \bar{d} = \mathcal{O}(n^2), \qquad i = 0, ..., l. \tag{32}$$

While the length of steps may vary arbitrarily with respect to the physical time increment they represent, we assume throughout that the total number l of time steps is a priori known. When this is not the case, an upper bound on l may be used, which results of course in some loss of efficiency. Fully adaptive nested reversal schedules are under development.

Let us first ignore the third sweep and recapitulate the known results for a simple reversal with checkpointing. Figure 3 illustrates an example of a reversal schedule **S** with ten primal and adjoint steps and three available checkpoints. The temporal complexity for primal and adjoint steps is given by $t = 1$ and $\bar{t} = 2$, respectively. Primal steps F_i, $1 \leq i \leq 10$, are plotted along the vertical axis, whereas time required for the evaluation of single actions is plotted along the horizontal axis. Time is measured in t-units. Horizontal lines in Figure 3 represent checkpoints. Since the initial checkpoint contains data of the primal state x_0 and lies on the computational axis, this axis represents a checkpoint itself.

The execution of the reversal schedule **S** in Figure 3 can be described as follows. Firstly, the initial primal state x_0 is read out from the first checkpoint. After that, the four primal steps F_1, F_2, F_3, and F_4 are evaluated consecutively. Then, the primal state x_4 is stored into the second checkpoint. The following three primal

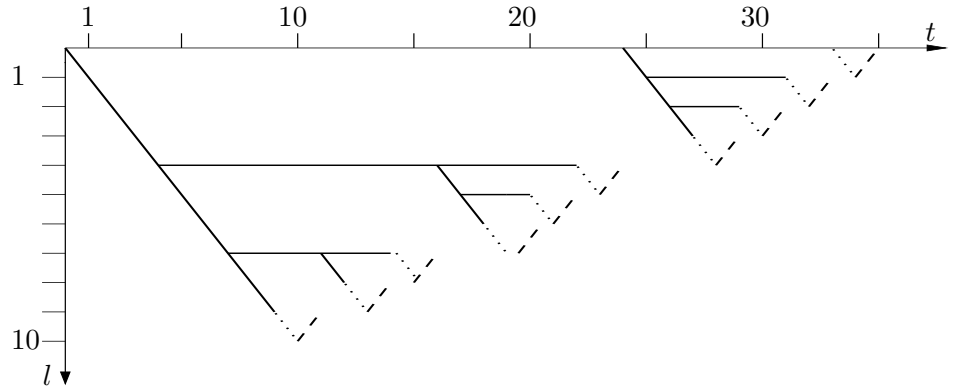

Figure 3: Reversal schedule **S** with ten primal and adjoint steps and three checkpoints

steps F_5, F_6, and F_7 are carried out successively. The primal state x_7 is stored into the third checkpoint. The following two primal steps F_8 and F_9 are carried out consecutively. After that, the adjoint step $\bar{F}_{10}$ is evaluated. Then, data of the primal state x_7 is read out from the third checkpoint. Using this data, the primal step F_8 is evaluated. Now, it is possible to perform the adjoint step $\bar{F}_9$. After that, the same principle is applied several times until all adjoint steps $\bar{F}_1, ..., \bar{F}_{10}$ are performed successively.

The reversal schedule **S** in Figure 3 can be executed using 35 t-units and represents a binomial reversal schedule (see [5, 15] for a detailed explanation). Binomial reversal schedules absolutely minimize the overall evaluation cost by simple reversal for given number l of time steps and c available checkpoints. There are some explicit representations for the minimal evaluation cost $\mathbf{T}_{bin}(l, c)$ of binomial reversal schedules. One of them is

$$\mathbf{T}_{bin}(l, c) = rl - \beta(c + 1, r - 1), \tag{33}$$

with $r = r_{bin}(c, l)$ being the unique integer satisfying

$$\frac{(c + r - 1)!}{c!\,(r - 1)!} = \beta(c, r - 1) \;<\; l \;\leq\; \beta(c, r) = \frac{(c + r)!}{c!\,r!}. \tag{34}$$

For the derivation of this expression see [4, 5]. The idea of binomial reversal schedules and its evaluation cost is crucial by the construction of nested reversal schedules, which will be describe in the following.

Obviously, for the implementation of a triple sweep evolution $\mathcal{E}_{3\times l}$ using nested checkpointing it is not required to store intermediate states of the final sweep as checkpoints since information computed during this sweep is required just for subsequent time steps within this sweep but not for previous sweeps. If only a restricted amount of memory is available, it is convenient to measure its size in terms of the

number of fat checkpoints which can be accommodated by them. Moreover, it is assumed that one thin and one fat checkpoints are available additionally. The initial primal state and the final adjoint state are to be kept in these extra checkpoints throughout the execution of a nested reversal schedule.

Since fat checkpoints, i.e. checkpoints of the size $\bar{d}$, have to be stored during the adjoint sweep, we can use available memory on the primal sweep to store thin checkpoints, i.e. checkpoints of the size d, to reduce the total number of evaluated primal steps F_i. A parameter c denotes how many thin checkpoints can be stored in place of a fat one. Obviously, the following is valid

$$\text{size(fat)} = \bar{d} = c\,\text{size(thin)} = c\,d, \qquad \Rightarrow \qquad c = \left\lfloor \frac{\bar{d}}{d} \right\rfloor . \tag{35}$$

On the adjoint sweep we successively remove thin checkpoints and store fat checkpoints instead of them. This has to be done as soon as required memory is available, i.e. as soon as a sufficient number of thin checkpoints is removed. Each nested reversal schedule that successfully implement an evolution $\mathcal{E}_{3\times l}$ with fixed properties using given resources, i.e. memory available for storing a certain number C of fat checkpoints, is called an admissible one. Such schedules will be denoted by $\mathbf{S}$ throughout this paper. For the formal definition of an admissible nested reversal schedule $\mathbf{S}$ see [16].

For an admissible nested reversal schedule $\mathbf{S}$ the **repetition numbers** $r_i \equiv r(i)$, $\bar{r}_i \equiv \bar{r}(i)$, and $\bar{\bar{r}}_i \equiv \bar{\bar{r}}(i)$, defined by the functions

$$r, \bar{r}, \bar{\bar{r}} : [1, l] \to \mathbb{N}, \tag{36}$$

count how often the ith primal, the ith adjoint, and the ith final step, is evaluated during the execution of the nested reversal schedule $\mathbf{S}$.

Figure 4 shows one example for an admissible nested reversal schedule $\mathbf{S}$ for an evolution $\mathcal{E}_{3\times 9}$ with the size distribution $\vec{\mathbf{d}}_{\mathbf{3\times 9}} = (d, \bar{d}, \bar{\bar{d}})^\top = (1, 1, 1)^\top$. $\mathcal{E}_{3\times 9}$ contains three sweeps each of which consists of nine time steps. Two fat checkpoints are available, i.e. two adjoint states can be stored except for the final adjoint state $\bar{x}_9$ which data is stored into the extra fat checkpoint. Moreover, one thin checkpoint can be stored in place of a single fat one, i.e. $c = 1$. Furthermore, the initial primal state x_0 is stored into the extra thin checkpoint. In Figure 4 time steps are plotted along the vertical axis. Time required for the implementation of the evolution $\mathcal{E}_{3\times 9}$ measured in number of executed steps is represented by the horizontal axis. Hence, the horizontal axis can be thought of as a computational axis. Each solid thin horizontal line including the horizontal axis itself represents a thin checkpoint, i.e. a checkpoint of the size $d = 1$. Each solid thick horizontal line represents a fat checkpoint, i.e. a checkpoint of the size $\bar{d} = 1$. Solid slanted thin lines represent primal steps $F_i,\ i = 1, \ldots, 9$, whereas adjoint steps $\bar{F}_i,\ i = 1, \ldots, 9$, are visualized by dotted slanted lines. Final steps $\bar{\bar{F}}_i,\ i = 1, \ldots, 9$, are drawn by slanted dashed-dotted thick lines.

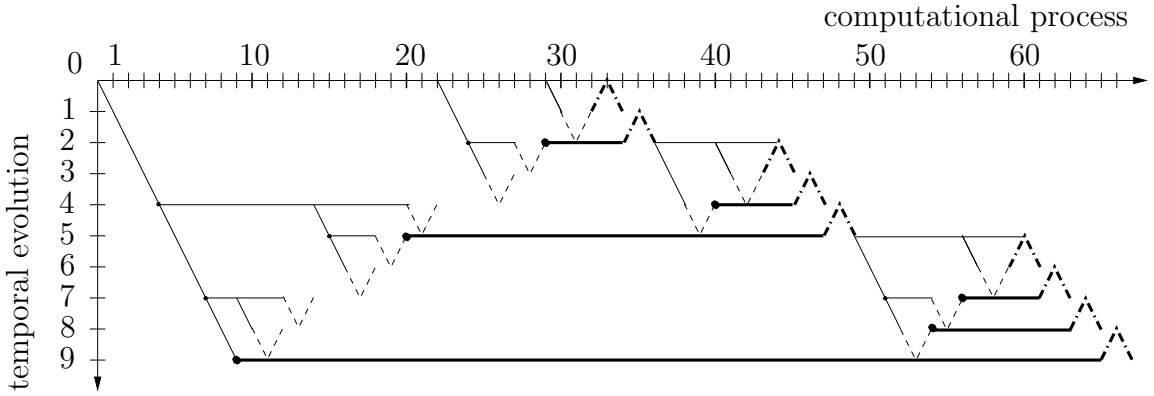

Figure 4: An example for nested reversal schedules $\mathbf{S}$

The nested reversal schedule $\mathbf{S}$ starts with restoring the initial primal state x_0 from the zero-th thin checkpoint. Then, four primal steps F_1, F_2, F_3, and F_4 are performed consecutively. The primal state x_4 is stored into the first thin checkpoint. Now, again, three primal steps F_5, F_6, and F_7 are performed, and the primal state x_7 is stored into the second thin checkpoint. Subsequently, the primal steps F_8 and F_9 are executed and the adjoint state $\bar{x}_9$ is initialized. The adjoint state $\bar{x}_9$ is stored into the extra fat checkpoint. The adjoint sweep is started by this action.

Subsequently, data of the primal state x_7 is restored from the second thin checkpoint, the primal step F_8 is performed, and the adjoint states $\bar{x}_8$ and $\bar{x}_7$ can be evaluated. Then, the primal state x_4 is restored from the first thin checkpoint, the primal state x_5 is reevaluated, and its data is stored into the second thin checkpoint. After that, the primal state x_6 is evaluated. Then, the adjoint states $\bar{x}_6$ and $\bar{x}_5$ are evaluated. The adjoint state $\bar{x}_5$ is stored into the first fat checkpoint. Then, we go back to the adjoint state $\bar{x}_2$ by reevaluating required primal states and storing the adjoint state $\bar{x}_2$ into the second fat checkpoint. Subsequently, one goes back to the adjoint state $\bar{x}_1$ by reevaluating required intermediate states. Consequently, the first final step $\bar{\bar{F}}_1$ is performed. Subsequently, we store the current initial primal state x_1 into the extra thin checkpoint and continue in the same manner in order to execute all other final steps $\bar{\bar{F}}_2, \ldots, \bar{\bar{F}}_9$.

Using the nested reversal schedule $\mathbf{S}$ in Figure 4, it is necessary to perform 23 primal steps F_i, 13 adjoint steps $\bar{F}_i$, and nine final steps $\bar{\bar{F}}_i$. For this reversal schedule $\mathbf{S}$ the corresponding nested repetition profile can be defined by

$$\mathbf{r}(\mathbf{S}) = \begin{pmatrix} r_1 & \ldots & r_9 \\ \bar{r}_1 & \ldots & \bar{r}_9 \\ \bar{\bar{r}}_1 & \ldots & \bar{\bar{r}}_9 \end{pmatrix} = \begin{pmatrix} 3 & 2 & 4 & 2 & 2 & 4 & 2 & 3 & 1 \\ 0 & 1 & 1 & 2 & 2 & 1 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}. \tag{37}$$

This profile is illustrated graphically in Figure 5. Each graph in Figure 5 corresponds to a single row of the matrix $\mathbf{r}(\mathbf{S})$ in (29), i.e. to a single sweep of the multiple sweep evolution $\mathcal{E}_{3\times 9}$. x-axis of each graph gives a number i of a corresponding time step F_i, $\bar{F}_i$, $\bar{\bar{F}}_i$, $1 \leq i \leq 9$. y-axis gives a corresponding repetition number r_i, $\bar{r}_i$, or $\bar{\bar{r}}_i$

for a specified time step F_i, $\bar{F}_i$, or $\bar{\bar{F}}_i$, respectively. The vector $\vec{\mathbf{r}}_{max}(\mathbf{S})$ of maximal repetition numbers reaches $\vec{\mathbf{r}}_{max}(\mathbf{S}) = (4, 2, 1)^\top$.

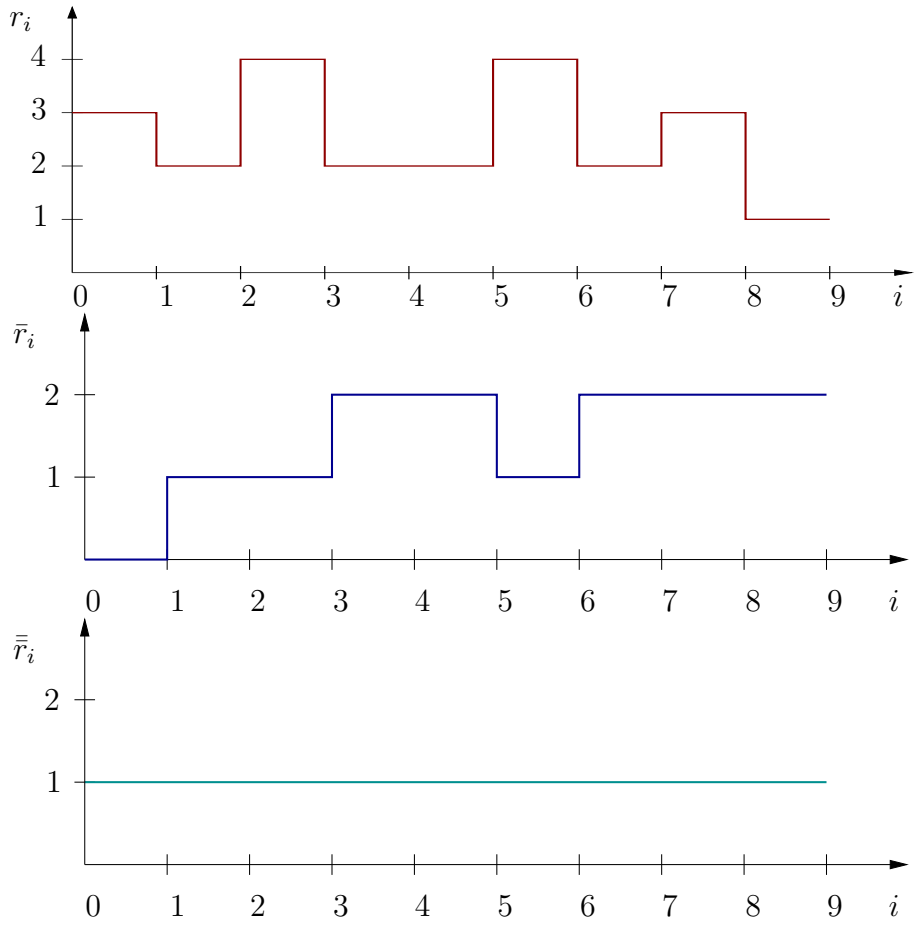

Figure 5: Repetition profiles for $\mathbf{S}$ in Figure 4

Provided a schedule is admissible, its total runtime complexity can be computed from the additional problem parameters $\vec{\mathbf{t}}_{\mathbf{3\times 1}} = (t, \bar{t}, \bar{\bar{t}})^\top$ by

$$\mathbf{T}(\mathbf{S}) = t \sum_{i=1}^{l} r_i + \bar{t} \sum_{i=1}^{l} \bar{r}_i + \bar{\bar{t}} \sum_{i=1}^{l} \bar{\bar{r}}_i. \tag{38}$$

The temporal complexity of the nested reversal schedule $\mathbf{S}$ in Figure 4 assuming the parameters $\vec{\mathbf{t}}_{\mathbf{3\times 9}} = (1, 5, 1)^\top$ is $\mathbf{T}(\mathbf{S}) = 1 \times 23 + 5 \times 13 + 1 \times 9 = 97$.

The optimal nested reversal schedule from the set of all admissible nested reversal schedules is required to minimize the evaluation cost, i.e. to achieve

$$\mathbf{T}_{min}(l, C) \equiv \min \{\mathbf{T}(\mathbf{S}),\ \mathbf{S} \text{ is admissible}\}. \tag{39}$$

The set of optimal nested reversal schedules is denoted $\mathbf{S}_{min}(l, C)$, so that

$$\mathbf{T}(\mathbf{S}_{min}(l, C)) \equiv \mathbf{T}_{min}(l, C). \tag{40}$$

For the construction of an appropriate optimal nested reversal schedule $\mathbf{S}_{min}(l, C)$ and for the computation of its evaluation cost $\mathbf{T}_{min}(l, C)$ see [16]. In this thesis exhaustive search technique is developed. The nested reversal schedule $\mathbf{S}$ in Figure 4 is an optimal one for the given parameters. This schedule and its minimal evaluation cost are determined using Algorithm 4.1 and routine **optimal** described in [16].

3.2 Heuristic

As explained in [16], the construction of an optimal nested reversal schedule $\mathbf{S}_{min}(l, C)$ by exhaustive search, as well as the computation of the minimal evaluation cost $\mathbf{T}_{min}(l, C)$, are extremely extensive in run-time and memory consumption. Therefore, it is required to construct an appropriate nested reversal schedule $\mathbf{S}_r(l, C)$ (r means here a recursive construction of reversal schedules), so that its evaluation cost $\mathbf{T}_r(l, C)$ differs not too much from the minimal evaluation cost $\mathbf{T}_{min}(l, C)$. Moreover, the construction of a nested reversal schedule $\mathbf{S}_r(l, C)$ has to be carried out using acceptable memory and run-time requirement. A suitable heuristic, which can be applied to the construction of a nested reversal schedule $\mathbf{S}_r(l, C)$, is developed in the present section.

Using the exhaustive search, one examines all possible distributions for thin and fat checkpoints and chooses from them the most efficient one. Obviously, such an exhaustive search is very expensive. The intent of the heuristic is to slightly restrict the placements of thin checkpoints. Due to the assumption (30), it is more convenient to a-priory prescribe the setting of thin checkpoints. This is because even a considerable increment in the number of evaluated primal steps does not cause a significant increase in the resulting evaluation cost w.r.t. the minimal evaluation cost $\mathbf{T}_{min}(l, C)$ (according to (30) and (38)). Since one fat checkpoint can be stored as soon as the required memory is available, we store thin checkpoints, so that after the removal of a sufficient number of thin checkpoints (c thin checkpoints), a corresponding fat checkpoint has to be stored in the same moment.

Then, an appropriate nested reversal schedule $\mathbf{S}_r(l, C)$ can be recursively constructed by minimizing the evaluation cost

$$(41)\ \mathbf{T}_r(l, C) \quad = \quad \min_{0<\hat{l}<l} \left\{ \mathbf{T}_{bin}(l - \hat{l} + 1, c)t + (l - \hat{l})\bar{t} + \mathbf{T}_r^1(l - \hat{l}, C) + \mathbf{T}_r(\hat{l}, , C - 1) \right\},$$

with

$$(42) \quad \mathbf{T}_r^1(l, C) \quad = \quad \min_{0<\hat{l}<l} \left\{ \mathbf{T}_{bin}(l - \hat{l}, c)t + (l - \hat{l})\bar{t} + \mathbf{T}_r^1(l - \hat{l}, C) + \mathbf{T}_r(\hat{l}, C - 1) \right\},$$

with $\mathbf{T}_{bin}(l, c)$ being a minimal evaluation cost of binomial reversal schedule (see 33).

The construction of a nested reversal schedule $\mathbf{S}_r(l, C)$, according to (41) - (42), leads to the following observation: a nested reversal schedule $\mathbf{S}_r(l, C)$ for the implementation of a multiple sweep evolution $\mathcal{E}_{3\times l}$, using up to C fat checkpoints, can

be decomposed into subschedules $\mathbf{S}_r(l-\hat{l},C)$ and $\mathbf{S}_r(\hat{l},C-1)$. $\mathbf{S}_r(l-\hat{l},C)$ and $\mathbf{S}_r(\hat{l},C-1)$ deal with the evolutions $\mathcal{E}_{3\times(l-\hat{l})}$ and $\mathcal{E}_{3\times\hat{l}}$, respectively. After storing the last adjoint state $\bar{x}_l$ into the extra fat checkpoint, the adjoint sweep to the adjoint state $\bar{x}_{\hat{l}}$ is performed. Data of the adjoint state $\bar{x}_{\hat{l}}$ is stored into the first fat checkpoint. Then, the sequence $\bar{F}_1,\ldots,\bar{F}_{\hat{l}}$ is 'reversed', i.e. the final steps $\bar{\bar{F}}_1,\ldots,\bar{\bar{F}}_{\hat{l}}$ are performed, using the remaining $(C-1)$ fat checkpoints. After that, the final step $\bar{\bar{F}}_{\hat{l}}$ causes the release of the fat checkpoint, where the adjoint state $\bar{x}_{\hat{l}}$ is stored. Finally, the reversal of the sequence $\bar{F}_{\hat{l}+1},\ldots,\bar{F}_l$ is performed, using again C fat checkpoints. We depict reversal schedules by quadrilaterals with two parallel horizontal lines as shown in Figure 6.

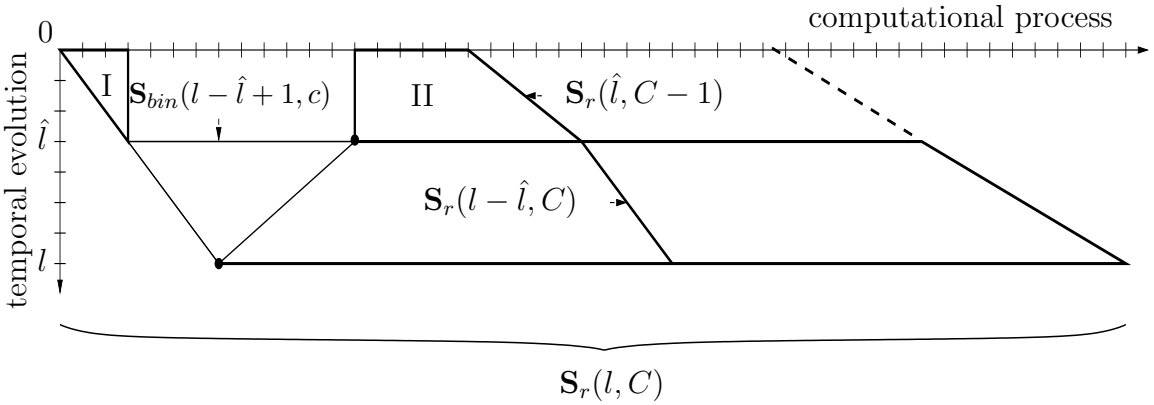

Figure 6: Decomposition of a nested reversal schedule $\mathbf{S}_r(l,C)$

Here, $\mathbf{S}_r(l,C)$ denotes a corresponding nested reversal schedule, which is constructed using the heuristic described above. This schedule is decomposed into two parts $\mathbf{S}_r(\hat{l},C-1)$ and $\mathbf{S}_r(l-\hat{l},C)$ by storing the first fat checkpoint. The adjoint state stored into the first fat checkpoint corresponds to $\hat{l}$. $\mathbf{S}_{bin}(l-\hat{l}+1,c)$ denotes the binomial reversal schedule with up to c checkpoints. This schedule is applied to the reversal of $(l-\hat{l}+1)$ primal steps, using minimal run-time and memory requirement.

Consider the slopes of the slanted lines in the subschedules $\mathbf{S}_r(\hat{l},C-1)$ and $\mathbf{S}_r(l-\hat{l},C)$. The top left corner of the schedule in Figure 6 is the initial point and the bottom right corner the final point. The bottom left corner represents the end of the (first) forward sweep and is connected to the top left corner by a slanted line of slope $1/t$, which equals one without loss of generality. The top right corner represents the second reversal, i.e. the transition from the adjoint sweep to the final sweep. The slope of the right edge represents the average speed of the final sweep, which is typically smaller than one as shown in Figure 6.

According to the decomposition in Figure 6, a distribution of thin checkpoints depends on a distribution of fat checkpoints. Thus, if the adjoint state $\bar{x}_{\hat{l}}$ is to be stored into the first fat checkpoint, then, during the primal sweep up to $c(C-1)$ thin checkpoints can be accommodated between the primal states $x_0,\ldots,x_{\hat{l}}$. The initial primal state x_0 is stored into the extra thin checkpoint. Moreover, up to c

thin checkpoints can be accommodated between the primal states $x_{\hat{l}}, \ldots, x_l$. The primal state $x_{\hat{l}}$ is stored into one of the c available thin checkpoints.

Therefore, during the primal sweep, the sequence of l primal steps is divided into $(C+1)$ parts. The situation is illustrated graphically in Figure 7. Here, up to c thin

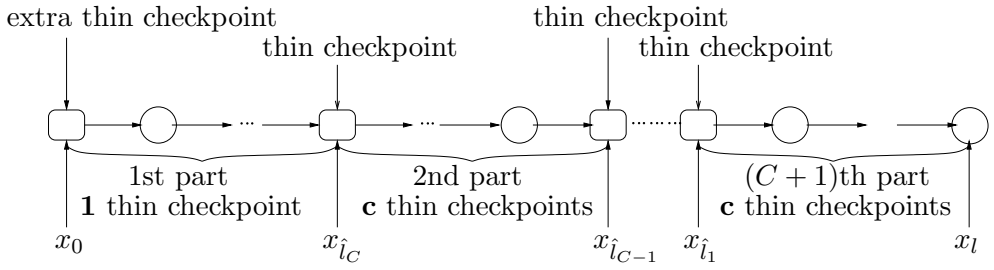

Figure 7: Partition of the primal sweep of a multiple sweep evolution $\mathcal{E}_{3\times l}$ corresponding to setting fat checkpoints

checkpoints are available for each of $(C+1)$ parts of a step sequence (except for the first one). The initial primal state of each part is stored into a thin checkpoint. The distribution of thin checkpoints within each part corresponds to the binomial distribution (see [6, 15]).

From (41) - (42) it is obvious that the temporal complexity $\mathbf{T}_r(l, C)$ and, consequently, a corresponding nested reversal schedule $\mathbf{S}_r(l, C)$ can be evaluated using dynamic programming. If we compute them recursively, namely, searching in the whole interval $(0, l)$ for the optimal value $\hat{l}_\star$ and storing values for $\mathbf{T}_r(m, s)$ and $\mathbf{T}_r^1(m, s)$ with $m < l$ and $s < C$ once they are evaluated, the run-time and memory requirement, needed for these computations, is quadratic with respect to the number l of time steps and the number C of fat checkpoints. This effort can be reduced to linear by exploiting the monotonicity property of the function $\mathbf{T}_r(l, C)$ described in the following. To this end a so-called **Monotonic Approach** is developed. Similar monotonicity properties were used by Knuth in [8] for the efficient construction of optimal binary trees for searching keys that occur with prespecified frequencies, as well as by Andrea Walther [19, 20] in context of determining of optimal reversals for non-uniform step costs.

3.3 Monotonic Approach

For the development of Monotonic Approach it is useful to introduce functions $\mathbf{T}_r^{\hat{l}}(l, C)$ and $\mathbf{T}_r^{1,\hat{l}}(l, C)$, defined by

$$\mathbf{T}_r^{\hat{l}}(l, C) = \mathbf{T}_{bin}(l - \hat{l} + 1, c + 1)t + (l - \hat{l})\bar{t} + \mathbf{T}_r^1(l - \hat{l}, C) + \mathbf{T}_r(\hat{l}, C - 1), \tag{43}$$

$$\mathbf{T}_r^{1,\hat{l}}(l, C) = \mathbf{T}_{bin}(l - \hat{l}, c + 1)t + (l - \hat{l})\bar{t} + \mathbf{T}_r^1(l - \hat{l}, C) + \mathbf{T}_r(\hat{l}, C - 1). \tag{44}$$

$\mathbf{T}_r^{\hat{l}}(l,C)$ and $\mathbf{T}_r^{1,\hat{l}}(l,C)$ are functions of $\hat{l}$ for l and C fixed. In the following the properties of the function $\mathbf{T}_r^{\hat{l}}(l,C)$ will be investigated. Similarly, one can obtain properties of $\mathbf{T}_r^{1,\hat{l}}(l,C)$ by applying the same arguments. The main property of the function $\mathbf{T}_r^{\hat{l}}(l,C)$, which provides the accuracy of Monotonic Approach, is the convexity of $\mathbf{T}_r^{\hat{l}}(l,C)$ with respect to its parameter $\hat{l}$. This is due to the construction of $\mathbf{T}_r^{\hat{l}}(l,C)$ as a sum of convex functions according to (43). Therefore, each local minimum of the function $\mathbf{T}_r^{\hat{l}}(l,C)$ is its global minimum. The convexity property of the function $\mathbf{T}_r^{\hat{l}}(l,C)$ is exploited for the development of Monotonic Approach. Due to the construction of $\mathbf{T}_r^{\hat{l}}(l,C)$ we obtain the following relation

$$\mathbf{T}_r(l,C) \;=\; \min_{0<\hat{l}<l} \mathbf{T}_r^{\hat{l}}(l,C). \tag{45}$$

Figure 8 illustrates three examples for $\mathbf{T}_r^{\hat{l}}(100,C)$ as a function of $\hat{l}$ for $l=100$, step cost distribution $\vec{\mathbf{t}}_{\mathbf{3\times 100}} = (1,1,1)^\top$, size distributions $\vec{\mathbf{d}}_{\mathbf{3\times 100}} = (1,3,1)^\top$, and different numbers C of fat checkpoints. In Figure 8 values of the function $\mathbf{T}_r^{\hat{l}}(100,C)$

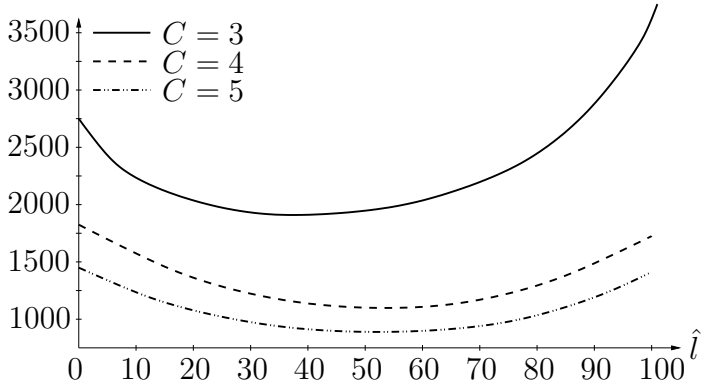

Figure 8: $\mathbf{T}_r^{\hat{l}}(100,C)$ as a function on $\hat{l}$

are plotted along the vertical axis, so that l and C are constant and $\hat{l}$ is varying. Values of $\hat{l}$ are plotted along the horizontal axis. Each line corresponds to different numbers $C=3,4,5$ of available fat checkpoints. In Figure 8 it can be also seen that the function $\mathbf{T}_r^{\hat{l}}(100,C)$ is convex.

For each function $\mathbf{T}_r^{\hat{l}}(l,C)$ there is at least one optimal value $\hat{l}_\star$, so that

$$\mathbf{T}_r(l,C) \;=\; \mathbf{T}_r^{\hat{l}_\star}(l,C) = \min_{0<\hat{l}<l} \mathbf{T}_r^{\hat{l}}(l,C). \tag{46}$$

Note, for $\hat{l}$ satisfying $\hat{l} \in (0,\hat{l}_\star)$ the function $\mathbf{T}_r^{\hat{l}}(l,C)$ decreases. For $\hat{l}$ satisfying $\hat{l} \in (\hat{l}_\star, l)$ the function $\mathbf{T}_r^{\hat{l}}(l,C)$ increases. To show this property consider the difference

$$\mathbf{T}_r^{\hat{l}}(l,C) - \mathbf{T}_r^{\hat{l}-1}(l,C)$$

and the statement of the following lemma proved in [16].

Lemma 3.1. Monotonic property for evaluation cost
Consider an evolution $\mathcal{E}_{3\times l}$ traversing l time steps in three alternative sweeps. $\vec{\mathbf{t}}_{\mathbf{3\times(l-1)}} = \vec{\mathbf{t}}_{\mathbf{3\times l}} = \vec{\mathbf{t}}_{\mathbf{3\times(l+1)}} = (t, \bar{t}, \bar{\bar{t}})^\top$ and $\vec{\mathbf{d}}_{\mathbf{3\times(l-1)}} = \vec{\mathbf{d}}_{\mathbf{3\times l}} = \vec{\mathbf{d}}_{\mathbf{3\times(l+1)}} = (d, \bar{d}, \bar{\bar{d}})^\top$ are corresponding step cost and size distributions. Let $C \in \mathbb{N}$ fat checkpoints be available each of which can accommodate one intermediate state vector of the size $\bar{d}$, i.e. one adjoint state $\bar{x}_i$, $0 \le i \le l$. Moreover, c thin checkpoints can be stored in place of a single fat one, i.e. $\bar{d} = c\,d$. Assume that the optimal value $\hat{m}_\star$ for $m \le l+1$ takes values $\hat{m}_\star \in [\hat{n}_\star, \hat{n}_\star + 1]$ with n defined by $n = m - 1$. Then, the following relations are satisfied

$$\mathbf{T}_r(l+1, C) - 2\mathbf{T}_r(l, C) + \mathbf{T}_r(l-1, C) \ge 0, \tag{47}$$

$$\mathbf{T}^1_r(l+1, C) - 2\mathbf{T}^1_r(l, C) + \mathbf{T}^1_r(l-1, C) \ge 0. \tag{48}$$

To establish a search algorithm for the construction of admissible nested reversal schedules $\mathbf{S}_r(l, C)$, the optimal point $\hat{l}_\star$ can be evaluated recursively. To this end the following assertion is proved.

Lemma 3.2. Optimal decomposition value
Consider an evolution $\mathcal{E}_{3\times l}$ traversing l time steps in three alternative sweeps. $\vec{\mathbf{t}}_{\mathbf{3\times l}} = (t, \bar{t}, \bar{\bar{t}})^\top$ and $\vec{\mathbf{d}}_{\mathbf{3\times l}} = (d, \bar{d}, \bar{\bar{d}})^\top$ are a corresponding step cost and size distribution. Let $C \in \mathbb{N}$ fat checkpoints be available each of which can accommodate one intermediate state vector of the size $\bar{d}$, i.e. one adjoint state $\bar{x}_i$, $0 \le i \le l$. Moreover, c thin checkpoints can be stored in place of a single fat one, i.e. $\bar{d} = c\,d$. Assume that $\hat{l}_\star$ and $\hat{l}^1_\star$ satisfying

$$\mathbf{T}_r(l, C) = \mathbf{T}^{\hat{l}_\star}_r(l, C) = \min_{0<\hat{l}<l} \mathbf{T}^{\hat{l}}_r(l, C), \qquad \mathbf{T}^1_r(l, C) = \mathbf{T}^{1,\hat{l}^1_\star}_r(l, C) = \min_{0<\hat{l}<l} \mathbf{T}^{1,\hat{l}}_r(l, C). \tag{49}$$

are determined. Let $\vec{\mathbf{t}}_{\mathbf{3\times m}}$ and $\vec{\mathbf{d}}_{\mathbf{3\times m}}$ $(m \ge l)$ be the corresponding step cost and size distributions, defined by $\vec{\mathbf{t}}_{\mathbf{3\times m}} = \vec{\mathbf{t}}_{\mathbf{3\times l}}$ and $\vec{\mathbf{d}}_{\mathbf{3\times m}} = \vec{\mathbf{d}}_{\mathbf{3\times l}}$, respectively. Then, for an evolution $\mathcal{E}_{3\times m}$ with $\vec{\mathbf{t}}_{\mathbf{3\times m}}$ and $\vec{\mathbf{d}}_{\mathbf{3\times m}}$ being its step cost and size distributions, and C available fat checkpoints, the optimal point $\hat{m}_\star$ takes a value from the interval $\left[\hat{l}_\star, \ldots, \hat{l}_\star + (m-l)\right]$, i.e. for the optimal point $\hat{m}_\star$ the following relation is satisfied

$$\hat{m}_\star \in \left[\hat{l}_\star, \ldots, \hat{l}_\star + (m-l)\right]. \tag{50}$$

Moreover, for the optimal point $\hat{m}^1_\star$ the following relation is satisfied

$$\hat{m}^1_\star \in \left[\hat{l}^1_\star, \ldots, \hat{l}^1_\star + (m-l)\right]. \tag{51}$$

Proof. It is sufficient to prove the assertion of Lemma 3.2 for $m = l + 1$. Then, the result can be extended for any arbitrary m. First, prove the property (50). This relation can be proved by induction over l for fixed C.

Trivial Case ($l \leq C$). For this trivial case the assertion (50) is obviously satisfied, since $\hat{m}_\star = l = \hat{l}_\star + 1$, where $m = l + 1$ and $\hat{l}_\star = l - 1$.

Induction step over l. The numbers C and l are given. Assume that the assertion (50) is true for all $(C, \tilde{l})$ with $\tilde{l} \leq l$. Now, it will be shown that the property (50) is valid for the pair $(C, l + 1)$.

Since $\hat{l}_\star$ satisfies (46), the following relations are also satisfied

$$\mathbf{T}_r^{\hat{l}_\star}(l, C) - \mathbf{T}_r^{\hat{l}_\star - 1}(l, C) \leq 0, \qquad (\hat{l}_\star > 1), \tag{52}$$

and

$$\mathbf{T}_r^{\hat{l}_\star + 1}(l, C) - \mathbf{T}_r^{\hat{l}_\star}(l, C) \geq 0, \qquad (\hat{l}_\star < l - 1). \tag{53}$$

If the optimal decomposition value $\hat{l}_\star$ is unique, then the inequalities (52) - (53) are strictly satisfied. In the case, if we have an interval of optimal values containing at least two numbers, the strict inequalities in (52) - (53) may not be satisfied. This situation is graphically depicted in Figure 9.

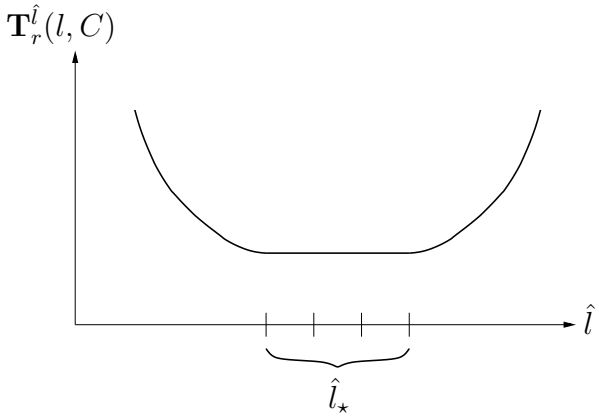

Figure 9: Example for the function $\mathbf{T}_r^{\hat{l}}(l, C)$ with an interval of optimal decomposition values

To show that $\hat{m}_\star \in [\hat{l}_\star, \hat{l}_\star + 1]$, it is sufficient to point out that

$$\mathbf{T}_r^{\hat{l}_\star}(l + 1, C) - \mathbf{T}_r^{\hat{l}_\star - 1}(l + 1, C) \leq 0, \ (\Leftrightarrow \hat{m}_\star \geq \hat{l}_\star > 1), \tag{54}$$

and

$$\mathbf{T}_r^{\hat{l}_\star + 2}(l + 1, C) - \mathbf{T}_r^{\hat{l}_\star + 1}(l + 1, C) \geq 0, \ (\Leftrightarrow \hat{m}_\star \leq \hat{l}_\star + 1 < l). \tag{55}$$

Firstly, we point out that (54) is fulfilled. For this purpose, we evaluate the difference between (52) and (54). After that, we prove that this difference is non-negative. Therefore, because of relation (52), the inequality (54) is satisfied. Thus,

$$\begin{aligned} &\mathbf{T}_r^{\hat{l}_\star}(l, C) - \mathbf{T}_r^{\hat{l}_\star - 1}(l, C) - \mathbf{T}_r^{\hat{l}_\star}(l + 1, C) + \mathbf{T}_r^{\hat{l}_\star - 1}(l + 1, C) = \\ &= \left\{ \mathbf{T}_{bin}(l - \hat{l}_\star + 1, c) - 2\mathbf{T}_{bin}(l - \hat{l}_\star + 2, c) + \mathbf{T}_{bin}(l - \hat{l}_\star + 3, c) \right\} t + \\ &+ \mathbf{T}_r^1(l - \hat{l}_\star + 2, C) - 2\mathbf{T}_r^1(l - \hat{l}_\star + 1, C) + \mathbf{T}_r^1(l - \hat{l}_\star, C) \geq 0. \end{aligned} \tag{56}$$

The inequality (56) is proved, using Lemma 3.1. This is because the assertion of Lemma 3.2 is satisfied for $l-\hat{l}_\star+2 \le l$ with $\hat{l}_\star > 1$ due to the induction assumption. Thus, the Lemma 3.1 can be applied. Moreover,

$$
\begin{aligned}
&\mathbf{T}_{bin}(l-\hat{l}_\star+1,c) - 2\mathbf{T}_{bin}(l-\hat{l}_\star+2,c) + \mathbf{T}_{bin}(l-\hat{l}_\star+3,c) = \\
&= \mathbf{T}_{bin}(l-\hat{l}_\star+3,c) - \mathbf{T}_{bin}(l-\hat{l}_\star+2,c) - \\
&-(\mathbf{T}_{bin}(l-\hat{l}_\star+2,c) - \mathbf{T}_{bin}(l-\hat{l}_\star+1,c)) = \\
&= r_{bin}(l-\hat{l}_\star+3,c) - r_{bin}(l-\hat{l}_\star+2,c) \ge 0.
\end{aligned} \tag{57}
$$

Here, in (57) we use the monotonicity property of binomial repetition numbers $r_{bin}(l,c)$ with respect to l, which is obviously follows from the definition of repetition numbers (see also (34)). Therefore, (56) and, respectively, (54) are satisfied.

Now, we point out that (55) is fulfilled. For this purpose, we evaluate the difference between (55) and (53). After that, we prove that this difference is non-negative. Therefore, because of relation (53), the inequality (55) is satisfied. Thus,

$$
\begin{aligned}
&\mathbf{T}_r^{\hat{l}_\star+2}(l+1,C) - \mathbf{T}_r^{\hat{l}_\star+1}(l+1,C) - \mathbf{T}_r^{\hat{l}_\star+1}(l,C) + \mathbf{T}_r^{\hat{l}_\star}(l,C) = \\
&= \mathbf{T}_r(\hat{l}_\star+2,C-1) - 2\mathbf{T}_r(\hat{l}_\star+1,C-1) + \mathbf{T}_r(\hat{l}_\star,C-1) \ge 0.
\end{aligned} \tag{58}
$$

The inequality (58) is proved, using Lemma 3.1. This is because the assertion of Lemma 3.2 is satisfied for $\hat{l}_\star+2 \le l$ with $\hat{l}_\star < l-1$ due to the induction assumption. Thus, the Lemma 3.1 can be applied. Therefore, for $m = l+1$ we obtain: $\hat{m}_\star = \hat{l}_\star+1$ or $\hat{m}_\star = \hat{l}_\star$. Thus, the property (50) is proved for $m = l+1$.
Any arbitrary m can be represented in the following way

$$
m = l + \underbrace{1 + \cdots + 1}_{(m-l)}.
$$

Applying the arguments above $(m-l)$ times, we obtain that

$$
\hat{m}_\star \in \left[\hat{l}_\star, \ldots, \hat{l}_\star + (m-l)\right].
$$

Thus, the assertion (50) is proved for any m.

Obviously, consideration above can be applied to the function $\mathbf{T}_r^1(l,C)$. They are to be straightforwardly transformed by replacing function $\mathbf{T}_r(l,C)$ by function $\mathbf{T}_r^1(l,C)$. Therefore, the property (51) is also proved. This concludes the proof of Lemma 3.2.

□

The result of Lemma 3.2 in the special case $m = l+1$ is the following one: Once we have determined an optimal decomposition value $\hat{l}_\star$ for l time steps, the optimal value $\hat{m}_\star$ for $l+1$ time steps is shifted at least by one with respect to $\hat{l}_\star$. Thus, to determine $\hat{m}_\star$ we have to consider merely two cases. This fact is also true for $\hat{m}_\star^1$. The results of Lemma 3.2 represent a basic idea for construction $\mathbf{S}_r(l,C)$ recursively.

To find minimal points $\hat{l}_\star$ and $\hat{l}^1_\star$ for the functions $\mathbf{T}_r(l,C)$ and $\mathbf{T}^1_r(l,C)$, consider the functions $\mathbf{F}_r(l,C)$ and $\mathbf{F}^1_r(l,C)$, defined by

$$\mathbf{F}_r(l,C) = \mathbf{T}_r(l,C) - \mathbf{T}_r(l-1,C), \qquad \mathbf{F}^1_r(l,C) = \mathbf{T}^1_r(l,C) - \mathbf{T}^1_r(l-1,C). \tag{59}$$

For a specified step cost distribution $\vec{\mathbf{t}}_{\mathbf{3}\times\mathbf{1}}$, size distribution $\vec{\mathbf{d}}_{\mathbf{3}\times\mathbf{1}}$, and a number C of available fat checkpoints, values of the functions $\mathbf{F}_r(l,C)$ and $\mathbf{F}^1_r(l,C)$ can be evaluated recursively. Simultaneously, the appropriate nested reversal schedule $\mathbf{S}_r(l,C)$ is constructed. The temporal complexity of this nested reversal schedule, i.e. its evaluation cost is equal to $\mathbf{T}_r(l,C)$. The detailed recursive procedure for evaluating $\mathbf{F}_r(l,C)$ and $\mathbf{F}^1_r(l,C)$ is presented in [16].

As a conclusion of Lemma 3.2 and procedure for evaluation $\mathbf{F}_r(l,C)$ and $\mathbf{F}^1_r(l,C)$ (see [16]), the Algorithm A.1 is established (detailed description of Algorithm A.1 is given in Appendix A). This algorithm constructs an admissible nested reversal schedule $\mathbf{S}_r(l,C)$. The memory and run-time requirement for the implementation of Algorithm A.1 and, consequently, for the construction of an appropriate nested reversal schedule $\mathbf{S}_r(l,C)$ is of order $\mathcal{O}(lC)$. For the implementation of Algorithm A.1 there is a software available **heuristic**, which is coded using the C++ programming language environment. Among various evaluations- and run-time tests, this software was utilized to establish the next example and the following run-time tests.

Figure 10 shows one example for a nested reversal schedule $\mathbf{S}_r(9,2)$ for an evolution $\mathcal{E}_{3\times 9}$ with the step cost distribution $\vec{\mathbf{t}}_{\mathbf{3}\times\mathbf{9}} = (1,5,1)^\top$ and the size distribution $\vec{\mathbf{d}}_{\mathbf{3}\times\mathbf{9}} = (1,1,1)^\top$. Nested reversal schedule $\mathbf{S}_r(9,2)$ starts with restoring the initial

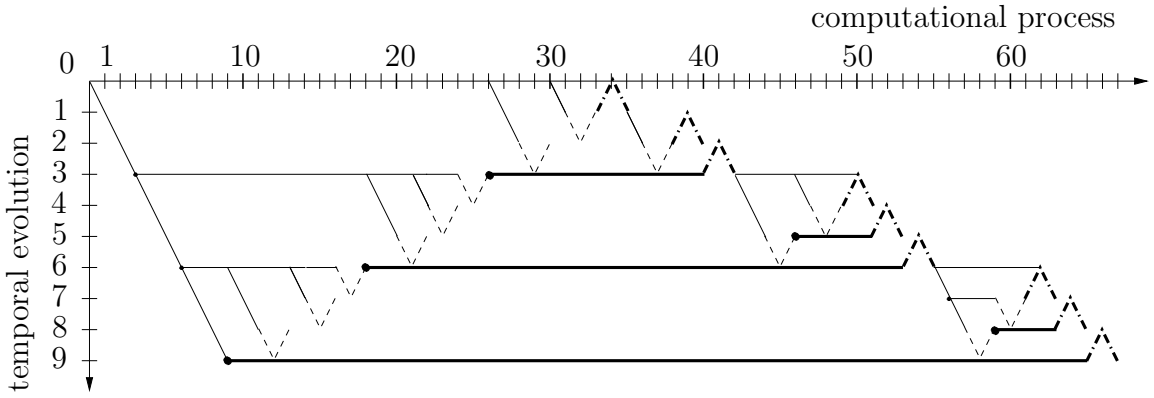

Figure 10: Nested reversal schedules $\mathbf{S}_r(9,2)$ constructed using the heuristic in Algorithm A.1

primal state x_0 from the zero-th thin checkpoint. Then, three primal steps F_1, F_2, and F_3 are performed, consecutively. The primal state x_3 is stored into the first thin checkpoint. Now, again, three primal steps F_4, F_5, and F_6 are performed, and the primal state x_6 is stored into the second thin checkpoint. Subsequently, the primal steps F_7, F_8, and F_9 are executed and the adjoint state $\bar{x}_9$ is initialized. The adjoint state $\bar{x}_9$ is stored into the extra fat checkpoint. The adjoint sweep is started by this action.

Subsequently, data of the primal state x_6 is restored from the second thin checkpoint, the primal steps F_7 and F_8 are performed, and the adjoint state $\bar{x}_8$ can be evaluated. Then, the primal state x_6 is anew restored from the second thin checkpoint, the primal state x_7 is reevaluated, the adjoint states $\bar{x}_7$ and $\bar{x}_6$ are evaluated. The adjoint state $\bar{x}_6$ is stored into the first fat checkpoint. Then, we go back to the adjoint state $\bar{x}_3$ by reevaluating required primal states and storing the adjoint state $\bar{x}_3$ into the second fat checkpoint. Subsequently, one goes back to the adjoint state $\bar{x}_1$ by reevaluating required intermediate states. Consequently, the first final step $\bar{\bar{F}}_1$ is performed. After that, we store the current initial primal state x_1 into the extra thin checkpoint and continue in the same manner in order to execute all other final steps $\bar{\bar{F}}_2, \ldots, \bar{\bar{F}}_9$.

Using the nested reversal schedule $\mathbf{S}_r(9,2)$ in Figure 10, it is necessary to perform 24 primal steps F_i, 13 adjoint steps $\bar{F}_i$, and nine final steps $\bar{\bar{F}}_i$. The nested reversal schedule $\mathbf{S}_r(9,2)$ and its evaluation cost $\mathbf{T}_r(9,2)$ are determined using Algorithm A.1 and routine **heuristic**.

If we compare Figure 4 and Figure 10, we see that nested reversal schedules shown in these figures are different, although the parameters $\vec{\mathbf{d}}_{\mathbf{3\times 9}}$, $\vec{\mathbf{t}}_{\mathbf{3\times 9}}$, and $C = 2$ are the same for the both schedules. The nested reversal schedule $\mathbf{S}_r(9,2)$ in Figure 10 can be represented as a decomposition of nested and binomial reversal schedules, according to Figure 6. Obviously, the optimal nested reversal schedule $\mathbf{S}_{min}(9,2)$ in Figure 4 does not possess this property. Moreover, nested reversal schedules $\mathbf{S}_{min}(9,2)$ and $\mathbf{S}_r(9,2)$ in Figure 4 and in Figure 10, respectively, differ from each other in the number of primal states, required for their execution. For performing $\mathbf{S}_{min}(9,2)$ 23 primal steps are needed. For performing $\mathbf{S}_r(9,2)$ this number increases up to one.

The resulting schedule $\mathbf{S}_r(l,C)$, constructed by Monotonic Approach, minimizes the temporal complexity in the context that the evaluation costs $\mathbf{T}_r(l,C)$ introduced by (41) are exactly attained. However, the resulting schedule $\mathbf{S}_r(l,C)$ in general is not an optimal one in that the strict inequality $\mathbf{T}_r(l,C) > \mathbf{T}_{min}(l,C)$ can be valid in some cases. Figure 11 shows deviations between the corresponding evaluation cost $\mathbf{T}_r(l,C)$ and the minimal evaluation cost $\mathbf{T}_{min}(l,C)$ for different values of l, C, and c, using the step cost distribution $\vec{\mathbf{t}}_{\mathbf{3\times 1}} = (1,5,1)^\top$ and different size distributions $\vec{\mathbf{d}}_{\mathbf{3\times 1}} = (1,c,1)^\top$, which depend on the parameter c. The horizontal axis denotes the number l of time steps. Deviations, evaluated by the formula

$$\frac{(\mathbf{T}_r(l,C) - \mathbf{T}_{min}(l,C))}{\mathbf{T}_{min}(l,C)} \times 100, \tag{60}$$

are plotted along the vertical axis. As we can see in Figure 11, for a combination $C = 2, c = 2$ deviations reach up to 3% w.r.t. the minimal evaluation cost $\mathbf{T}_{min}(l,C)$. In computational experience larger values of c and of ratio $\bar{t}/t$ lead to schedules that are closer to optimal. Therefore, in practice we can expect much smaller deviations. This is because in practice we have $c \gg 2$ and $\bar{t}/t \gg 1$. Due to the huge temporal complexity, needed to evaluate optimal nested reversal schedules $\mathbf{S}_{min}(l,C)$ and,

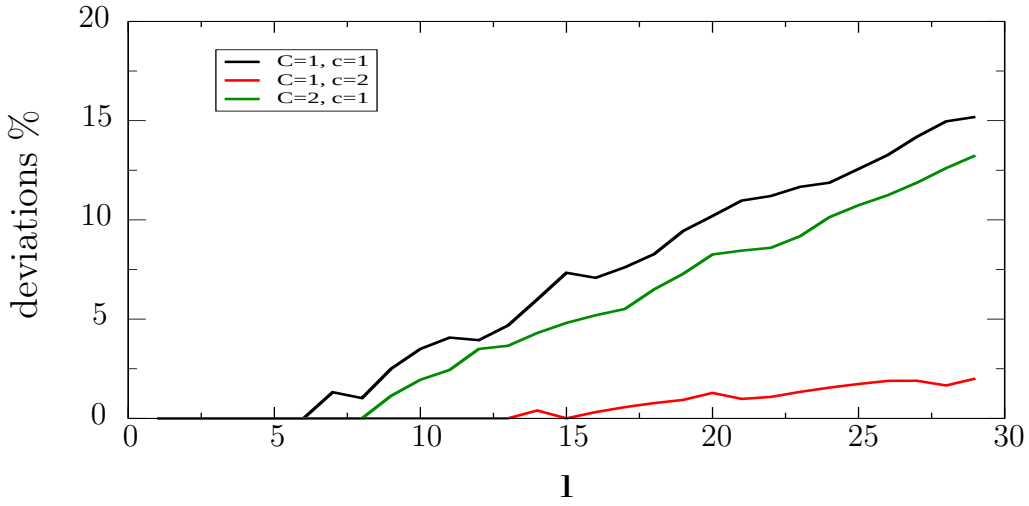

Figure 11: Deviations of $\mathbf{T}_r(l, C)$ from $\mathbf{T}_{min}(l, C)$

respectively, to compute minimal evaluation cost $\mathbf{T}_{min}(l, C)$, it is not possible to compute $\mathbf{T}_{min}(l, C)$ for more realistic combinations between C and c, than it is used in Figure 11. Nevertheless, the quality of the heuristic, presented in this section, can be investigated. For this end, use a lower bound for the minimal evaluation cost $\mathbf{T}_{min}(l, C)$, established in [16].

Figure 12 shows deviations between the corresponding evaluation cost $\mathbf{T}_r(l, C)$ and the lower bound $\mathbf{L}_{min}(l, C)$ for the minimal evaluation cost $\mathbf{T}_{min}(l, C)$ for different values of l, C, and c, using the previous parameters $\vec{\mathbf{t}}_{\mathbf{3\times 1}} = (1, 5, 1)^\top$ and $\vec{\mathbf{d}}_{\mathbf{3\times 1}} = (1, c, 1)^\top$. The horizontal axis denotes the number l of time steps. Deviations, evaluated by the formula

$$\frac{(\mathbf{T}_r(l, C) - \mathbf{L}_{min}(l, C))}{\mathbf{L}_{min}(l, C)} \times 100, \tag{61}$$

are plotted along the vertical axis. As we can see in Figure 12, deviations reach up to 20% w.r.t. the lower bound $\mathbf{L}_{min}(l, C)$. Consequently, deviations between the evaluation cost $\mathbf{T}_r(l, C)$ and the minimal evaluation cost $\mathbf{T}_{min}(l, C)$ are even smaller. Two last examples with $c = 100$ show situations close to realistic. Here, deviations only reach up to 3% w.r.t. the lower bound $\mathbf{L}_{min}(l, C)$.

Thus, nested reversal schedules as well as their evaluation cost, computed by the heuristic and Monotonic Approach give a very good approximation to optimal reversal schedules and their minimal evaluation cost. Moreover, these schedules can be constructed using an acceptable run-time and memory amount.

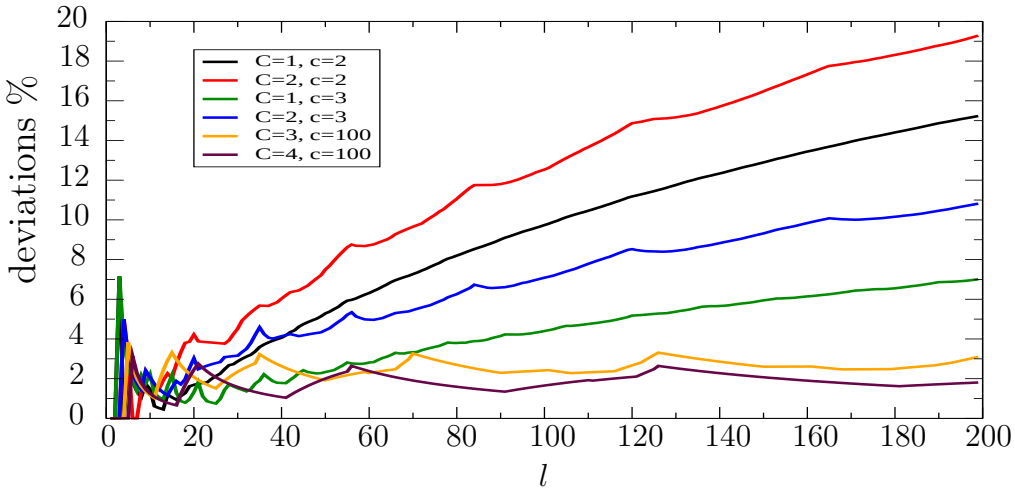

Figure 12: Deviations of $\mathbf{T}_r(l, C)$ from $\mathbf{L}_{min}(l, C)$

4 Numerical Application

We consider a control problem that describes the laser surface hardening of steel (see also [7]). How this process operates is depicted in Fig. 13. A laser beam

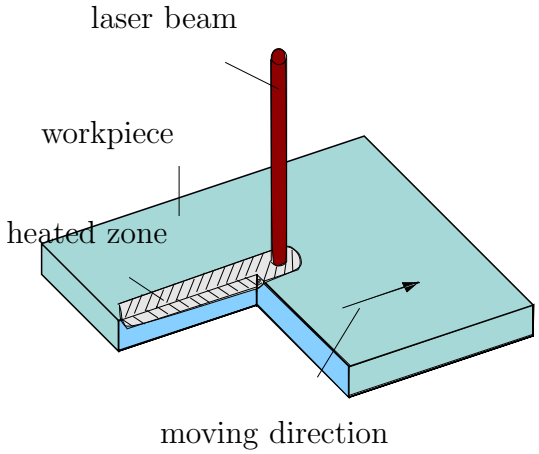

Figure 13: Sketch of a laser hardening process

moves along the surface of a workpiece, creating a heated zone around its trace. At the same time, the high temperature phase, the so called austenite, is produced in steel. Due to physical constraints, the moving velocity of the laser beam has to be kept constant during the whole heating process. Thus, the laser energy is chosen as a control parameter. Whenever the temperature in the heated zone exceeds the melting temperature of steel, the work-piece quality can be destroyed. Therefore, the goal of surface hardening is to achieve a desired hardening zone, which is described

by a desired phase distribution a_d of austenite inside the workpiece Ω, but to avoid a melting of the surface. Hence, we consider an optimal control problem with the cost functional $J(u)$ defined as

$$J(u) = \frac{\beta_1}{2}\int_\Omega (a(x,T) - a_d(x))^2 dx + \frac{\beta_2}{2}\int_0^T \int_\Omega [\theta - \theta_m]_+^2 dx\, dt + \frac{\beta_3}{2}\int_0^T u^2 dt, \tag{62}$$

where u is the laser energy and β_i $i = 1, 2, 3$ are positive constants. The second term in (62) penalizes temperatures above the melting temperature θ_m.

Let $\Omega := [0, 5] \times [-1, 0]$ with Lipschitz boundary $Q = \Omega \times (0, T)$, $\Sigma = \partial\Omega \times (0, T)$, $T = 5.25$. The system of state equations (63) - (67) consists of a semi-linear heat equation coupled with the initial-value problem for the phase transitions. a is the volume fraction of austenite, θ the temperature, τ a time constant and $[x]_+ = \max\{x, 0\}$ the positive part function. The equilibrium volume fraction a_{eq} is such that the austenite volume fraction increases during heating until it reaches some value $a < 1$. During cooling we have $a_t = 0$, and the value a is kept. The homogeneous Neumann conditions are assumed on the boundary. The term $-\rho L a_t$ describes the consumption of latent heat due to the phase transition. The term $u(t)\alpha(x, t)$ is the volumetric heat source due to laser radiation, where the laser energy $u(t)$ will serve as a control parameter. The density ρ, the heat capacity c_p, the heat conductivity k, and the latent heat L are assumed to be positive constants.

$$a_t = \frac{1}{\tau(\theta)}[a_{eq}(\theta) - a]_+, \qquad \text{in } Q, \tag{63}$$

$$a(0) = 0, \qquad \text{in } \Omega, \tag{64}$$

$$\rho c_\rho \theta_t - k\triangle\theta = -\rho L a_t + u\alpha, \qquad \text{in } Q, \tag{65}$$

$$\frac{\partial\theta}{\partial\nu} = 0, \qquad \text{on } \Sigma, \tag{66}$$

$$\theta(0) = \theta_0, \qquad \text{in } \Omega. \tag{67}$$

We study the following optimal control problem for the cost functional $J(u)$ as defined in (62):

$$\min J(u), \text{ s.t. } (\theta, a, u) \text{ solves } (63) - (67). \tag{68}$$

The numerical implementation is obtained by an implicit FE Galerkin scheme. For the numerical implementation the positive part function $[x]_+$ is approximated by the function $x\mathcal{H}(x)$, where $\mathcal{H}$ is a regularized Heaviside function (for details see [16]). The FE triangulation of Ω is done by a nonuniform mesh. For the solution of large sparse systems of linear equations resulting through the FE discretization a Fortran-package MA47.f from the Harwell library is applied. The optimal control problem (68) is solved using the Pantoja algorithm 2.1. Nested reversal schedules are utilized for the reduction of memory requirement during the implementation of the Algorithm 2.1 applied to the optimal control problem (68). Backtracking line-search procedure is applied to choose a suitable step length for each Newton step.

As soon as the spatial discretization is performed and the mass matrix is established, this matrix will not be changed during the overall optimization run. Thus, it can be factorized once and for all, so that computation θ as well as its derivatives at each time step requires only one extra forward and backward substitution. As mentioned before, these actions are performed using appropriate procedures from MA47.f. Moreover, for matrix operations the sparsity of the mass matrix is exploited. As a result, the speed of the optimization run is raised.

In the following one example for a solution of laser surface hardening of steel using Pantoja algorithm is described. The desired volume fraction of austenite is shown in Figure 14. The spatial discretization scheme has 861 degrees of freedom. We choose $T = 5.25$ and want to achieve a constant hardening depth of 1mm. The time interval $(0, T)$ is discretized into 100 time steps. As the first iterate for the control we take $\mathbf{u}^0 = 1000$. Algorithm 2.1 needs 44 iterations and stops with $\Lambda(44) < 10^{-6}$ where $\Lambda(i) = \frac{\|\delta\mathbf{u}^i\|}{\|\delta\mathbf{u}^i + \mathbf{u}^i\|}$.

In Figure 15 the differences $a^0(T) - a_d$ and $a^{44}(T) - a_d$ are shown. Using the optimal control, we obtain a significant reduction of the residuum. Figure 16 shows the initial a^0 and the terminal a^{44} volume fraction of austenite at the time $t = T$. A comparison of plots in Figure 16 shows that the goal of a uniform hardening depth has been nearly achieved. The resulting optimal control $\mathbf{u}^{44}$ is depicted in Figure 17. As expected, the laser energy first has to be increased, then, during a long time it can be kept constant. Towards the end of the process, it has to be reduced to cope with the accumulation of heat at the end of the plate. The small oscillation in the resulting control can be explained by the high non-linearity in the state equation. The authors believe that these oscillations can be reduced or even omitted by the suitable spatial discretization with e.g. adaptive FE grids. This is the subject of the ongoing work.

Figure 18 gives two plots for the initial gradient ∇J^0 and the final gradient ∇J^{44} of the cost function as a function on time t. On these graphs we can see how the gradient ∇J^0 is reduced after 44 iterations.

Table 1 presents data obtained during the run of the program. The first column of this table gives the iteration step number i, the second one the step size λ^i obtained after the backtracking line search procedure for the corresponding iteration step. The value of the cost function $J(\mathbf{u}^i)$, euclidean norm for the gradient $\nabla J(\mathbf{u}^i)$ of the cost function, and the product $\nabla J(\mathbf{u}^i)\delta\mathbf{u}^i$, which identify whether the direction $\delta\mathbf{u}^i$ is a descent direction or not, are given in the third, fourth, and the fifths columns of Table 1. If the obtained direction $\delta\mathbf{u}^i$ is not a descent direction, i.e. if the product $\nabla J(\mathbf{u}^i)\delta\mathbf{u}^i$ is positive, then, we use the descent direction $\delta\mathbf{u}^i = -\nabla J(\mathbf{u}^i)$ in this iteration. The norm $\|a^{44}(T) - a_d\|_{L^2(\Omega)}$ for the difference $(a^i(T) - a_d)$ between the obtained state a^i and the desired state a_d, the norm $\|\theta^i\|_{L^\infty(Q)}$ of the temperature θ^i, and the relative norm $\Lambda(i)$ are given in the sixth, the seventh, and the eighth columns of Table 1. We observe that $\|\theta^{44}\|_{L^\infty(Q)} = 1097.968289$, $\|a^0(T) - a_d\|_{L^2(\Omega)} = 0.213576$, and $\|a^{44}(T) - a_d\|_{L^2(\Omega)} = 0.120977$. The norm $\|\nabla J(\mathbf{u}^i)\|_2$ of the gradient is reduced

from 0.600319 to 0.0. Moreover, the cost functional $J(\mathbf{u}^0) = 3081.146543$ is reduced to $J(\mathbf{u}^{44}) = 2580.659615$.

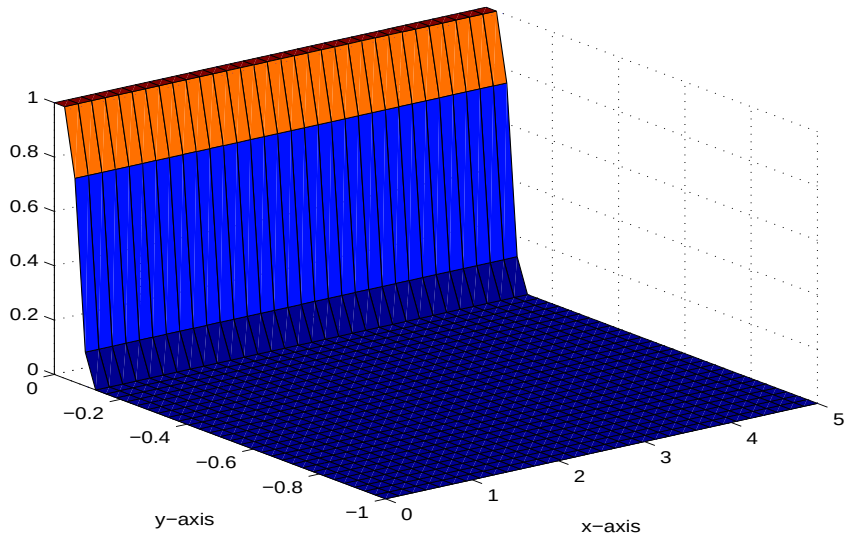

Figure 14: Desired state a_d

For the memory efficient implementation of model problem for laser surface hardening of steel nested reversal schedules are used. Therefore, the memory requirement is reduced. Naturally, the run-time requirement can increase. Figure 19 illustrates resulting run-time ratios. x-axis gives the number of time-steps. y-axis shows ratios between run-time needed by nested reversal schedule and run-time needed by basic approach, so that all intermediate states are stored. Each line of this figure corresponds to a certain number of checkpoints available.

The blue line shows run-time ratios if only three checkpoints are available. The green line corresponds to the case if five checkpoints can be stored. As we can see, ratios are reduced compare to the previous case.

The third case (red line) shows run-time ratios if the number of checkpoints available is evaluated by the natural logarithm of time-steps. Here we can see characteristic jumps in run-times if number of checkpoints is increased by one. From Figure 19 we can observe that run-times, resulting from nested reversal schedules, increase in up to six times compared to the basic approach. At the same time, the memory requirement is reduced in 100 times compared to the basic approach. Thus, using nested reversal schedules, memory requirement can be reduced enormously, while run-time requirement increases slightly.

Figure 20 illustrates two types of dependencies at the same time: Firstly, the dependence of run-time on the storage amount; secondly, the dependence of value of the objective function at the optimum on the storage amount. Run-time plotted

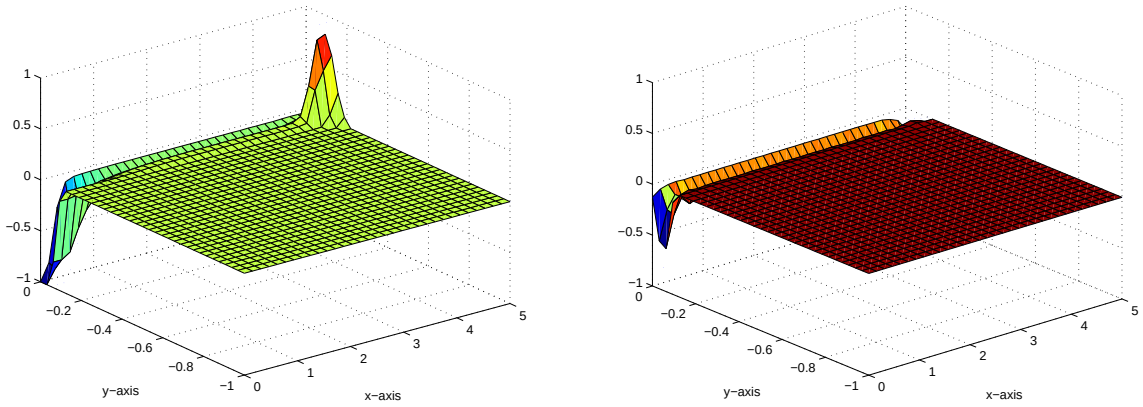

Figure 15: Differences $a^0(T) - a_d$ and $a^{44}(T) - a_d$ for the first iterate a^0 and for the optimal state a^{44}

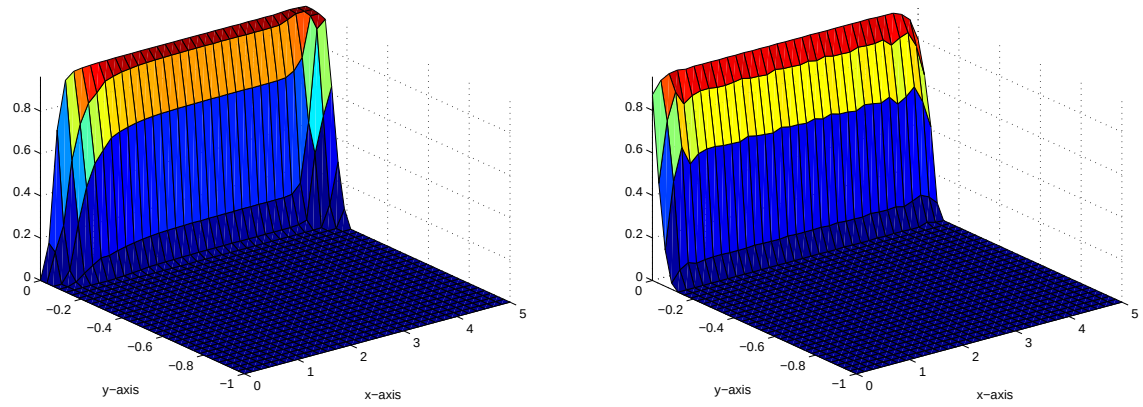

Figure 16: FE snapshot for the austenite at time $t = T$ for the first iterate a^0 and at time $t = T$ for the optimal state a^{44}

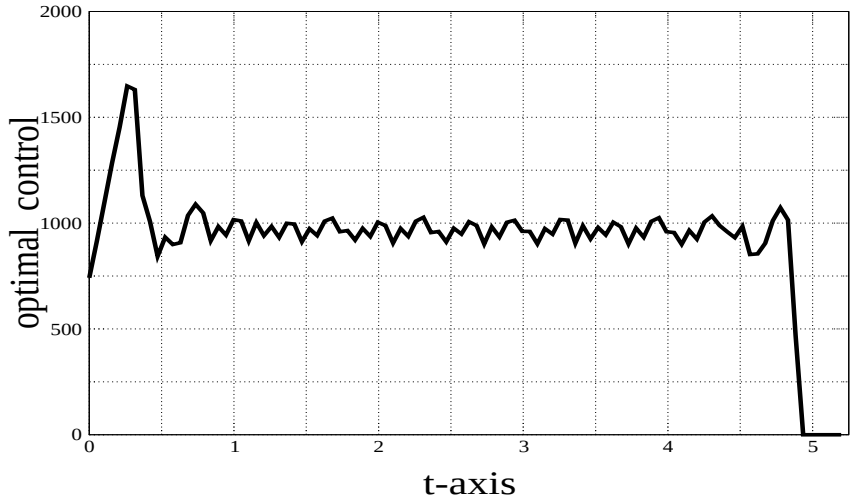

Figure 17: Optimal control $\mathbf{u}^{44}$

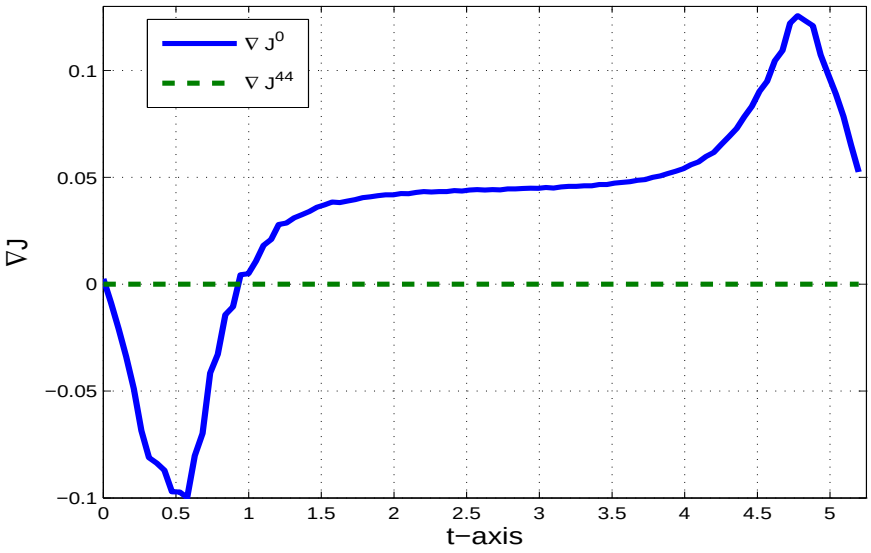

Figure 18: Gradients ∇J^0 and ∇J^{44} of the cost function

Table 1: Program data for laser surface hardening of steel example

i	λ^i	$J(\mathbf{u}^i)$	$\|\nabla J(\mathbf{u}^i)\|_2$	$\nabla J(\mathbf{u}^i)\delta\mathbf{u}^i$	$\|a^i(T)-a_d\|_{L^2(\Omega)}$	$\|\theta^i\|_{L^\infty(Q)}$	$\Lambda(i)$
0	0.000000	3081.146543	0.600319	−0.000000	0.213576	1170.914882	0.000000
1	0.100000	2973.616139	0.565653	−3149.260435	0.204818	1200.814136	0.277261
2	0.000195	2973.616139	0.565355	197.271856	0.204781	1200.707729	0.000057
3	0.000195	2973.616139	0.565058	44.634094	0.204744	1200.601287	0.000057
4	0.100000	2969.875370	0.536605	−26.581734	0.211618	1213.097537	0.042351
5	0.103570	2968.081061	0.521728	−2292.567972	0.209614	1202.763321	0.369994
6	0.204671	2947.416885	0.317659	−3545.005088	0.187332	1233.498086	0.631641
7	0.100000	2869.742635	0.257898	−845.682962	0.168136	1198.463786	0.074350
8	1.000000	2742.810703	0.181524	−478.226890	0.186381	1056.788317	0.355983
9	0.100000	2731.538299	0.160197	−117.440575	0.185802	1062.765592	0.038998
10	1.000000	2695.679423	0.075834	−132.601985	0.181731	1196.550516	0.199851
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
40	0.130447	2580.711193	0.005492	−1.312499	0.121144	1095.266703	0.006241
41	1.000000	2580.660008	0.000240	−0.106144	0.120987	1098.180298	0.004811
42	1.000000	2580.659615	0.000010	−0.000750	0.120979	1097.971548	0.000425
43	1.000000	2580.659615	0.000000	−0.000001	0.120977	1097.968294	0.000018
44	1.000000	2580.659615	0.000000	−0.000000	0.120977	1097.968289	0.000000

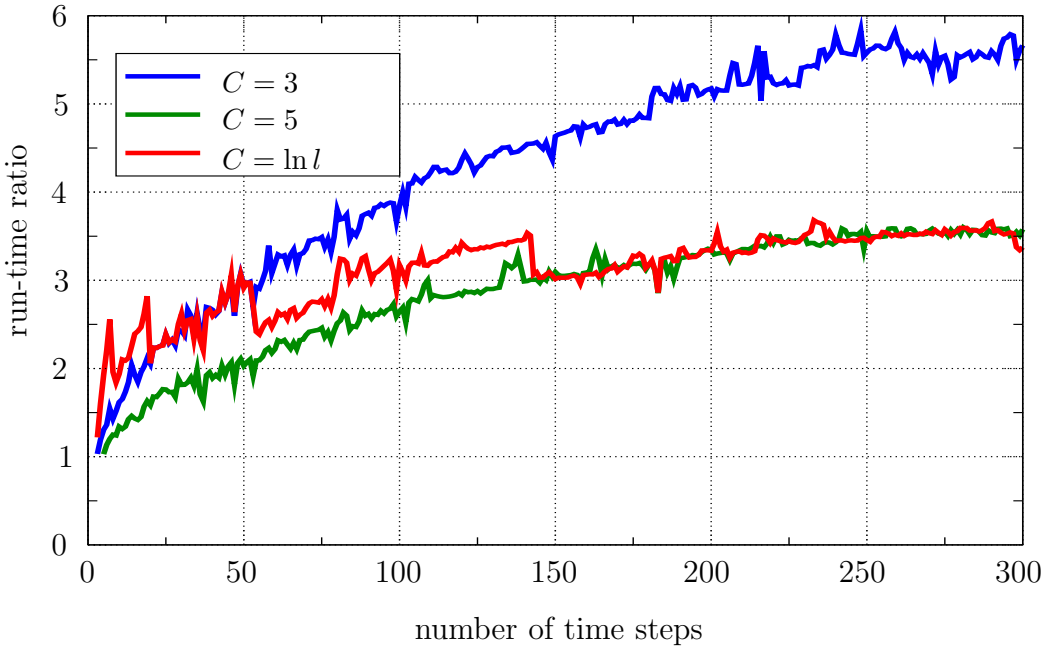

Figure 19: Run-time & Memory Trade-off

along the left vertical axis is counted in machine internal units. Storage amount is given by the number of fat checkpoints used by nested reversal schedules, and is plotted along the horizontal line. The value of the objective function $J_\star$ at the optimal point $\mathbf{u}_\star$ is plotted along the right vertical axis.

The blue solid line in Figure 20 represents the resulting run-time requirement, needed for the Pantoja algorithm applied to the optimal control problem of laser surface hardening of steel, as a function of the memory requirement. We observe a dramatic decrease in the run-time complexity as a number of available checkpoints increases from three to 15. After that, for the number of checkpoints grater than 15, the run-time requirement decreases just slightly. This means, that the reduction of memory requirement from 100 to 15 checkpoints does not imply a significant increase in the run-time requirement.

The red dashed line in Figure 20 represents the dependence of the value $J_\star$ on the memory requirement. The objective function optimal value $J_\star$ obviously stays constant for all numbers of checkpoints. This means, the optimal control problem is always solved exactly irrespective how many checkpoints are available. Consequently, one can find a suitable number of checkpoints, so that the memory requirement is acceptable, and solve the optimal control problem exactly. It can be arranged by choosing a sufficient number of checkpoints, which is significantly less than the number of time steps, that the resulting run-time complexity increases just

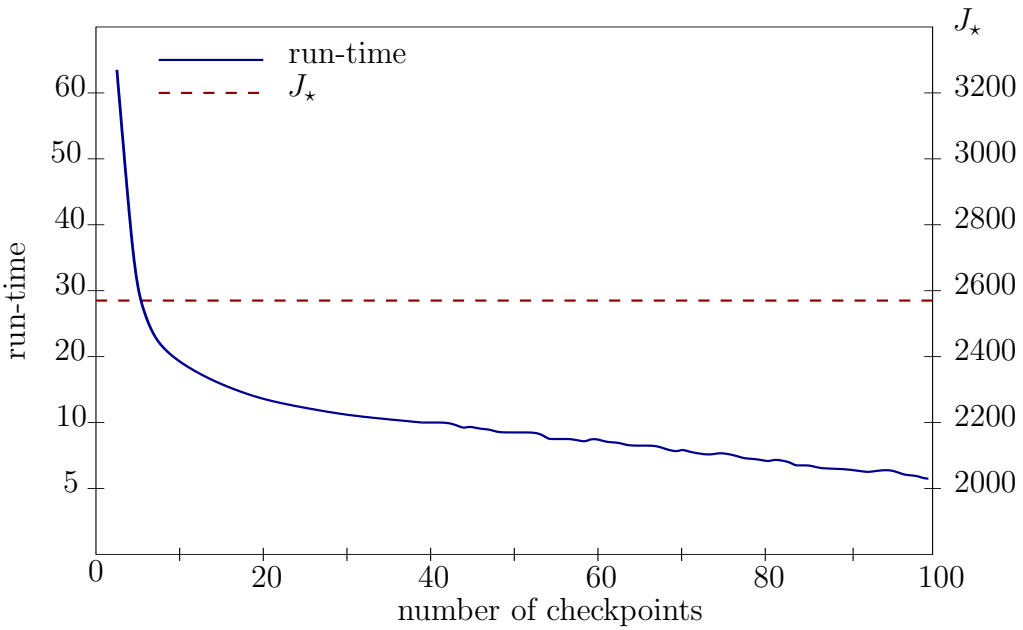

Figure 20: Run-time requirement and the optimal value of the objective function in dependence of the memory amount

slightly compared to the case, where all intermediate states have to be stored at each time interval (number of checkpoints = number of time steps). For instance, it was only possible to apply Pantoja approach to the optimal control problem of laser surface hardening of steel with the spatial discretization as above, using nested reversal schedules. Thus, the checkpointing techniques are inevitable for this example.

5 Conclusion and Outlook

The iterative solution of optimal control problems in ODEs by various methods leads to a succession of triple sweeps through the discretized time interval. The second, or adjoint sweep relies on information from the first, or primal sweep, and the third, or final sweep depends on both of them. This flow of information is depicted in Figure 1. Typically, the steps on the adjoint sweep involve more operations and require more storage than the other two. In order to avoid storing full traces of the primal and adjoint sweeps we consider nested reversal schedules that require only the storage of selected primal and adjoint intermediate states called thin and fat checkpoints. The schedules are designed to minimize the overall execution time given a certain total amount of storage for the checkpoints. The heuristic established in the present paper gives an opportunity to construct an appropriate nested reversal schedule for given parameters using an acceptable amount of time. To realize this task a Monotonic approach is introduced. It can be demonstrated that the dependence on l can be

arranged polylogarithmically [3] by nested checkpoint strategies. Consequently, the operations count also grows as a second power of $\ln l$, which need however not result in an increase of the actual run-time due to memory effect (for details see [16, 17]).

The proposed scheduling schemes have been applied to an optimal control problem describing a laser surface hardening of steel. It was observed throughout an enormous reduction of memory requirement, while run-time requirement increases just slightly. Moreover, the numerical example of laser surface hardening of steel confirms the analytic estimation for the growth of the run-time complexity.

A Algorithm for Construction of Nested Reversal Schedules

An algorithm for construction of a nested reversal schedule $\mathbf{S}_r(l, C)$ as described in Section 3.3 is listed below.

Algorithm A.1. Construction of $\mathbf{S}_r(l, C)$

Input: $\mathcal{E}_{3\times l}$ - multiple sweep evolution
$\vec{\mathbf{t}}_{\mathbf{3\times 1}} = (t, \bar{t}, \bar{\bar{t}})^\top$ - step cost distribution
$\vec{\mathbf{d}}_{\mathbf{3\times 1}} = (d, \bar{d}, \bar{\bar{d}})^\top$ - size distribution
C - number of available fat checkpoints

Output: values of $\hat{m}_\star$, $\hat{m}^1_\star$, $\mathbf{T}_r(m,n)$, $\mathbf{T}^1_r(m,n)$, $\mathbf{F}_r(m,n)$ and $\mathbf{F}^1_r(m,n)$ for $0 < m \le l$ and $0 \le n \le C$.

For $(m = 0;\ m < l;\ m++)$
  For $(n = 0;\ n \le C;\ n++)$
    If $(m + 1 \le n + 1)$
      $\hat{g}_\star = m$ for $g = m + 1$
      $\hat{g}^1_\star = m$ for $g = m + 1$
      $\mathbf{T}_r(m+1, n) = (m+1)t + m\bar{t} + (m+1)\bar{\bar{t}}$
      $\mathbf{F}_r(m+1, n) = t + \bar{t} + \bar{\bar{t}}$
      $\mathbf{T}^1_r(m+1, n) = mt + m\bar{t} + (m+1)\bar{\bar{t}}$
      $\mathbf{F}^1_r(m+1, n) = t + \bar{t} + \bar{\bar{t}}$
    Else
      If $(n == 0)$
        $$\mathbf{T}_r(m+1, 0) = (m+1)t + \sum_{i=1}^{m+1} \{\mathbf{T}_{bin}(i, 1)t + (i-1)\bar{t}\} + (m+1)\bar{\bar{t}}$$
        $$\mathbf{F}_r(m+1, 0) = \tfrac{m(m+1)+2}{2}t + m\bar{t} + \bar{\bar{t}}$$
        $$\mathbf{T}^1_r(m+1, 0) = \sum_{i=1}^{m+1} \{\mathbf{T}_{bin}(i, 1)t + (i-1)\bar{t}\} + (m+1)\bar{\bar{t}}$$
        $$\mathbf{F}^1_r(m+1, 0) = \tfrac{m(m+1)}{2}t + m\bar{t} + \bar{\bar{t}}$$
      Else

If $\left(\mathbf{F}_r(\hat{m}_\star + 1, n-1) < r(m - \hat{m}_\star + 2, c)t + \mathbf{F}_r^1(m - \hat{m}_\star + 1, n) + \bar{t}\right)$
$\hat{g}_\star = \hat{m}_\star + 1$ for $g = m + 1$
$\mathbf{F}_r(m+1, n) = \mathbf{F}_r(\hat{m}_\star + 1, n-1)$
Else
$\hat{g}_\star = \hat{m}_\star$ for $g = m + 1$
$\mathbf{F}_r(m+1, n) = r(m - \hat{m}_\star + 2, c)t + \mathbf{F}_r^1(m - \hat{m}_\star + 1, n) + \bar{t}$
If $\left(\mathbf{F}_r(\hat{m}_\star^1 + 1, n-1) < r(m - \hat{m}_\star^1 + 1, c)t + \mathbf{F}_r^1(m - \hat{m}_\star^1 + 1, n) + \bar{t}\right)$
$\hat{g}_\star^1 = \hat{m}_\star^1 + 1$ for $g = m + 1$
$\mathbf{F}_r^1(m+1, n) = \mathbf{F}_r(\hat{m}_\star^1 + 1, n-1)$
Else
$\hat{g}_\star^1 = \hat{m}_\star^1$ for $g = m + 1$
$\mathbf{F}_r^1(m+1, n) = r(m - \hat{m}_\star^1 + 1, c)t + \mathbf{F}_r^1(m - \hat{m}_\star^1 + 1, n) + \bar{t}$
$\mathbf{T}_r(m+1, n) = \mathbf{T}_r(m, n) + \mathbf{F}_r(m+1, n)$
$\mathbf{T}_r^1(m+1, n) = \mathbf{T}_r^1(m, n) + \mathbf{F}_r^1(m+1, n)$

References

[1] B. Christianson. Cheap Newton steps for optimal control problems: Automatic differentiation and pantoja algorithm. *Optimization Methods and Software*, 10:729–743, 1999.

[2] B. Christianson and M. Bartholomew-Biggs. Globalization of pantoja's optimal control algorithm. In G.F. Corliss, C. Faure, A. Griewank, L. Hascoët, and U. Naumann, editors, *Automatic Differentiation: from Simulation to Optimization*, pages 125–130, New York, 2001. Springer.

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

[4] A. Griewank. *Evaluating derivatives. Principles and techniques of algorithmic differentiation.* SIAM, Philadelphia, 2000.

[5] A. Griewank and A. Walther. Treeverse: An implementation of checkpointing for the reverse or adjoint mode of computational differentiation. *Tech. Report IOKOMO-04-1997, Techn. Univ. Dresden*, 1997.

[6] M. Hinze, A. Walther, and J. Sternberg. Discrete approximation schemes for reduced gradients and reduced hessians in Navier-Stokes control utilizing an optimal memory-reduced procedure for calculating adjoints. *Optimal Control Applications and Methods*, 27:19–40, 2006.

[7] D. Hömberg and S. Volkwein. Control of laser surface hardening by a reduced-order approach using proper orthogonal decomposition. *Mathematical and Computer Modelling*, 38:1003–1028, 2003.

[8] D.E. Knuth. *The Art of Computer Programming*. Addison-Wesley, second ed., Reading, MA, 1998.

[9] K. Malanowski and H. Maurer. Sensitivity analysis for parametric control problems with control-state constraints. *Computational Optimisation and Applications*, 5:253–283, 1996.

[10] H. Maurer and D. Augustin. Sensitivity analysis and real-time control of parametric optimal control problems using boundary value methods. *Online Optimization of Large Scale Systems, Springer-Verlag*, pages 17–56, 2001.

[11] H. Maurer and S. Pickenhain. Second order sufficient conditions for optimal control problems with mixed control-state constraints. *Journal Optimisation Theory and Applications*, 86:649–667, 1995.

[12] J. Pantoja. Differential dynamic programming and Newton's method. *International Journal on Control*, 47:1539–1553, 1988.

[13] J. F. A. De O. Pantoja. *Algorithms for Constrained Optimization Problems*. PhD thesis, Imperial College of Science and Technology, University of London, 1983.

[14] S. Pickenhain. Sufficiency conditions for weak local minima in multidimensional optimal control problems with mixed control-state restrictions. *Zeitschrift für Analysis und ihre Anwendungen*, 11:559–568, 1992.

[15] J. Sternberg. Adaptive Umkehrschemata für Schrittfolgen mit nicht-uniformen Kosten. *Diploma thesis, Institute of Scientific Computing, TU Dresden*, 2002.

[16] J. Sternberg. Reduction of storage requirement by checkpointing for time-dependent optimal control problems. *PhD Thesis, Institute of Scientific Computing, TU Dresden*, 2005.

[17] J. Sternberg and A. Griewank. Logarothmic growth of temporal and spatial complexity in single and double reversal. *In preparation*.

[18] J. Sternberg and A. Griewank. Reduction of storage requirement by checkpointing for time-dependent optimal control problems in ODEs. In M. Bücker, G. Corliss, P. Hovland, U. Naumann, and B. Norris, editors, *Automatic Differentiation: Applications, Theory, and Implementations*, pages 99–110. Springer, 2006.

[19] A. Walther. *Program Reversal Schedules for Single- and Multi-processor Machines*. PhD thesis, Institute of Scientific Computing, TU Dresden, 1999.

[20] A. Walther. Program reversals for evolutions with non-uniform step costs. *Acta Informatica*, 40:235–263, 2004.

[21] V. Zeidan. The riccati equation for optimal control problems with mixed state-control constraints: necessity and sufficiency. *SIAM Journal Control and Optimization*, 32:1297–1321, 1994.