Technische Universität Dresden
Fachrichtung Mathematik
Institut für Wissenschaftliches Rechnen

Memory efficient approaches of second order for optimal control problems

Dissertation
zur Erlangung des zweiten akademischen Grades

Dr.rer.nat.

vorgelegt von

Name:	Sternberg	Vorname:	Julia
geboren am:	22. Mai 1978	in:	Rubzowsk

Tag der Einreichung:

Gutachter:

First of all, I thank my supervisor Prof. Andreas Griewank for his encouragement, support, excellent advices and many fruitful discussions over the last three years. Also, I am very grateful to Prof. Michael Hinze and Prof. Dietmar Hömberg for their helpful suggestions and advices during my work.

Moreover I thank my colleagues at the Institute of Scientific Computing, Technical University Dresden, for the pleasant working atmosphere. Financial support by “Sächsisches Landesstipendium” is gratefully acknowledged.

Finally, I thank my family for the support and encouragement at home.

Contents

Notations

$\mathcal{E}_{N\times l}$	Multiple sweep evolution
$\mathbf{S}$	Reversal schedule
l	Number of time steps
N	Number of alternative sweeps
s	Number of checkpoints
F, $\bar{F}$, $\bar{\bar{F}}$	Forward, adjoint, and final simulations
F_i, $\bar{F}_i$, $\bar{\bar{F}}_i$	Forward, adjoint, and final steps
x_i, $\bar{x}_i$, $\bar{\bar{x}}_i$	Forward, adjoint, and final states
t_i, $\bar{t}_i$, $\bar{\bar{t}}_i$	Forward, adjoint, and final step cost
r_i, $\bar{r}_i$, $\bar{\bar{r}}_i$	Forward, adjoint, and final repetition numbers
$r_{max}(\mathbf{S})$	Maximal repetition number for $\mathbf{S}$
$s_{max}(\mathbf{S})$	Maximal number for checkpoints
$l_{max}(\mathbf{S})$	Maximal number for time steps
$l_{max}(s,r)$	$\beta(s,r) = \binom{s+r}{s} = \frac{(s+r)!}{s!\,r!}$
$r_{bin}(s,l)$	The unique integer satisfying:
	$\beta(s, r_{bin}(s,l) - 1) < l \leq \beta(s, r_{bin}(s,l))$
$m_j(\mathbf{S})$	Frequency number for $\mathbf{S}$
$m_j(l,s)$	$\beta(s-1,j)$
$\mathbf{t_l}$	Step cost distribution for l time steps
$\mathbf{t_{N\times l}}$	Step cost distribution for $\mathcal{E}_{N\times l}$
$\mathbf{d_{N\times l}}$	Size distribution for $\mathcal{E}_{N\times l}$
$\mathbf{r}(\mathbf{S})$	Repetition number profile for $\mathbf{S}$
$\mathbf{m}(\mathbf{S})$	Frequency number profile for $\mathbf{S}$
$\vec{\mathbf{r}}_{max}(\mathbf{S})$	Vector of maximal repetition numbers for $\mathbf{S}$
$\mathbf{S}_{min}(l,s)$	Optimal reversal schedule
$\mathbf{S}_{bin}(l,s)$	Binomial reversal schedule
TIME	Run-time functional
$\mathbf{T}(\mathbf{S})$	Temporal complexity for $\mathbf{S}$
$\mathbf{T}_{min}(l,s)$	Minimal temporal complexity
$\mathbf{T}_{bin}(l,s)$	Temporal complexity for $\mathbf{S}_{bin}(l,s)$

Chapter 1

Introduction

Consider a time-dependent optimal control problem

$$\min_{\mathbf{x},\mathbf{u}} \phi(\mathbf{x}(T)), \tag{1.1}$$

where the state evolution is described by an initial value problem

$$\dot{\mathbf{x}} = f(\mathbf{x}(t), \mathbf{u}(t), t), \quad \mathbf{x}(0) = \mathbf{x}_0. \tag{1.2}$$

Here, $\mathbf{x} : [0, T] \to \mathbf{R}^n$, $\mathbf{u} : [0, T] \to \mathbf{R}^m$, $f : \mathbf{R}^n \times \mathbf{R}^m \times [0, T] \to \mathbf{R}$, and $\phi : \mathbf{R}^n \to \mathbf{R}$.

The task is to find the function $\mathbf{u}(t)$ that minimises (1.1). In order to characterise an optimal control function $\mathbf{u}(t)$ for the minimisation problem (1.1) and (1.2), consider the following adjoint state equation

$$\dot{\bar{\mathbf{x}}} = -H_{\mathbf{x}}^T = -f_{\mathbf{x}}^T \, \bar{\mathbf{x}}, \qquad \bar{\mathbf{x}}(T) = \phi_{\mathbf{x}}^T(\mathbf{x}(T)), \tag{1.3}$$

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

$$H(\mathbf{x}(t), \mathbf{u}(t), \bar{\mathbf{x}}(t), t) = \bar{\mathbf{x}}^T(t) \, f(\mathbf{x}(t), \mathbf{u}(t), t). \tag{1.4}$$

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

$$\left(\frac{\partial H}{\partial \mathbf{u}}\right)^T = 0, \qquad 0 \le t \le T, \tag{1.5}$$

for the optimal control problem (1.1) - (1.2).

A large variety of numerical methods for solving optimal control problems has been proposed and used in various fields of applications [9, 17, 18, 19, 20]. Their relations amongst each other are often not very clear due to the lack of a generally accepted terminology. One fairly popular concept is to juxtapose approaches that first discretise and then optimise with those that first optimise and then discretise. Methods of the first type are sometimes called **direct** (see e.g. [20]), as they treat the discretised control problem immediately as a finite dimensional nonlinear program, which can be handed over to increasingly sophisticated and robust NLP codes.

In the alternative approach, one firstly derives optimality conditions in a suitable function space setting and then discretises the resulting boundary value problem with algebraic side constraints. Often such **indirect** methods (see e.g. [65]) yield highly accurate results, but they have some disadvantages as well. Sometimes it is not possible to construct the boundary value problem explicitly, as it requires that one can express the control function $\mathbf{u}$ in terms of $\mathbf{x}$ and $\bar{\mathbf{x}}$ from the relation (1.5). The second disadvantage is that often one has to find a very good initial guess including good estimates for the adjoint variables to achieve convergence to the solution. Alternatively, one can solve the problem as a DAE with (1.5) representing a possibly discontinuous algebraic constraint [15]. Obviously, there is a range of intermediate strategies, as one may for example discretise at first the controls and then, the states only later. The class of indirect methods is considered in this thesis more detailed.

The boundary value problem (1.2) - (1.3) is in general non-linear with separated boundary conditions. A quasilinearisation scheme is used to solve it iteratively. First, one linearises (1.2) - (1.3), and (1.5); after that, the resulting linear boundary value problem is solved using one of multiple shooting approaches (see e.g. [3, 11, 12, 13, 71, 86]). It will be shown how the memory requirement can be kept within reasonable bounds nevertheless.

The so-called Pantoja method ([24, 25, 30, 67, 73]) describes a computationally efficient stage-wise construction of the Newton direction for the discrete-time optimal control problem. There are many relationships between multiple shooting techniques and Pantoja method. These relationships will be clarified in this thesis. Moreover, an efficient implementation of these approaches using checkpointing techniques will be presented. For this purpose, the well known aspect of checkpointing has to be extended.

The concept of checkpointing is introduced and investigated in [8, 32, 35, 87]. Here, it is assumed that a forward simulation F consisting of l consecutive time steps has to be reversed. This well-known technique is closely related to the reverse mode of algorithmic differentiation (AD) [35], also called automatic or computational differentiation. Automatic differentiation (AD) is a set of techniques for obtaining derivatives of numerical functions to the same order of accuracy as the function values themselves, but without the labour of forming explicit symbolic expressions for the derivative functions. For the adjoint mode of AD it is necessary to reverse the forward simulation F (see [8, 35]). It was shown already in 1973 (see [6]) that a logarithmical growth in the spatial complexity might be achievable. Moreover, in [34] it was shown that the growth of the temporal complexity, needed to reverse a given simulation F, can be arranged logarithmically as well. For this purpose, optimal checkpointing strategies were developed: see e.g. [35, 38] for single-processor machines and [91] for multi-processor machines. Furthermore, for cases, where the number l of time steps to be reversed is not known a-priory, adaptive checkpointing techniques were established: see [83] for single-processor machines and [60] for multi-processor machines.

The checkpointing techniques can be applied upon other methods to the memory reduced implementation of approaches for solving boundary value problems. These are e.g. some types of multiple shooting methods. This is because these methods feature a structure consisting of two alternative sweeps. One of them has to proceed in the forward direction, the other one in the backward direction.

The well-known checkpointing approach is extended to a 'nested checkpointing' for multiple transversals. The motivation behind this is to efficiently implement Pantoja, Riccati, or some other types of multiple shooting methods. Here, one has more than two alternative sweeps to be going through. This particular type of simulations requires specific checkpointing techniques for its memory reduced implementation. This is because each iteration of these methods can be divided in three alternative sweeps. Some of multiple shooting techniques applied to the optimal control problem as well as Pantoja approach possess this property. Therefore, for the memory reduced implementation of these algorithms, nested checkpointing techniques are necessary.

This thesis is organised as follows. In Chapter 2 checkpointing techniques for double sweep simulations are introduced. Here, reversal schedules are classified and defined. Moreover, notations, definitions, and some basic results are introduced. They will be needed for the further development of nested checkpointing later on.

Chapter 3 gives a survey of numerical methods for separated boundary value problems. Here, it is shown how these methods can be applied to the solution of unconstrained optimal control problems. Moreover, some numerical tests for their memory reduced implementation using serial checkpointing are carried out. The resulting run-times are presented. Furthermore, in this chapter the equivalence between Pantoja method and the multiple shooting method of Riccati type is shown. Moreover, Pantoja method is extended to the case where the state equations (1.2) are discretised using one of implicit numerical methods. The concept of symplecticness and Hamiltonian systems is introduced. In this regard, a suitable numerical method is presented, which can be applied to unconstrained optimal control problems. Moreover, it is proved that this method is a symplectic one.

Nested checkpointing techniques are developed in Chapter 4. Firstly, their main properties are introduced. Then, a special case of multiple sweep evolutions, consisting of three alternative sweeps, is considered. Here, firstly main characteristics of nested reversal schedules for this type of evolutions are discussed. After that, a search algorithm for determining optimal nested reversal schedules by exhaustive search techniques is presented. Minimal temporal complexity, required to implement optimal nested reversal schedules can be estimated using the lower bound evaluated in this chapter. Moreover, for an efficient construction of nested reversal schedules some heuristics are introduced. Their benefits are discussed. Moreover, their results are compared to the optimal schedules computed by exhaustive search techniques. The temporal complexity, necessary to construct corresponding nested reversal schedules, using these heuristics, is reduced to a linear behaviour in the number of time steps and checkpoints available. Moreover, it is shown in this chapter, that the temporal complexity, required to implement these nested reversal schedules, grows as a small power of the natural logarithm of the number of time steps. Finally, software available to implement nested reversal schedules is introduced.

Chapter 5 gives numerical examples. Here, among other problems, the optimal problem of laser surface hardening of steel is considered (see [52, 53, 88, 89, 90]). Pantoja approach for implicit discretisation of state equations, introduced in Chapter 3, is applied to the solution of this problem. For the memory reduced implementation of this approach nested reversal schedules, developed in Chapter 4, are used. Some numerical results and run-time tests are given.

Finally, in Chapter 6 some conclusions are presented. Moreover, some proposals regarding further theoretical investigations and numerical applications are formulated.

Chapter 2

Checkpointing Techniques

The key difficulty in adjoining large evaluation programs is their reversal within a reasonable amount of memory. If the total number of intermediate states is to be stored, the memory requirement will be proportional to runtime needed to run a program and can be enormous. A remedy for these memory problems is the so-called **Checkpointing**. There are many theoretical techniques regarding this technique. Definitions, notations and basic results are introduced in this chapter.

2.1 Notations and Definitions

Many mathematical applications contain a **forward simulation**. An example for a forward simulation is a time discretisation of an ODE or a PDE. A forward simulation is denoted F.

A certain sequence of intermediate states is to be computed during the evaluation of a forward simulation F. Each action from a previous intermediate state to the next intermediate state can be seen as a **time step** F_i. Therefore, without loss of generality, we can say that the evaluation of a given forward simulation F can be represented as a computation of l time steps $F_i,\ 1 \leq i \leq l$. A certain number of assignments are to be combined to a single step. Figure 2.1 illustrates this situation. Time steps $F_i,\ 1 \leq i \leq l$, are depicted by arrows. An intermediate state x_i is a large dimensional vector. The intermediate state x_i can be evaluated if the time step F_i is applied to the previous state x_{i-1}. The time step F_i evaluates the intermediate state x_i using only data from the previous state x_{i-1}.

If we solve an adjoint problem associated with a given forward simulation F,

$$\underbrace{x_0 \xrightarrow{F_1} x_1 \xrightarrow{F_2} \cdots \xrightarrow{F_i} x_i \xrightarrow{F_{i+1}} \cdots \xrightarrow{F_{l-1}} x_{l-1} \xrightarrow{F_l} x_l}_{F}$$

Figure 2.1: Forward simulation F

or if we apply the adjoint mode of the algorithmic or automatic differentiation, it is necessary to reverse this forward simulation F.

The basic method to invert a forward simulation F is to store all intermediate states x_i, $0 \leq i \leq l$, on a sequential data file and to restore them if required. In the literature this method is called the **basic approach**.

The memory requirement for the basic approach is proportional to the runtime of a forward simulation F, i.e. to the number l of time steps. Therefore, it could be huge. Practical applications of the basic approach are restricted in spite of the fact that the memory capacity grows. This is the sort of problems, checkpointing deals with.

Another approach for the reversal of a forward simulation F does not require to store all intermediate states, but a smaller number of them. Stored intermediate states are called **checkpoints**. Consequently, the method is called **checkpointing**. Checkpointing is introduced and investigated in [8, 32, 35, 87]. Depending on an available memory capacity, we can store only a small number of intermediate states at the same time. Therefore, this approach enormously reduces the memory requirement.

Using checkpointing, data of selected intermediate states is to be stored as checkpoints. Thus, during the reversal of a forward simulation F, the evaluation of some time steps as well as some intermediate states could occur several times. Consequently, the evaluation procedure has to be carried out repeatedly. There is no need to restart a forward simulation F from the initial intermediate state. It can be restarted from a suitable checkpoint.

In order to provide the reversal or the adjoint procedure of a forward simulation F, we expand it by adding to each time step F_i an additional **adjoint** step $\bar{F}_i$. The resulting adjoint simulation is denoted $\bar{F}$. In the following, we denote a forward simulation F an original sweep and an adjoint simulation $\bar{F}$ an adjoint sweep. Therefore, corresponding time steps and intermediate states are denoted time steps or intermediate states and adjoint steps or adjoint states, for original and adjoint sweeps, respectively. The situation is depicted in Figure 2.2, which is to be seen as an extension of Figure 2.1. Here, to each time step F_i an adjoint step $\bar{F}_i$ is associated. A corresponding

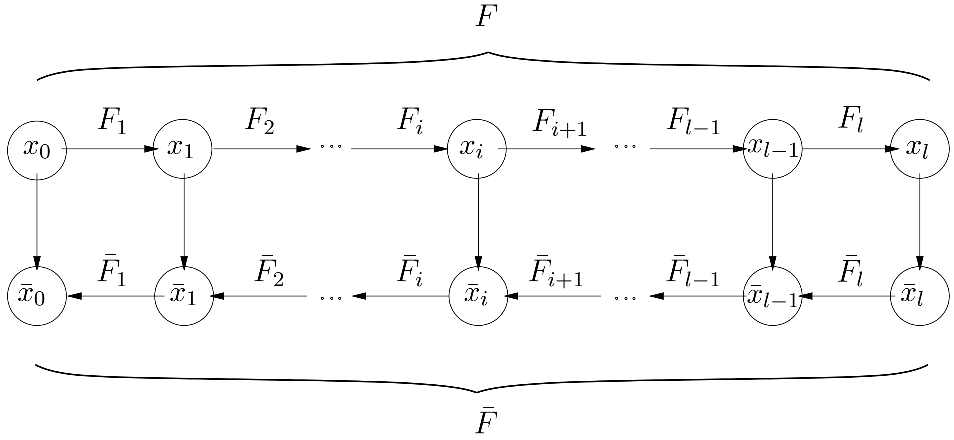

Figure 2.2: Forward simulation F and its associated adjoint simulation $\bar{F}$

adjoint state, associated with the intermediate state x_i, is represented by the counter $\bar{x}_i$. The adjoint state $\bar{x}_{i-1}$ can be evaluated after the time step F_i is applied to the intermediate state x_{i-1} in order to evaluate the consecutive intermediate state x_i. Data, produced during this evaluation, is to be stored on a tape. After that, the adjoint state $\bar{x}_{i-1}$ can be evaluated using data of the consecutive adjoint state $\bar{x}_i$ and recently stored data from the evaluation F_i. Without loss of generality, we combine these assignments to the adjoint step $\bar{F}_i$. Thus, the adjoint step $\bar{F}_i$ evaluates the adjoint state $\bar{x}_{i-1}$ using data from the consecutive adjoint state $\bar{x}_i$ as well as data produced during the evaluation of the intermediate state x_i. The adjoint step $\bar{F}_i$ can be evaluated only if data of the intermediate state x_{i-1} and data of the consecutive adjoint state $\bar{x}_i$ is available.

2.1.1 Characterisation of Step Cost

Provided that run-time is associated with a step cost, the overall run-time requirement for the evaluation of the original sweep F can be expressed as a sum of run-times of single time steps, i.e.

$$TIME(Evaluation\ F) = \sum_{i=1}^{l} TIME(Evaluation\ F_i). \tag{2.1}$$

Moreover, we introduce evaluation costs, i.e. the computational effort for intermediate steps of different sweeps

$$t_i = TIME(F_i), \quad \bar{t}_i = TIME(\bar{F}_i), \quad 1 \leq i \leq l. \tag{2.2}$$

Basically, we distinguish between two cases of step costs, uniform and non-uniform case.

Definition 2.1 *Consider a forward simulation F. This comprises l time steps F_i, $1 \leq i \leq l$. If there is a constant w, so that $t_i = w$ for all time steps F_i, $1 \leq i \leq l$, then, F denotes a forward simulation with* **uniform costs***. Otherwise, we speak about a forward simulation with* **non-uniform costs***.*

A step cost distribution for a forward simulation F is defined by the set

$$\mathbf{t_l} = \langle t_1, ..., t_l \rangle^\top. \tag{2.3}$$

$\mathbf{t_l}$ contains exactly l elements t_i, $1 \leq i \leq l$. In non-uniform case there are at least two indexes i, j, $i \neq j$, so that $t_i \neq t_j$, $1 \leq i, j \leq l$. Without loss of generality, we set $w = 1$ in uniform case. Thus, the uniform step cost distribution is given by

$$\mathbf{t_l} = \langle 1, ..., 1 \rangle^\top, \tag{2.4}$$

with $t_i = 1$, $1 \leq i \leq l$.

2.1.2 Definition of Reversal Schedules

The reversal of a forward simulation F can be interpreted as a sequence of assignments. They are to combine into appropriate actions. Each action defines which time step is to evaluate, which intermediate state is to store or restore from a corresponding checkpoint. A certain combination of actions represents a **reversal schedule**. More formally, use the following definition.

Definition 2.2 (Reversal Schedule S) *Consider a forward simulation F which comprises l time steps F_i, $1 \leq i \leq l$. Let $s \in \mathbb{N}$ checkpoints be available each of which can accommodate data of a single intermediate state. l denotes the number of the currently final time step, i denotes the current state and j denotes the number of stored checkpoints. Initially, $i = i_0 \geq 0$, $j = j_0 \geq 0$, and $i < l \leq \infty$. Then, a reversal schedule* $\mathbf{S}$ *consists of a finite sequence of the following actions:*

A	$=$	*If* $i < l-1$, *carry out the time step* F_{i+1} *and set* $i = i+1$.
W_k	$=$	*Write data of the current intermediate state into the checkpoint* $j_0 + k$.
R_k	$=$	*Read out an intermediate state from the checkpoint* $j_0 + k$ *and set the counter* i *to its number.*
U	$=$	*If* $0 \leq l-1 = i$, *carry out the adjoint step* $\bar{F}_{i+1}$ *and set* $l = l-1$, *else terminate.*

We normally assume that all reversal schedules terminate successfully in that $l = 0$ at the end. We then say that the schedule is admissible for the given combination (l, s). Without loss of generality, we assume that each admissible reversal schedule $\mathbf{S}$ begins with the action R_0, so that the initial intermediate state is read out from the checkpoint j_0. If an action A is carried out up to k times, it is denoted A^k.

Note that some results from [35, 38, 91] are already built in Definition 2.2. In particular, a reversal schedule $\mathbf{S}$ is defined in such a manner that each adjoint step $\bar{F}_i,\ 1 \leq i \leq l$, is evaluated exactly one time. Therefore, the overall run-time requirement $\bar{t}$ for the evaluation of all adjoint steps $\bar{F}_i,\ 1 \leq i \leq l$, is defined by

$$\bar{t} = \sum_{i=1}^{l} \bar{t}_i. \tag{2.5}$$

Furthermore, for each reversal schedule $\mathbf{S}$, $l_{max}(\mathbf{S})$ denotes the maximum of the achieved index $(i - i_0)$ and $s_{max}(\mathbf{S})$ denotes the maximum of stored checkpoints, i.e. the maximum of $(j - j_0 + 1)$.

Figure 2.3 illustrates an example of a reversal schedule $\mathbf{S}$ with $l_{max}(\mathbf{S}) = 10$ and $s_{max}(\mathbf{S}) = 3$, given that $i_0 = 0$ and $j_0 = 0$. It is assumed that the evaluation of all time steps $F_i,\ 1 \leq i \leq 10$, and all adjoint steps $\bar{F}_i,\ 1 \leq i \leq 10$, requires uniform temporal complexity $w = 1$ and $w = 2$, respectively. Time steps $F_i,\ 1 \leq i \leq 10$, are plotted along the vertical y-axis, whereas time required for the evaluation of single actions is plotted along the horizontal x-axis. Time is measured in w-units. Horizontal lines in Figure 2.3 represent checkpoints. Since the initial checkpoint contains data of the intermediate state x_0 and lies on the computational axis, this axis represents a checkpoint itself.

The evaluation of the reversal schedule $\mathbf{S}$ in Figure 2.3 can be described as follows. Firstly, the action R_0 is performed, i.e. the initial intermediate state x_0 is read out from the first checkpoint $j_0 = 0$. After that, the four time steps F_1, F_2, F_3, and F_4 are evaluated consecutively, which can be described

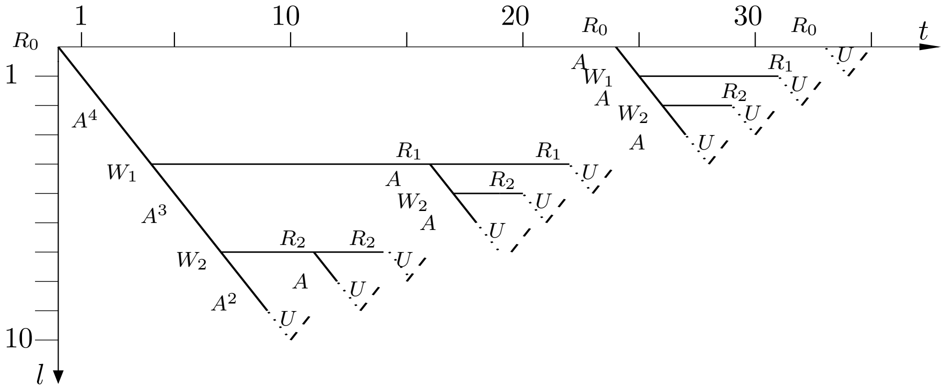

Figure 2.3: Reversal schedule $\mathbf{S}$ with $l_{max}(\mathbf{S}) = 10$ and $s_{max}(\mathbf{S}) = 3$

by the action A^4. Then, the intermediate state x_4 is stored into the first checkpoint by the action W_1. The following 3 time steps F_5, F_6, and F_7 are carried out by the action A^3. The intermediate state x_7 is stored into the second checkpoint by the action W_2. The following two time steps F_8 and F_9 are carried out by the action A^2 consecutively. After that, the action U is applied for the first time, so that the adjoint step $\bar{F}_{10}$ is evaluated. Then, data of the intermediate state x_7 is read out from the second checkpoint by the action R_2. Using this data, the time step F_8 is evaluated by the action A. Now, it is possible to perform the adjoint step $\bar{F}_9$ using the action U. 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 $\mathbf{S}$ in Figure 2.3 can be represented by the following sequence of actions:

$$
\begin{aligned}
\mathbf{S} \;=\;& R_0 \triangleright A^4 \triangleright W_1 \triangleright A^3 \triangleright W_2 \triangleright A^2 \triangleright U \triangleright R_2 \triangleright A \triangleright R_2 \triangleright U \triangleright \\
& \triangleright R_1 \triangleright A \triangleright W_2 \triangleright A \triangleright U \triangleright R_2 \triangleright U \triangleright R_1 \triangleright U \triangleright R_0 \triangleright A \triangleright \\
& \triangleright W_1 \triangleright A \triangleright W_2 \triangleright U \triangleright R_2 \triangleright U \triangleright R_1 \triangleright U \triangleright R_0 \triangleright U.
\end{aligned}
\tag{2.6}
$$

The operator '$\triangleright$' between two actions can be interpreted in the following way: the left action is evaluated first, the action on the right hand side is evaluated second.

The reversal schedule $\mathbf{S}$ in Figure 2.3 can be executed using 35 w-units. During the evaluation of $\mathbf{S}$, memory for 3 checkpoints is required. Moreover, memory for the storage of data produced during a single time step F_i, $1 \leq i \leq 10$, which is required for the performing of a corresponding adjoint step $\bar{F}_i$, $1 \leq i \leq 10$, is needed. If the basic approach is applied, 20 w-units will be needed to perform ten adjoint steps $\bar{F}_i$, $1 \leq i \leq 10$, but all intermediate states x_i, $0 \leq i \leq 10$, as well as data produced during all single time steps F_i, $1 \leq i \leq 10$, which is required for the performance of corresponding adjoint steps $\bar{F}_i$, $1 \leq i \leq 10$, will be stored.

Now, a naturally raised question is, which reversal schedule from the total amount of possible reversal schedules for a forward simulation F and s available checkpoints is an optimal one. The second question is, how to construct an appropriate optimal reversal schedule. In order to find answers for these questions, we introduce some needful characteristic values first.

2.1.3 Repetition Number and Profile

For a reversal schedule $\mathbf{S}$, it is convenient to determine a number of evaluations for each time step.

Definition 2.3 (Repetition Number) *Consider a reversal schedule* $\mathbf{S}$ *with* $l_{max}(\mathbf{S}) = l$. *The* **repetition number** $r_i = r(i)$, *defined by the function*

$$r : [1, l] \to \mathbb{N}, \tag{2.7}$$

counts how often the time step F_i *is evaluated during the execution of the reversal schedule* $\mathbf{S}$.

Definition 2.3 can be interpreted regarding computations of intermediate states. Thus, r_i denotes how often the intermediate state x_i, $1 \leq i \leq l$, is computed during the execution of the reversal schedule $\mathbf{S}$.

Definition 2.4 (Repetition Profile) *Consider a reversal schedule* $\mathbf{S}$ *with* $l_{max}(\mathbf{S}) = l$. *The sequence of repetition numbers*

$$\mathbf{r}(\mathbf{S}) = \langle r_1, ..., r_l \rangle^\top \tag{2.8}$$

defines the **repetition profile** $\mathbf{r}(\mathbf{S})$ *for the reversal schedule* $\mathbf{S}$.

Definition 2.5 (Maximal Repetition Number) *For a reversal schedule* $\mathbf{S}$ *with* $l_{max}(\mathbf{S}) = l$ *the* **maximal repetition number** $r_{max}(\mathbf{S})$ *is defined by*

$$r_{max}(\mathbf{S}) = \|\mathbf{r}(\mathbf{S})\|_\infty = \max_{1 \leq i \leq l} r_i. \tag{2.9}$$

For the reversal schedule $\mathbf{S}$ in Figure 2.3 the corresponding repetition profile is given by

$$\mathbf{r}(\mathbf{S}) = \langle 2, 2, 2, 1, 2, 2, 1, 2, 1, 0 \rangle^\top. \tag{2.10}$$

This profile is illustrated in Figure 2.4. The maximal repetition number $r_{max}(\mathbf{S})$ reaches 2.

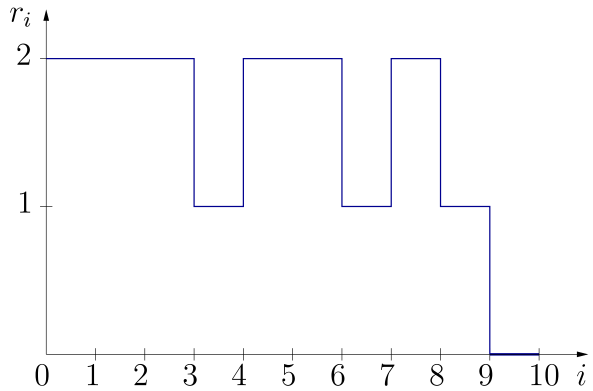

Figure 2.4: Repetition profile for the reversal schedule $\mathbf{S}$ in Figure 2.3

2.1.4 Frequency Numbers

For each reversal schedule $\mathbf{S}$ with $l_{max}(\mathbf{S}) = l$, it is convenient to determine a number of time steps $F_i,\ 1 \leq i \leq l$, which are evaluated $j,\ 0 \leq j \leq r_{max}(\mathbf{S})$, times during the execution of the reversal schedule $\mathbf{S}$. More formally, use the following definition.

Definition 2.6 (Frequency Number) *Consider a reversal schedule* $\mathbf{S}$ *with* $l_{max}(\mathbf{S}) = l$. **Frequency numbers** $m_j(\mathbf{S}),\ 0 \leq j \leq r_{max}(\mathbf{S})$, *defined by the function*

$$m_j(\mathbf{S}) \ = \mid \{r_i = j,\ 1 \leq i \leq l\} \mid = \mid r^{-1}(j) \mid, \tag{2.11}$$

count how many steps are evaluated j *times during the execution of the reversal schedule* $\mathbf{S}$.

The operator $| \cdot |$ in Definition 2.6 determines the cardinal number of a given set. Obviously, for a reversal schedule $\mathbf{S}$ the following equation is satisfied

$$\sum_{j=0}^{r_{max}(\mathbf{S})} m_j(\mathbf{S}) = l. \tag{2.12}$$

Definition 2.7 (Frequency Profile) *Consider a reversal schedule* $\mathbf{S}$ *with* $l_{max}(\mathbf{S}) = l$. **Frequency profile** $\mathbf{m}(\mathbf{S})$ *is defined by the sequence*

$$\mathbf{m}(\mathbf{S}) = \langle m_0(\mathbf{S}), ..., m_{r_{max}(\mathbf{S})}(\mathbf{S})\rangle^{\top}, \tag{2.13}$$

where $r_{max}(\mathbf{S})$ *denotes the maximal repetition number of the reversal schedule* $\mathbf{S}$.

The frequency profile $\mathbf{m}(\mathbf{S})$ for the reversal schedule $\mathbf{S}$ in Figure 2.3 is given by

$$\mathbf{m}(\mathbf{S}) = \langle m_0(\mathbf{S}), m_1(\mathbf{S}), m_2(\mathbf{S})\rangle^\top = \langle 1, 3, 6\rangle^\top. \tag{2.14}$$

This profile is illustrated in Figure 2.5.

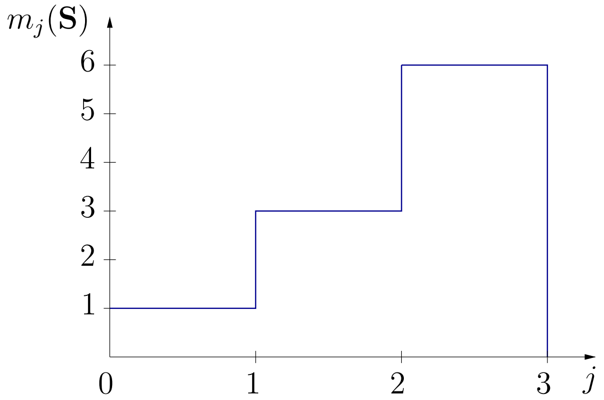

Figure 2.5: Frequency profile for the reversal schedule $\mathbf{S}$ in Figure 2.3

2.1.5 Temporal Complexity of Reversal Schedules

The temporal complexity of a reversal schedule $\mathbf{S}$, i.e. the run-time effort, which is required to execute the reversal schedule $\mathbf{S}$, is defined by

$$\mathbf{T}(\mathbf{t_l}, \mathbf{S}) = \begin{cases} \mathbf{r}^\top(\mathbf{S}) \cdot \mathbf{t_l} = \sum\limits_{i=1}^{l} r_i t_i & : \quad if\ l \le l_{max}(\mathbf{S}), \\ \infty & : \quad else. \end{cases} \tag{2.15}$$

$\mathbf{t_l} = \langle t_1, ..., t_l\rangle^\top$ denotes a step cost distribution of a sequence of l time steps $F_i,\ 1 \le i \le l$. Here, we have omitted the constant effort $\sum\limits_{i=1}^{l} \bar{t}_i$ for executing all adjoint steps exactly once.

In uniform case the temporal complexity $\mathbf{T}(\mathbf{S})$ is defined by

$$\mathbf{T}(\mathbf{S}) = \sum_{i=1}^{l} r_i, \qquad if \qquad l \le l_{max}(\mathbf{S}). \tag{2.16}$$

The sum (2.16) gives the overall number of time steps $F_i,\ 1 \le i \le l$, evaluated during the execution of a corresponding reversal schedule $\mathbf{S}$. Using

Definition 2.6 of frequency numbers, (2.16) can be represented in the following way:

$$\mathbf{T}(\mathbf{S}) = \sum_{j=0}^{r_{max}(\mathbf{S})} j \cdot m_j(\mathbf{S}). \tag{2.17}$$

For the example of a reversal schedule $\mathbf{S}$ in Figure 2.3 the temporal complexity $\mathbf{T}(\mathbf{S})$ attains $\mathbf{T}(\mathbf{S}) = 15$.

The temporal complexity $\mathbf{T}(\mathbf{S})$, which results from the execution of a reversal schedule, determines the optimality of the corresponding reversal schedule. Therefore, the temporal complexity is one of the most significant properties for reversal schedules.

2.1.6 Decomposition of Reversal Schedules

To establish explicit formulae for quantities introduced in previous sections reversal schedules will be decomposed into smaller substructures which will be considered separately. To make this, the following assertion is necessary.

Lemma 2.1 (Checkpoint Persistency) *Consider a forward simulation F. This comprises l time steps with a step cost distribution $\mathbf{t_l}$. Let $\mathbf{S}$ be a reversal schedule with $l_{max}(\mathbf{S}) = l$ and $s_{max}(\mathbf{S}) = s$. $\mathbf{T}(\mathbf{t_l}, \mathbf{S})$ denotes the evaluation cost of the reversal schedule $\mathbf{S}$ applied to the implementation of the forward simulation F. Then, $\mathbf{S}$ can be modified without increasing the cost $\mathbf{T}(\mathbf{t_l}, \mathbf{S})$, so that the following properties hold. After the checkpoint writing W_j, storing the intermediate state x_i into the checkpoint j, the next action W_j occurs only when l has already been reduced to i or below. Moreover, until that time no actions involving the intermediate states between x_0 and x_i are taken.*

Proof: For the proof of this lemma see [35, 91].

Using the checkpoint persistency, each reversal schedule $\mathbf{S}$ can be decomposed into two subschedules $\mathbf{S}^1$ and $\mathbf{S}^2$. This can be done by storing the intermediate state $x_{\check{l}}$ into the checkpoint $(j_0 + 1)$ using the action W_1. The reversal schedule $\mathbf{S}^1$ represents a set of actions regarding the time steps $F_1, ..., F_{\check{l}}$, whereas the reversal schedule $\mathbf{S}^2$ contains actions regarding the remaining time steps $F_{\check{l}+1}, ..., F_l$. Obviously, $l_{max}(\mathbf{S}^1) = \check{l}$ and $l_{max}(\mathbf{S}^2) = l - \check{l}$. This decomposition is illustrated in Figure 2.6. This can be defined by the following binary composition

$$\mathbf{S} = \mathbf{S}^1 \circ \mathbf{S}^2. \tag{2.18}$$

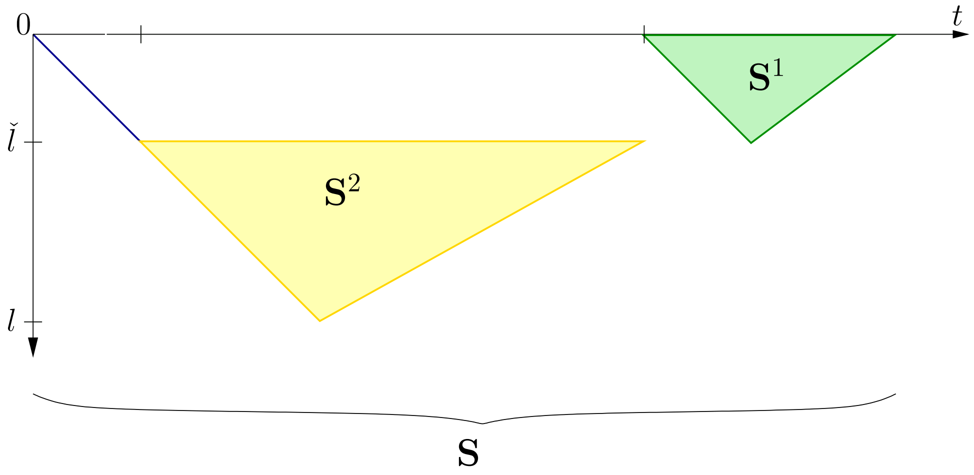

Figure 2.6: Decomposition of reversal schedule $\mathbf{S} = \mathbf{S}^1 \circ \mathbf{S}^2$

The operator $\circ$ in (2.18) is to be interpreted in the following way: actions representing the reversal schedule $\mathbf{S}^1$ affect time steps $F_1, ..., F_{\check{l}}$; actions representing the reversal schedule $\mathbf{S}^2$ affect time steps $F_{\check{l}+1}, ..., F_l$. This operator $\circ$ is certainly not commutative. Also, in contrast to $\triangleright$, the operator $\circ$ is not associative. This can be seen in the following example.

Example 2.1 *Suppose, we have three different reversal schedules $\mathbf{S}^1$, $\mathbf{S}^2$, and $\mathbf{S}^3$. If the operator $\circ$ is associative, then, the following equation will be valid.*

$$(\mathbf{S}^1 \circ \mathbf{S}^2) \circ \mathbf{S}^3 = \mathbf{S}^1 \circ (\mathbf{S}^2 \circ \mathbf{S}^3). \tag{2.19}$$

Firstly, we denote $s_{max}(\mathbf{S}^i)$, $i = 1, 2, 3$, the maximal number of checkpoints available throughout the evaluation of a reversal schedule $\mathbf{S}^i$, $i = 1, 2, 3$. Therefore, using the definition of the binary decomposition (2.18) above, we obtain the following relations from the left hand side of the equation (2.19).

$$\begin{aligned} s_{max}(\mathbf{S}^2) &= s_{max}(\mathbf{S}^1) - 1, \\ s_{max}(\mathbf{S}^3) &= s_{max}(\mathbf{S}^1 \circ \mathbf{S}^2) - 1 = s_{max}(\mathbf{S}^1) - 1. \end{aligned} \tag{2.20}$$

If we apply similar considerations to the right-hand side of the equation (2.19), we obtain the following relations.

$$\begin{aligned} s_{max}(\mathbf{S}^3) &= s_{max}(\mathbf{S}^2) - 1, \\ s_{max}(\mathbf{S}^2 \circ \mathbf{S}^3) &= s_{max}(\mathbf{S}^2) = s_{max}(\mathbf{S}^1) - 1, \\ s_{max}(\mathbf{S}^3) &= s_{max}(\mathbf{S}^1) - 2. \end{aligned} \tag{2.21}$$

Therefore, if we compare the two last equations (2.20) and (2.21), then, we obtain that if (2.19) is true, i.e. if the operator $\circ$ *is associative, then, the following equation will be satisfied.*

$$s_{max}(\mathbf{S}^1) - 1 = s_{max}(\mathbf{S}^1) - 2. \tag{2.22}$$

Obviously, the relation (2.22) does not hold. Therefore, the operator $\circ$ *is not associative.*

An example for a reversal schedule $\mathbf{S}$ *given by the decomposition* $\mathbf{S} = (\mathbf{S}^1 \circ \mathbf{S}^2) \circ \mathbf{S}^3$ *is shown in Figure 2.7. Another example for a reversal schedule*

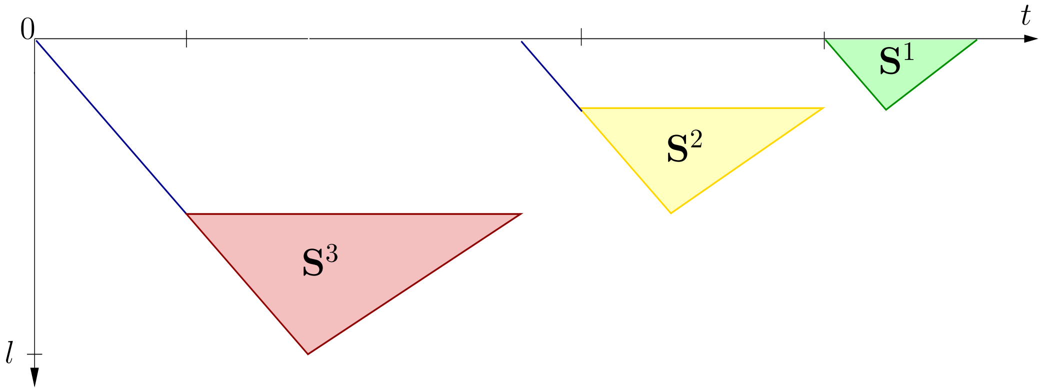

Figure 2.7: Decomposition of reversal schedule $\mathbf{S} = (\mathbf{S}^1 \circ \mathbf{S}^2) \circ \mathbf{S}^3$

$\mathbf{S}$ *given by the decomposition* $\mathbf{S} = \mathbf{S}^1 \circ (\mathbf{S}^2 \circ \mathbf{S}^3)$ *is shown in Figure 2.8. Comparing Figure 2.7 with Figure 2.8, we can see that the reversal schedules*

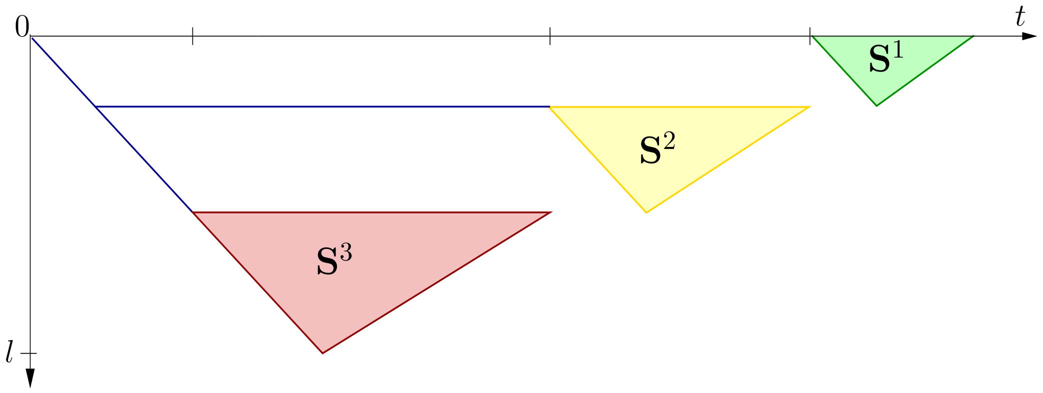

Figure 2.8: Decomposition of reversal schedule $\mathbf{S} = \mathbf{S}^1 \circ (\mathbf{S}^2 \circ \mathbf{S}^3)$

$\mathbf{S}$ *given by this figures are different, although the subschedules* $\mathbf{S}^1$*,* $\mathbf{S}^2$*, and*

$\mathbf{S}^3$, *used in these decompositions are assumed to be the same. Therefore, comparing Figure 2.7 with Figure 2.8 we can also see that the operator* $\circ$ *is not associative.*

□

Using the decomposition in the equation (2.18), the following relations can be observed

$$l_{max}(\mathbf{S}^1 \circ \mathbf{S}^2) = l_{max}(\mathbf{S}^1) + l_{max}(\mathbf{S}^2), \tag{2.23}$$

$$s_{max}(\mathbf{S}^1 \circ \mathbf{S}^2) = \max\left\{s_{max}(\mathbf{S}^1), s_{max}(\mathbf{S}^2) + 1\right\}, \tag{2.24}$$

$$r_{max}(\mathbf{S}^1 \circ \mathbf{S}^2) = \max\left\{r_{max}(\mathbf{S}^1) + 1, r_{max}(\mathbf{S}^2)\right\}, \tag{2.25}$$

$$m_j(\mathbf{S}) = m_{j-1}(\mathbf{S}^1) + m_j(\mathbf{S}^2). \tag{2.26}$$

Using the relations (2.23) - (2.26), one can evaluate explicit formulae for l_{max}, s_{max}, r_{max}, and m_j, which are the topics of the next sections.

2.1.7 Maximal Length $\mathbf{l_{max}(s, r)}$

We denote $l_{max}(s,r)$ the maximal number of time steps which can be reversed using s available checkpoints. Each time step F_i, $1 \le i \le l$, is evaluated up to r times during this reversal. $l_{max}(s,r)$ can be defined by

$$l_{max}(s,r) = \max\left\{l_{max}(\mathbf{S}) : \ s_{max}(\mathbf{S}) \le s, \ r_{max}(\mathbf{S}) \le r\right\}. \tag{2.27}$$

For the evaluation of the maximal length $l_{max}(s,r)$ for given values s and r we apply the decomposition of a reversal schedule $\mathbf{S}$ according to (2.18). Therefore, we obtain

$$\begin{aligned}
l_{max}(s,r) &= \max\Big\{l_{max}(\mathbf{S}^1 \circ \mathbf{S}^2) : \ s_{max}(\mathbf{S}^1) \le s, \ s_{max}(\mathbf{S}^2) \le s-1, \\
&\qquad\qquad r_{max}(\mathbf{S}^1) \le r-1, \ r_{max}(\mathbf{S}^2) \le r\Big\} = \\
&= \max\Big\{l_{max}(\mathbf{S}^1) + l_{max}(\mathbf{S}^2) : \ s_{max}(\mathbf{S}^1) \le s, \ s_{max}(\mathbf{S}^2) \le s-1, \\
&\qquad\qquad r_{max}(\mathbf{S}^1) \le r-1, \ r_{max}(\mathbf{S}^2) \le r\Big\} = \\
&= \max\Big\{l_{max}(\mathbf{S}^1) : \ s_{max}(\mathbf{S}^1) \le s, \ r_{max}(\mathbf{S}^1) \le r-1 \Big\} + \\
&+ \max\Big\{l_{max}(\mathbf{S}^2) : \ s_{max}(\mathbf{S}^2) \le s-1, \ r_{max}(\mathbf{S}^2) \le r \Big\} = \\
&= l_{max}(s, r-1) + l_{max}(s-1, r).
\end{aligned} \tag{2.28}$$

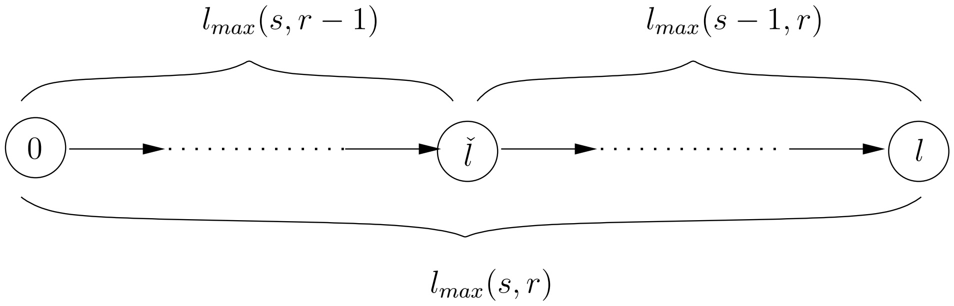

Figure 2.9: Decomposition of a step sequence according to (2.28)

The relation (2.28) is illustrated in Figure 2.9.

Values of $l_{max}(s, r)$ for $s = 1$ and $r = 1$ can be determined using Definition 2.2 of reversal schedules. Thus,

$$l_{max}(1, r) = r + 1, \qquad l_{max}(s, 1) = s + 1, \qquad s, r \geq 1. \tag{2.29}$$

The equations (2.28) and (2.29) yield the following result from [35]. Here and throughout the thesis, we use the abbreviation $\beta(s, r) = \frac{(s+r)!}{s!\,r!}$, which is also defined in the list of symbols.

Lemma 2.2 (Binomial Rule) *The maximal number of steps $l_{max}(s, r)$ can be evaluated explicitly by*

$$l_{max}(s, r) = \beta(s, r) = \binom{s+r}{s} = \frac{(s+r)!}{s!\,r!}. \tag{2.30}$$

Figure 2.10 illustrates level sets of the maximal length $l_{max}(r, s)$ as a function of r and s. Each curve in Figure 2.10 corresponds to a specified number $l_{max}(r, s)$. We also indicate the approximate functional dependence of l as a function of r and s when either of them is constant or both are proportional to each other. For a fixed repetition number r the number s of required checkpoints grows like the length l raised to the reciprocal of the repetition number r. For a fixed number s of available checkpoints the relative reversal cost r grows like the length l raised to the reciprocal of the number s of checkpoints available. In the case $r \sim s$ the relative reversal cost r as well as the number s of checkpoints available grow like the logarithm of the length l.

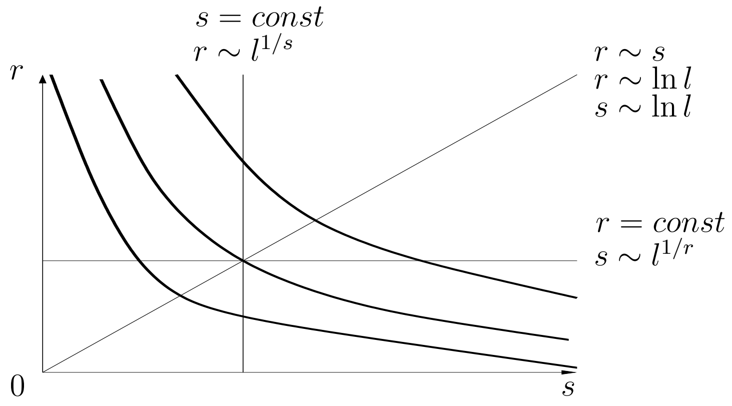

Figure 2.10: Level sets of maximal length $l_{max}(r, s)$

2.2 Optimal Reversal Schedules

Consider a forward simulation F which comprises l time steps. $\mathbf{t_l} = \langle t_1, ..., t_l \rangle^\top$ denotes a corresponding step cost distribution. To find an optimal admissible reversal schedule from the set

$$\big\{\mathbf{S}:\; l \le l_{max}(\mathbf{S}) \; and \; s_{max}(\mathbf{S}) \le s\big\} \tag{2.31}$$

of admissible reversal schedules for the forward simulation F and s available checkpoints, it is required to minimise the evaluation cost first. This is equivalent to determining

$$\mathbf{T}_{min}(l, s) = \min_{\mathbf{S}} \big\{\mathbf{T}(\mathbf{S}),\; l \le l_{max}(\mathbf{S}) \; and \; s_{max}(\mathbf{S}) \le s\big\}, \tag{2.32}$$

in uniform case and

$$\mathbf{T}_{min}(\mathbf{t_l}, s) = \min_{\mathbf{S}} \big\{\mathbf{T}(\mathbf{t_l}, \mathbf{S}),\; l \le l_{max}(\mathbf{S}) \; and \; s_{max}(\mathbf{S}) \le s\big\}, \tag{2.33}$$

in non-uniform case. In order to explain the procedure for the construction of optimal reversal schedules as well as to derive explicit formulae for the minimal evaluation cost $\mathbf{T}_{min}(l, s)$ and $\mathbf{T}_{min}(\mathbf{t_l}, s)$, uniform and non-uniform cases are considered separately.

2.2.1 Optimal Reversal Schedule for Uniform Case

Obviously, optimal reversal schedules feature checkpoint persistency (Lemma 2.1) as well as the decomposition of reversal schedules according to (2.18). Therefore, it is possible to prove the following assertion:

Theorem 2.1 (Minimal Evaluation Cost) *Consider a forward simulation F which comprises l time steps. Let s checkpoints be available, step cost distribution is uniform. Then, the function $\mathbf{T}_{min}(l,s)$, which denotes the minimal evaluation cost to reverse l time steps with up to s checkpoints accommodated at any time, yields the form*

$$\mathbf{T}_{min}(l,s) = \min_{0<\check{l}<l}\{\check{l} + \mathbf{T}_{min}(l-\check{l}, s-1) + \mathbf{T}_{min}(\check{l}, s)\}. \tag{2.34}$$

This implies

$$\mathbf{T}_{min}(l,s) = rl - \beta(s+1, r-1), \tag{2.35}$$

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

$$\beta(s, r-1) \;\; < \;\; l \;\; \leq \;\; \beta(s,r). \tag{2.36}$$

Proof: For the proof of this theorem see [35, 83, 91].

Theorem 2.1 implies the following corollary, which gives a new representation of the minimal evaluation cost $\mathbf{T}_{min}(l,s)$ and a proof for its explicit formula.

Corollary 2.1 *Let the assumptions of Theorem 2.1 be satisfied. Then, the function $\mathbf{T}_{min}(l,s)$ can be evaluated by*

$$\mathbf{T}_{min}(l,s) = \sum_{i=1}^{l} r(s,i), \tag{2.37}$$

with $r(s,i) = r_{bin}(s,i),\ 1 \leq i \leq l$, being the unique integers satisfying

$$\beta(s, r(s,i)-1) \;\; < \;\; i \;\; \leq \;\; \beta(s, r(s,i)). \tag{2.38}$$

Proof: The relation (2.37) is proved by induction.

For $s = 1$ and $l \in \mathbb{N}$, we obtain

$$\beta(1, r-1) = r < l \leq r+1 = \beta(1,r) \qquad \Longleftrightarrow \qquad r = l-1.$$

The single checkpoint must be set to the initial intermediate state. For the first reverse step a minimum of r time steps is needed. To perform the next reverse step, $r-1$ time steps are necessary, and so on. Therefore, the minimal number of time steps is equal to

$$\mathbf{T}_{min}(l,1) = r + (r-1) + \cdots + 1 + 0 = \sum_{i=1}^{l}(i-1) = \sum_{i=1}^{l} r(1,i),$$

since $r(1,i) = i-1$ for $1 \leq i \leq l$.

For $s \in \mathbb{N}$ and $1 < l \leq s+1$, we obtain

$$\beta(s,0) = 1 < l \leq s+1 = \beta(s,1) \qquad \Longleftrightarrow \qquad r=1.$$

The s checkpoints must be set to all intermediate states except for the last one. For the first reverse step a minimum of $l-1$ time steps is needed. To perform other reverse steps, no time steps are necessary. Therefore, the minimal number of time steps is equal to

$$\mathbf{T}_{min}(l,s) = l-1 = \underbrace{0+1+1+...+1}_{l} = \sum_{i=1}^{l} r(s,i),$$

since $r(s,1) = 0$ and $r(s,i) = 1$ for $2 \leq i \leq l \leq s+1$.

Assume that the assertion of Corollary 2.1 is true for a fixed s and all $\tilde{l}$ with $\tilde{l} \leq l$. Now, it will be shown that equality (2.37) is valid for $l+1$, i.e.

$$\mathbf{T}_{min}(l+1,s) = \sum_{i=1}^{l+1} r(s,i). \tag{2.39}$$

Obviously, $\mathbf{T}_{min}(l+1,s) - \mathbf{T}_{min}(l,s) = \sum_{i=1}^{l+1} r(s,i) - \sum_{i=1}^{l} r(s,i) = r(s,l+1)$.

Therefore, in order to prove (2.39) is sufficient, it is to be shown that $\mathbf{T}_{min}(l+1,s) - \mathbf{T}_{min}(l,s) = r(s,l+1)$, with $\mathbf{T}_{min}(l+1,s)$ and $\mathbf{T}_{min}(l,s)$ evaluated using (2.35). Thus, applying (2.35), we obtain

$$\begin{aligned}&\mathbf{T}_{min}(l+1,s) - \mathbf{T}_{min}(l,s) = r(s,l+1)(l+1) - \\ &-\beta(s+1, r(s,l+1)-1) - r(s,l)l + \beta(s+1, r(s,l)-1).\end{aligned}$$

Here, consider two different cases:

1st case: $r(s,l+1) = r(s,l)$. Therefore,

$$\begin{aligned}&\mathbf{T}_{min}(l+1,s) - \mathbf{T}_{min}(l,s) = r(s,l+1)(l+1) - \\ &-\beta(s+1, r(s,l+1)-1) - r(s,l)l + \beta(s+1, r(s,l)-1) = \\ &= r(s,l)(l+1) - r(s,l)l = r(s,l) = r(s,l+1),\end{aligned}$$

which proves the assertion of Corollary 2.1 for this case.

2nd case: $r(s,l+1) = r(s,l)+1$. Note that if $r(s,l+1) = r(s,l)+1$, (2.36) implies that $l = \beta(s, r(s,l))$. Therefore,

$$\begin{aligned}
&\mathbf{T}_{min}(l+1,s) - \mathbf{T}_{min}(l,s) = r(s,l+1)(l+1) - \\
&-\beta(s+1, r(s,l+1)-1) - r(s,l)l + \beta(s+1, r(s,l)-1) = \\
&= (r(s,l)+1)(l+1) - r(s,l)l - \beta(s+1, r(s,l)) + \beta(s+1, r(s,l)-1) = \\
&= (l+1) + r(s,l) - \beta(s, r(s,l)) = (l+1) + r(s,l) - l = \\
&= r(s,l)+1 = r(s,l+1).
\end{aligned}$$

This proves the assertion of Corollary 2.1 for the second case. Here, we use the relation

$$\beta(s+1, r(s,l)) - \beta(s+1, r(s,l)-1) = \beta(s, r(s,l)).$$

This is straightforwardly proved applying the definition of the function β in (2.30).

□

Alternative proof: Alternatively to the proof of Corollary 2.1 given above, where the equivalence between the equation (2.35) and the equation (2.37) is shown, we can straightforwardly prove Corollary 2.1 in the following way. The minimal evaluation cost $\mathbf{T}_{min}(l,s)$, defined by the formula (2.34), can be evaluated explicitly using (2.37). Here, we show the equivalence between the equation (2.34) and the equation (2.37).

This equivalence is also proved by induction.

For the both trivial cases, namely for $s = 1$ and $l \in \mathbb{N}$; and for $s \in \mathbb{N}$ and $1 < l \leq s+1$, we obtain

$$\mathbf{T}_{min}(l,s) = \sum_{i=1}^{l} r(s,i).$$

In order to show this formula, we might use the same considerations as above.

Assume that the assertion of Corollary 2.1 is true for a fixed s and all $\tilde{l}$ with $\tilde{l} \leq l$. Now, it will be shown that equality (2.37) is valid for $l+1$, i.e.

$$\mathbf{T}_{min}(l+1,s) = \sum_{i=1}^{l+1} r(s,i). \tag{2.40}$$

Obviously, $\mathbf{T}_{min}(l+1,s) - \mathbf{T}_{min}(l,s) = \sum_{i=1}^{l+1} r(s,i) - \sum_{i=1}^{l} r(s,i) = r(s,l+1)$.

Therefore, in order to prove (2.37) is sufficient, it is to be shown that $\mathbf{T}_{min}(l+1,s) - \mathbf{T}_{min}(l,s) = r(s,l+1)$, with $\mathbf{T}_{min}(l+1,s)$ and $\mathbf{T}_{min}(l,s)$ being evaluated using (2.34). This can be also proved by induction.

For $s = 1$ and $\tilde{l} \in \mathbb{N}$, as well as for $s \in \mathbb{N}$ and $1 < \tilde{l}+1 \leq s+1$, we obviously have

$$\mathbf{T}_{min}(\tilde{l}+1,s) - \mathbf{T}_{min}(\tilde{l},s) = r(s,\tilde{l}+1), \tag{2.41}$$

with $\mathbf{T}_{min}(\tilde{l}+1,s)$ and $\mathbf{T}_{min}(\tilde{l},s)$ being evaluated using (2.34). This is because for these cases the equivalence between (2.34) and (2.37) is already proved.

Assume that the equation (2.41) is true for a fixed s and all $\tilde{l}$ with $\tilde{l} < l$. Now, it will be shown that equality (2.41) is valid for $\tilde{l} = l$, i.e.

$$\mathbf{T}_{min}(l+1,s) - \mathbf{T}_{min}(l,s) = r(s,l+1), \tag{2.42}$$

with $\mathbf{T}_{min}(l+1,s)$ and $\mathbf{T}_{min}(l,s)$ being evaluated using (2.34). Assume that an optimal value $\check{l}^\star$ is determined, so that

$$\mathbf{T}_{min}(l,s) = \check{l}^\star + \mathbf{T}_{min}(l-\check{l}^\star, s-1) + \mathbf{T}_{min}(\check{l}^\star, s). \tag{2.43}$$

Then, due to Lemma 2.3 to be proved later on, an optimal value $\check{m}^\star$ for $m = l+1$ has two possibilities to choose from, namely $\check{m}^\star = \check{l}^\star$ or $\check{m}^\star = \check{l}^\star+1$. Then, the optimal evaluation cost $\mathbf{T}_{min}(l+1,s)$ is evaluated by

$$\mathbf{T}_{min}(l+1,s) = \check{m}^\star + \mathbf{T}_{min}(l+1-\check{m}^\star, s-1) + \mathbf{T}_{min}(\check{m}^\star, s), \tag{2.44}$$

with an appropriate value $\check{m}^\star$. Therefore, in order to prove (2.42), we have to find the difference between the equations (2.44) and (2.43) for two cases of $\check{m}^\star$.

1st case: $\check{m}^\star = \check{l}^\star$. Therefore,

$$\begin{gathered}\mathbf{T}_{min}(l+1,s) - \mathbf{T}_{min}(l,s) = \check{l}^\star + \mathbf{T}_{min}(l+1-\check{l}^\star, s-1) + \\ + \mathbf{T}_{min}(\check{l}^\star, s) - \check{l}^\star - \mathbf{T}_{min}(l-\check{l}^\star, s-1) - \mathbf{T}_{min}(\check{l}^\star, s) = \\ = \mathbf{T}_{min}(l+1-\check{l}^\star, s-1) - \mathbf{T}_{min}(l-\check{l}^\star, s-1) = \\ = r(s-1, l+1-\check{l}^\star) = r(s,l+1).\end{gathered} \tag{2.45}$$

This is due to the assumption above. Since $l - \check{l}^\star < l$, we obtain $\mathbf{T}_{min}(l+1-\check{l}^\star, s-1) - \mathbf{T}_{min}(l-\check{l}^\star, s-1) = r(s-1, l+1-\check{l}^\star)$. The last equality in (2.45) is because of Lemma 2.4 to be proved later on.

2nd case: $\check{m}^\star = \check{l}^\star + 1$. Therefore,

$$
\begin{aligned}
\mathbf{T}_{min}(l+1,s) - \mathbf{T}_{min}(l,s) &= \check{l}^\star + 1 + \mathbf{T}_{min}(l-\check{l}^\star, s-1) + \\
+\mathbf{T}_{min}(\check{l}^\star+1,s) - \check{l}^\star - \mathbf{T}_{min}(l-\check{l}^\star, s-1) - \mathbf{T}_{min}(\check{l}^\star, s) &= \\
= 1 + \mathbf{T}_{min}(\check{l}^\star+1,s) - \mathbf{T}_{min}(\check{l}^\star, s) &= \\
= r(s, \check{l}^\star + 1) + 1 = r(s, l+1).
\end{aligned}
\tag{2.46}
$$

This is due to the assumption above. Since $\check{l}^\star < l$, we obtain $\mathbf{T}_{min}(\check{l}^\star+1,s) - \mathbf{T}_{min}(\check{l}^\star,s) = r(s,\check{l}^\star+1)$. The last equality in (2.46) is because of Lemma 2.4 to be proved later on.

This proves the equation (2.42). Therefore, the equivalence between the equation (2.34) and the equation (2.37) is shown, which immediately yields the statement of Corollary 2.1.

□

In the following, we prove two lemmas, those results are used throughout the alternative proof of Corollary 2.1.

Lemma 2.3 *Assume that the following equation is true for a fixed s and all $\tilde{l}$ with $\tilde{l} < l$:*

$$
\mathbf{T}_{min}(\tilde{l}+1,s) - \mathbf{T}_{min}(\tilde{l},s) = r(s,\tilde{l}+1), \tag{2.47}
$$

with $\mathbf{T}_{min}(\tilde{l}+1,s)$ and $\mathbf{T}_{min}(\tilde{l},s)$ being evaluated using (2.34). Moreover, assume that an optimal value $\check{l}^\star$ is determined, so that

$$
\mathbf{T}_{min}(l,s) = \check{l}^\star + \mathbf{T}_{min}(l-\check{l}^\star, s-1) + \mathbf{T}_{min}(\check{l}^\star, s). \tag{2.48}
$$

Then, an optimal value $\check{m}^\star$ for $m = l+1$ has two possibilities to choose from, namely $\check{m}^\star = \check{l}^\star$ or $\check{m}^\star = \check{l}^\star + 1$.

Proof:

Since, an optimal value $\check{l}^\star$ is determined, so that (2.48) satisfied, then, the following two relations are valid.

$$
\check{l}^\star + \mathbf{T}_{min}(l-\check{l}^\star, s-1) + \mathbf{T}_{min}(\check{l}^\star, s) \le \check{l}^\star - 1 + \mathbf{T}_{min}(l-\check{l}^\star+1, s-1) + \mathbf{T}_{min}(\check{l}^\star-1, s), \tag{2.49}
$$

for $\check{l}^\star > 1$ and

$$
\check{l}^\star + \mathbf{T}_{min}(l-\check{l}^\star, s-1) + \mathbf{T}_{min}(\check{l}^\star, s) \le \check{l}^\star + 1 + \mathbf{T}_{min}(l-\check{l}^\star-1, s-1) + \mathbf{T}_{min}(\check{l}^\star+1, s), \tag{2.50}
$$

for $\check{l}^\star < l-1$. Since the function $\mathbf{T}_{min}(l,s)$ is convex, the relations (2.49) and (2.50) are equivalent to $\check{l}^\star$ being its global minimum defined by (2.34). Because of the assumption (2.47), the equations (2.49) and (2.50) are equivalent to

$$r(s,\check{l}^\star)+1 \le r(s-1,l-\check{l}^\star+1), \tag{2.51}$$

for $\check{l}^\star > 1$ and

$$r(s,\check{l}^\star+1)+1 \ge r(s-1,l-\check{l}^\star), \tag{2.52}$$

for $\check{l}^\star < l-1$, respectively. The relations (2.51) and (2.52) are satisfied if $\check{l}^\star$ is an optimal value satisfying (2.48).

In order to prove the assertion of Lemma 2.3, i.e. to prove that an optimal value $\check{m}^\star$ for $m = l+1$ has two possibilities to choose from, namely $\check{m}^\star = \check{l}^\star$ or $\check{m}^\star = \check{l}^\star + 1$, we have to show the following two relations are satisfied.

$$\check{l}^\star+\mathbf{T}_{min}(l+1-\check{l}^\star,s-1)+\mathbf{T}_{min}(\check{l}^\star,s) \le \check{l}^\star-1+\mathbf{T}_{min}(l-\check{l}^\star+2,s-1)+\mathbf{T}_{min}(\check{l}^\star-1,s), \tag{2.53}$$

for $\check{l}^\star > 1$ and

$$\check{l}^\star+1+\mathbf{T}_{min}(l-\check{l}^\star,s-1)+\mathbf{T}_{min}(\check{l}^\star+1,s) \le \check{l}^\star+2+\mathbf{T}_{min}(l-\check{l}^\star-1,s-1)+\mathbf{T}_{min}(\check{l}^\star+2,s), \tag{2.54}$$

for $\check{l}^\star < l-1$. (2.53) and (2.54) identify $\check{l}^\star$ or $\check{l}^\star+1$ as a global minimum defined by (2.34) for $l+1$ time steps. Because of the assumption (2.47), the equations (2.53) and (2.54) are equivalent to the following two relations

$$r(s,\check{l}^\star)+1 \le r(s-1,l-\check{l}^\star+2), \tag{2.55}$$

for $\check{l}^\star > 1$ and

$$r(s,\check{l}^\star+2)+1 \ge r(s-1,l-\check{l}^\star), \tag{2.56}$$

for $\check{l}^\star < l-1$, respectively. Recalling (2.51) and using the fact that the function $r(s,l)$ is monotone increasing in l, we obtain that $r(s-1,l-\check{l}^\star+1) \le r(s-1,l-\check{l}^\star+2)$. Therefore, the inequality (2.55) is true.

Moreover, with the observations above we obtain $r(s,\check{l}^\star+2) \ge r(s,\check{l}^\star+1)$. Consequently, the inequality (2.56) is also satisfied due to (2.52). Hence, the assertion of Lemma 2.3 is proved.

□

Lemma 2.4 *Consider a forward simulation F which comprises $m = l + 1$ time steps. Let s checkpoints be available, step cost distribution is uniform. Moreover, consider an optimal reversal schedule $\mathbf{S}(m, s)$, so that*

$$\mathbf{T}(\mathbf{S}(m,s)) = \mathbf{T}_{min}(m,s) = \min_{0<\check{m}<m}\{\check{m} + \mathbf{T}_{min}(m-\check{m}, s-1) + \mathbf{T}_{min}(\check{m}, s)\}. \tag{2.57}$$

Assume that the following equation is true for a fixed s and all $\tilde{l}$ with $\tilde{l} < l$ and $l \geq s-1$.

$$\mathbf{T}_{min}(\tilde{l}+1, s) - \mathbf{T}_{min}(\tilde{l}, s) = r(s, \tilde{l}+1), \tag{2.58}$$

with $\mathbf{T}_{min}(\tilde{l}+1, s)$ and $\mathbf{T}_{min}(\tilde{l}, s)$ evaluated using (2.34). Furthermore, an optimal value $\check{m}^\star$ is determined, so that

$$\mathbf{T}_{min}(m,s) = \check{m}^\star + \mathbf{T}_{min}(m-\check{m}^\star, s-1) + \mathbf{T}_{min}(\check{m}^\star, s). \tag{2.59}$$

Then, for the maximal repetition numbers $r_{max}(\mathbf{S}(m,s))$, $r_{max}(\mathbf{S}(\check{m}^\star, s))$, and $r_{max}(\mathbf{S}(m-\check{m}^\star, s-1))$ for the reversal schedules $\mathbf{S}(m,s)$, $\mathbf{S}(\check{m}^\star, s)$, and $\mathbf{S}(m-\check{m}^\star, s-1)$, respectively, so that $\mathbf{S}(m,s) = \mathbf{S}(\check{m}^\star, s) \circ \mathbf{S}(m-\check{m}^\star, s-1)$, the following relations are satisfied

$$r_{max}(\mathbf{S}(m,s)) = r(s,m) = r(s,\check{m}^\star) + 1 = r(s-1, m-\check{m}^\star). \tag{2.60}$$

Proof:

Suppose, relations (2.60) are not satisfied, i.e. $r(s,\check{m}^\star)+1 > r(s-1, m-\check{m}^\star)$ or $r(s,\check{m}^\star)+1 < r(s-1, m-\check{m}^\star)$. In the following, consider each case separately.

1st case: Suppose, $r(s,\check{m}^\star)+1 > r(s-1, m-\check{m}^\star)$. If $\check{m}^\star = 1$, then $1 > r(s-1, m-1) \geq 1$, which follows immediately the contradiction. Thus, we exclude the case $\check{m}^\star = 1$ in the following. Therefore, this is equivalent to

$$r(s,\check{m}^\star) + 1 \geq r(s-1, m-\check{m}^\star) + 1 \geq r(s-1, m+1-\check{m}^\star). \tag{2.61}$$

Since an optimal value $\check{m}^\star$ is determined, so that (2.59) is satisfied, then, the following relation is valid.

$$\begin{aligned}\check{m}^\star + \mathbf{T}_{min}(m-\check{m}^\star, s-1) + \mathbf{T}_{min}(\check{m}^\star, s) \leq \check{m}^\star - 1 \;&+\; \mathbf{T}_{min}(m+1-\check{m}^\star, s-1) + \\ &+\; \mathbf{T}_{min}(\check{m}^\star - 1, s).\end{aligned} \tag{2.62}$$

Because of the assumption (2.58) and $\check{m}^\star > 1$, the equation (2.62) is equivalent to

$$r(s,\check{m}^\star) + 1 \leq r(s-1, m+1-\check{m}^\star). \tag{2.63}$$

If the last inequality in (2.61) and the relation (2.63) are satisfied strictly, then, they contradict each over. Therefore, the assumption $r(s, \check{m}^\star) + 1 > r(s-1, m - \check{m}^\star)$ is not true. In the other case, i.e. if $r(s, \check{m}^\star) + 1 = r(s-1, m+1-\check{m}^\star)$, we can move the optimal value $\check{m}^\star$ towards the left by one without increasing the resulting evaluation cost $\mathbf{T}(\mathbf{S}(m,s)) = \mathbf{T}_{min}(m,s)$ due to (2.62). Applying the considerations above sufficiently many times, we achieve that

$$r(s, \check{m}^\star) + 1 = r(s-1, m - \check{m}^\star)$$

is true.

2nd case: Suppose, $r(s, \check{m}^\star) + 1 < r(s-1, m - \check{m}^\star)$. If $\check{m}^\star = m-1$, then $1 > 1 + r(s, m-1) \geq 1$, which follows immediately the contradiction. Thus, we exclude the case $\check{m}^\star = m-1$ in the following. Therefore, this is equivalent to

$$r(s, \check{m}^\star) + 2 \leq r(s-1, m - \check{m}^\star). \tag{2.64}$$

Since, an optimal value $\check{m}^\star$ is determined, so that (2.59) is satisfied, then, the following relation is valid.

$$\begin{aligned} \check{m}^\star + \mathbf{T}_{min}(m - \check{m}^\star, s-1) + \mathbf{T}_{min}(\check{m}^\star, s) \leq \check{m}^\star + 1 \quad &+ \quad \mathbf{T}_{min}(m - \check{m}^\star - 1, s-1) \\ &+ \quad \mathbf{T}_{min}(\check{m}^\star + 1, s). \end{aligned} \tag{2.65}$$

Because of the assumption (2.58) and $\check{m}^\star < m-1$, the equation (2.65) is equivalent to

$$r(s, \check{m}^\star + 1) + 1 \geq r(s-1, m - \check{m}^\star). \tag{2.66}$$

Since $1 + r(s, \check{m}^\star) \geq r(s, \check{m}^\star + 1)$, we obtain from (2.66)

$$r(s, \check{m}^\star) + 2 \geq r(s-1, m - \check{m}^\star). \tag{2.67}$$

If the inequalities (2.64) and (2.67) are satisfied strictly, then, they contradict each over. Therefore, the assumption $r(s, \check{m}^\star) + 1 < r(s-1, m - \check{m}^\star)$ is not true. In the other case, i.e. if $r(s, \check{m}^\star) + 2 = r(s-1, m - \check{m}^\star)$, we can move the optimal value $\check{m}^\star$ towards the right by one without increasing the resulting evaluation cost $\mathbf{T}(\mathbf{S}(m,s)) = \mathbf{T}_{min}(m,s)$ due to (2.65). Applying the considerations above sufficiently many times, we achieve that

$$r(s, \check{m}^\star) + 1 = r(s-1, m - \check{m}^\star)$$

is true.

Furthermore, we have to prove at least one of the relations

$$r(s,m) = r(s, \check{m}^\star) + 1, \qquad \text{or} \qquad r(s,m) = r(s-1, m - \check{m}^\star). \tag{2.68}$$

They follow immediately from the assumption that the reversal schedule $\mathbf{S}(m,s)$ is an optimal one and the decomposition $\mathbf{S}(m,s) = \mathbf{S}(\check{m}^\star, s) \circ \mathbf{S}(m - \check{m}^\star, s-1)$ is valid. Therefore, due to (2.25), we obtain

$$\begin{aligned} r(s,m) = r_{max}(\mathbf{S}(m,s)) = r_{max}(\mathbf{S}(\check{m}^\star, s) \circ \mathbf{S}(m - \check{m}^\star, s-1)) = \\ \max\left\{r_{max}(\mathbf{S}(\check{m}^\star, s)) + 1, r_{max}(\mathbf{S}(m - \check{m}^\star, s-1))\right\} = \\ \max\left\{r(s,\check{m}^\star) + 1, r(s-1, m-\check{m}^\star)\right\} = \\ = r(s,\check{m}^\star) + 1 = r(s-1, m - \check{m}^\star). \end{aligned} \tag{2.69}$$

Hence, the assertion of Lemma 2.4 is proved.

□

Corollary 2.1 implies the following conclusion. If a sequence of l time steps is reversed, using a minimal number $\mathbf{T}_{min}(l,s)$ of forward steps and s checkpoints, each time step F_i, $1 \leq i \leq l$, can be associated with a unique number k_i, $1 \leq k_i \leq l$. The repetition number $r(s,k_i) = r_{bin}(s,k_i)$ counts how often the time step F_i is evaluated during the reversal of the step sequence. These relations are illustrated in Figure 2.11. Obviously, it can occur that

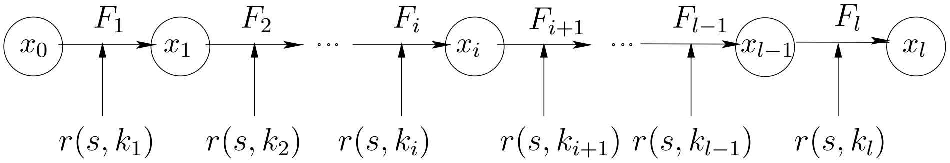

Figure 2.11: Correspondence between time steps and repetition numbers

there are several different steps F_i and consequently different numbers k_i with $r(s,k_i) = j$ for a specified j, $1 \leq j \leq r_{bin}(s,l)$. The overall number of these steps corresponds to the frequency number $m_j(\mathbf{S})$ for a reversal schedule $\mathbf{S}$ with $\mathbf{T}(\mathbf{S}) = \mathbf{T}_{min}(l,s)$.

The dependence of the minimal evaluation cost $\mathbf{T}_{min}(l,s)$ on l is shown in Figure 2.12 for various numbers s of checkpoints. $\mathbf{T}_{min}(l,s)$ being a function of l produces a piecewise linear graph.

In uniform case the value $\mathbf{T}_{min}(l,s)$ of the minimal evaluation cost can be achieved by many reversal schedules. Therefore, it is convenient to introduce a second quantity $\mathbf{q}(\mathbf{S})$, defined by

$\mathbf{q}(\mathbf{S})$ = number of checkpoints writes W_k, which are required during the execution of a reversal schedule $\mathbf{S}$.

Using the decomposition of reversal schedules according to (2.18), the value $\mathbf{q}(\mathbf{S})$ of checkpoints writes can be evaluated as follows

$$\mathbf{q}(\mathbf{S}) = \mathbf{q}(\mathbf{S}^1 \circ \mathbf{S}^2) = \mathbf{q}(\mathbf{S}^1) + \mathbf{q}(\mathbf{S}^2) + 1. \tag{2.70}$$

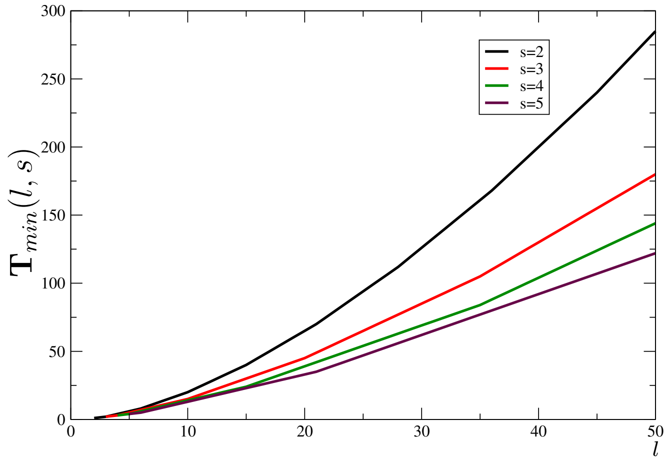

Figure 2.12: Minimal evaluation cost $\mathbf{T}_{min}(l,s)$ being a function depending on l and s

For the example of a reversal schedule $\mathbf{S}$ in Figure 2.3 the value $\mathbf{q}(\mathbf{S})$ of checkpoints writes reaches 5.

In order to find an optimal reversal schedule out of all admissible ones which evaluation cost $\mathbf{T}(\mathbf{S})$ achieves the minimal value $\mathbf{T}_{min}(l,s)$, we choose reversal schedules which values $\mathbf{q}(\mathbf{S})$ minimise the second objective $\mathbf{q}$, i.e.

(2.71)

$$\mathbf{q}_{min}(l,s) = \min_{\mathbf{S}} \left\{ \mathbf{q}(\mathbf{S}) : \; l \leq l_{max}(\mathbf{S}) \; , \; s_{max}(\mathbf{S}) \leq s \; , \; \mathbf{T}(\mathbf{S}) = \mathbf{T}_{min}(l,s) \right\}.$$

Note that the number of read actions R_k can not be minimised since this number is always l due to the checkpoint persistency.

The values $\mathbf{T}_{min}(l,s)$ and $\mathbf{q}_{min}(l,s)$ can be evaluated explicitly. Simultaneously, an optimal reversal schedule can be constructed. Explicit formula for $\mathbf{q}_{min}(l,s)$ is given by the next theorem.

Theorem 2.2 (Minimisations of Checkpoints Writes) *Provided the assumptions of Theorem 2.1 are satisfied, the minimal number $\mathbf{q}_{min}(l,s)$ of checkpoints writes takes the following form*

$$\mathbf{q}_{min}(l,s) = \begin{cases} \beta(s-1,r-1)-1, & if \quad l \leq \beta(s,r-1)+\beta(s-1,r-1), \\ l-\beta(s,r-1)-1, & if \quad l \geq \beta(s,r-1)+\beta(s-1,r-1). \end{cases}$$

$r = r_{bin}(s,l)$ is to be determined according to (2.36).

Proof: For the proof of this Theorem see Proposition 2 in [38].

Now, it is to clarify, how an optimal reversal schedule $\mathbf{S}$ for the given sequence of l time steps and s available checkpoints is to construct. At first, the sum over all evaluated steps and then, the number of all checkpoints writes of this reversal schedule achieve the minimal values $\mathbf{T}_{min}(l,s)$ and $\mathbf{q}_{min}(l,s)$. Obviously, this can be attained by a suitable distribution of checkpoints. The following theorem gives an explicit formula for the construction of an optimal checkpoint distribution.

Theorem 2.3 (Optimal Reversal Schedule) *In order to construct an optimal reversal schedule* $\mathbf{S}$ *for a sequence of* l *time steps with uniform step cost distribution and* s *available checkpoints, and consequently, first to minimise the value* $\mathbf{T}(\mathbf{S})$ *of evaluation cost and then, the value* $\mathbf{q}(\mathbf{S})$ *of checkpoints writes for this reversal schedule, the step sequence* $F_1, ..., F_l$ *has to be decomposed according to (2.18) with* $\check{l}$ *defined by*

$$\check{l} = \begin{cases} \beta(s, r-2), & if\ l \leq \beta(s, r-1) + \beta(s-2, r-1), \\ \beta(s, r-1), & if\ l \geq \beta(s, r) - \beta(s-3, r), \\ l - \beta(s-1, r-1) - \beta(s-2, r-1), & else. \end{cases}$$

$r = r_{bin}(s, l)$ *is to be determined according to (2.36).*

Proof: For the proof of this Theorem see the conclusion from Proposition 2 in [38].

The following notations are introduced for optimal reversal schedules constructed according to Theorem 2.3:

> $\mathbf{S}_{bin}(l,s)$ is an optimal reversal schedule for l time steps and s available checkpoints, provided a uniform step cost distribution is fulfilled,
>
> $\mathbf{T}_{bin}(l,s)$ is minimal evaluation cost required to reverse l time steps with up to s available checkpoints, provided a uniform step cost distribution is fulfilled.

Obviously, due to the construction of binomial reversal schedules $\mathbf{S}_{bin}(l,s)$, the following relations are satisfied.

(2.72)

$$\mathbf{T}(\mathbf{S}_{bin}(l,s)) = \mathbf{T}_{min}(l,s) = \mathbf{T}_{bin}(l,s), \qquad \mathbf{q}(\mathbf{S}_{bin}(l,s)) = \mathbf{q}_{min}(l,s).$$

Therefore, each binomial reversal schedule $\mathbf{S}_{bin}(l,s)$ is decomposed into two subschedules: the binomial schedule $\mathbf{S}_{bin}(\check{l},s)$ for a sequence of $\check{l}$ time steps

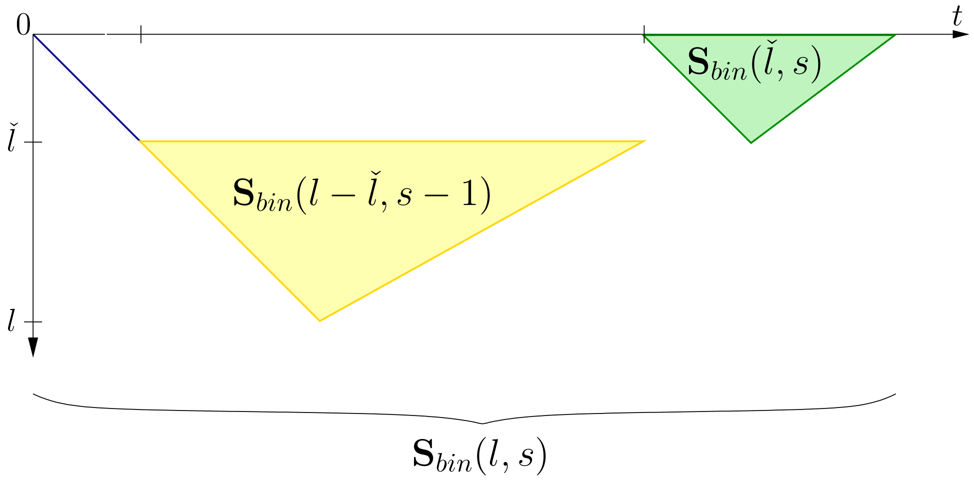

Figure 2.13: Decomposition of reversal schedule $\mathbf{S}_{bin}(l, s) = \mathbf{S}_{bin}(\check{l}, s) \circ \mathbf{S}_{bin}(l - \check{l}, s-1)$

$F_1, ..., F_{\check{l}}$ and the binomial schedule $\mathbf{S}_{bin}(l-\check{l}, s-1)$ for a sequence of $(l-\check{l})$ time steps $F_{\check{l}+1}, ..., F_l$. This situation is illustrated in Figure 2.13. Consider triangles denoting corresponding subschedules $\mathbf{S}_{bin}(\check{l}, s)$ and $\mathbf{S}_{bin}(l-\check{l}, s-1)$. Each of these triangles consists of a single horizontal line and two slanted lines. Considering slopes of slanted lines more detailed, we observe that the slope of the left slanted line of both triangles is 1. These lines represent the evaluation of time steps $F_1, ..., F_{\check{l}}$ and $F_{\check{l}+1}, ..., F_l$, respectively, only once. Due to the assumption $w = 1$, the slopes of these lines are also exactly 1. Now, consider slopes of the right slanted lines of both triangles representing $\mathbf{S}_{bin}(\check{l}, s)$ and $\mathbf{S}_{bin}(l-\check{l}, s-1)$. These lines represent the evaluation of the adjoint steps $\bar{F}_1, ..., \bar{F}_{\check{l}}$ and $\bar{F}_{\check{l}+1}, ..., \bar{F}_l$, respectively. Because of the optimality of the reversal schedule $\mathbf{S}_{bin}(l, s)$, it is necessary to evaluate each time step F_i, $1 \leq i \leq l$, at most $r_{bin}(s, l)$ times. Moreover, the assertion of Lemma 2.4 is satisfied for the binomial reversal schedule $\mathbf{S}_{bin}(l, s)$, i.e.

$$r_{bin}(s, l) = r_{bin}(s, \check{l}) + 1 = r_{bin}(s-1, l-\check{l}). \tag{2.73}$$

Therefore, obviously

$$\begin{array}{rcll} \mathbf{T}_{bin}(l, s) & \leq & l\, r_{bin}(s, l), & \\ \mathbf{T}_{bin}(\check{l}, s) & \leq & \check{l}\, r_{bin}(s, \check{l}) & = \ \check{l}\, (r_{bin}(s, l) - 1), \\ \mathbf{T}_{bin}(l-\check{l}, s-1) & \leq & (l-\check{l})\, r_{bin}(s-1, l-\check{l}) & = \ (l-\check{l})\, r_{bin}(s, l). \end{array} \tag{2.74}$$

Summarising, we obtain that the slope of the right slanted line for the triangle representing $\mathbf{S}_{bin}(\check{l}, s)$ is estimated above by $1/(r_{bin}(s, l) - 1)$. Moreover, the slope of the right slanted line for the triangle representing $\mathbf{S}_{bin}(l-\check{l}, s-1)$ is estimated above by $1/r_{bin}(s,l)$.

For further decompositions of binomial reversal schedules $\mathbf{S}_{bin}(\check{l}, s)$ and $\mathbf{S}_{bin}(l-\check{l}, s-1)$, Theorem 2.3 has to be applied again. This time l denotes the number of time steps of an appropriate subsequence. As soon as all s available checkpoints are occupied, i.e. the number of stored intermediate states is equal to the number s of initially available checkpoints, actions A and U (according to Definition 2.2) have to be executed until the length l of the step sequence is reduced to the number of the last stored intermediate state or to 0. An example for the binomial reversal schedule $\mathbf{S}_{bin}(10,3)$ for $l=10$ and $s=3$ is illustrated in Figure 2.3.

2.2.2 Optimal Reversal Schedules for Non-uniform Case

Consider a forward simulation F consisting of l time steps F_i, $1 \le i \le l$, with non-uniform step cost distribution $\mathbf{t_l}$. Optimal reversal schedules for non-uniform forward simulations will be established in this section. Obviously, binomial reversal schedules $\mathbf{S}_{bin}(l,s)$ are not optimal in this case. Therefore, it is necessary to establish a new method.

This problem can be solved using Dynamic Programming Approach. Runtime, needed to construct an optimal reversal schedule for a non-uniform forward simulation, is of order $\mathcal{O}(sl^3)$ (see [91]). In order to find an optimal reversal schedule for given values l and s the evaluation cost $\mathbf{T}(\mathbf{t_l}, \mathbf{S})$ is minimised over all admissible reversal schedules $\mathbf{S}$ with $l \le l_{max}(\mathbf{S})$ and $s_{max}(\mathbf{S}) \le s$ (according to (4.30)).

Similar to uniform case, the decomposition of reversal schedules, according to (2.18), can be applied to the evaluation of the minimal cost $\mathbf{T}_{min}(\mathbf{t_l}, s)$. Therefore, $\mathbf{T}_{min}(\mathbf{t_l}, s)$ can be defined by

$$(2.75) \qquad \mathbf{T}_{min}(\mathbf{t_l}, s) = \min_{0<\tilde{l}<l} \left\{ \sum_{j=1}^{\tilde{l}} t_j + \mathbf{T}_{min}(\mathbf{t}_{\tilde{\mathbf{l}}+\mathbf{1},\mathbf{l}}, s-1) + \mathbf{T}_{min}(\mathbf{t}_{\mathbf{1},\tilde{\mathbf{l}}}, s) \right\},$$

with $\mathbf{t}_{\tilde{\mathbf{l}}+\mathbf{1},\mathbf{l}} = \langle t_{\tilde{l}+1}, ..., t_l \rangle^\top$ and $\mathbf{t}_{\mathbf{1},\tilde{\mathbf{l}}} = \langle t_1, ..., t_{\tilde{l}} \rangle^\top$. Furthermore, we introduce the following notation

> $\mathbf{S}_{min}(\mathbf{t_l}, s)$ is an optimal reversal schedule for l time steps and s available checkpoints, provided a non-uniform step cost distribution $\mathbf{t_l} = \langle t_1, ..., t_l \rangle^\top$ is fulfilled.

Obviously, due to the construction of optimal reversal schedules, the following relation is satisfied

$$\mathbf{T}(\mathbf{S}_{min}(\mathbf{t_l}, s)) = \mathbf{T}_{min}(\mathbf{t_l}, s). \tag{2.76}$$

Using the monotonic property (see [59]), run-time needed to construct an optimal reversal schedule $\mathbf{S}_{min}(\mathbf{t_l}, s)$ can be reduced to $\mathcal{O}(sl^2)$. For details see [91].

For a sequence of eight time steps an optimal reversal schedule $\mathbf{S}_{min}(\mathbf{t_8}, 3)$ is constructed using the approach in [91]. The corresponding step cost distribution $\mathbf{t_8}$ is shown in Figure 2.14. The reversal schedule $\mathbf{S}_{min}(\mathbf{t_8}, 3)$ is illustrated in Figure 2.15. Here, the following relation is satisfied

$$\mathbf{T}(\mathbf{S}_{min}(\mathbf{t_8}, 3)) = \mathbf{T}_{min}(\mathbf{t_8}, 3) = 14.$$

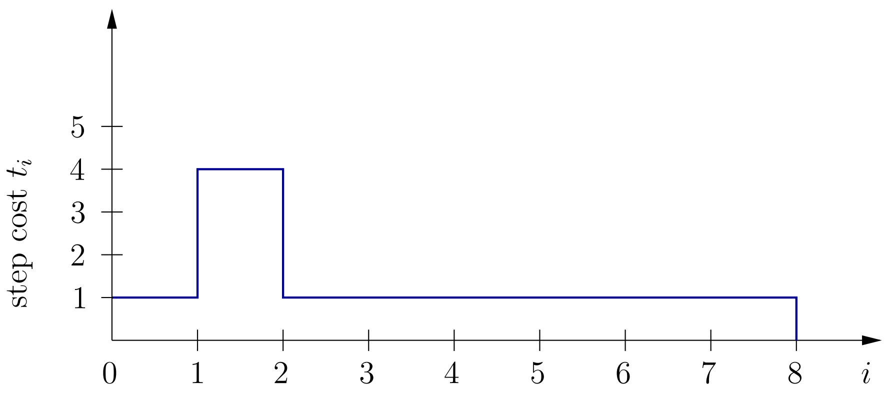

Figure 2.14: Non-uniform step cost distribution $\mathbf{t_8}$

2.3 Estimation of Frequency Numbers

To evaluate an upper bound for frequency numbers $m_j(\mathbf{S})$, $0 \leq j \leq r_{max}(\mathbf{S})$, of a reversal schedule $\mathbf{S}$, it is necessary to prove the following assertion.

Lemma 2.5 (Estimation of Frequency Numbers) *For frequency numbers $m_j(\mathbf{S})$ of any reversal schedule $\mathbf{S}$ with $l_{max}(\mathbf{S}) \geq l$ and $s_{max}(\mathbf{S}) \leq s$ the following property is satisfied*

$$m_j(\mathbf{S}) \leq \beta(s-1, j) = \binom{s-1+j}{s-1}, \qquad 0 \leq j \leq r_{bin}(s, l). \tag{2.77}$$

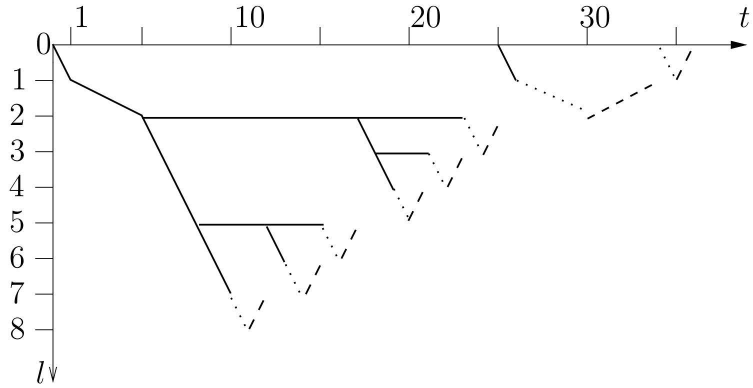

Figure 2.15: Optimal reversal schedule $\mathbf{S}_{min}(\mathbf{t_8}, 3)$ for a forward simulation F with non-uniform step cost distribution $\mathbf{t_8}$ illustrated in Figure 2.14

The upper bound $\beta(s-1, j)$ *is achieved by the frequency numbers* $m_j(l, s)$ *of the binomial reversal schedule* $\mathbf{S}_{bin}(l, s)$ *for* $0 \leq j < r_{bin}(s, l)$.

Proof: For the proof of this Lemma see [50, 83].

Lemma 2.5 implies for $j > r_{bin}(s, l)$ the following conclusion

$$m_j(\mathbf{S}) \geq 0, \qquad m_j(l, s) = 0, \qquad j > r_{bin}(s, l), \tag{2.78}$$

with $\mathbf{S}$ being any arbitrary reversal schedule satisfying the assumptions of Lemma 2.5. Now, it is to clarify the relation between the frequency numbers $m_j(l, s)$ and $m_j(\mathbf{S})$ of the binomial reversal schedule $\mathbf{S}_{bin}(l, s)$ and any arbitrary reversal schedule $\mathbf{S}$, respectively, for $j = r_{bin}(s, l)$. Actually, the relation $m_j(l, s) = \beta(s-1, j)$ is no more satisfied for any arbitrary number l.

For l satisfying

$$\beta(s, j-1) = l_{max}(s, j-1) < l < l_{max}(s, j) = \beta(s, j),$$

the following relation is valid

$$m_j(\mathbf{S}) \leq \beta(s-1, j) = m_j(l, s) + l_{max}(s, j) - l. \tag{2.79}$$

Therefore, $m_j(l, s) = l + \beta(s-1, j) - \beta(s, j) = l - \beta(s, j-1)$. If l achieves the upper bound $l_{max}(s, j)$, i.e.

$$l = l_{max}(s, j) = \beta(s, j),$$

we obtain the following inequality

$$m_j(\mathbf{S}) \leq m_j(l, s) = \beta(s-1, j). \tag{2.80}$$

Thus, for $l = l_{max}(s, j) = \beta(s, j)$ the upper bound $\beta(s-1, j)$ is achieved by the frequency numbers $m_j(l, s)$ of the binomial reversal schedule $\mathbf{S}_{bin}(l, s)$ (For details see [50, 83]).

2.4 Evaluation of Lower and Upper Bounds

Consider a sequence of l time steps F_i, $1 \leq i \leq l$, with a step cost distribution $\mathbf{t_l} = \langle t_1, ..., t_l \rangle^\top$ and s available checkpoints. An appropriate optimal reversal schedule $\mathbf{S}_{min}(\mathbf{t_l}, s)$ for this sequence can be constructed using one of the algorithms in Section 2.2. Run-time requirement needed for the construction of optimal reversal schedules in non-uniform case could be huge. Therefore, it is convenient to determine an estimation for the minimal evaluation cost $\mathbf{T}_{min}(\mathbf{t_l}, s)$ before an optimal reversal schedule is constructed. Lower and upper bounds for the minimal evaluation cost $\mathbf{T}_{min}(\mathbf{t_l}, s)$ can be computed using appropriate frequency numbers. The following notations are introduced:

$$\begin{aligned} U(\mathbf{t_l}, s) &= \text{ upper bound of the minimal evaluation cost } \mathbf{T}_{min}(\mathbf{t_l}, s), \\ L(\mathbf{t_l}, s) &= \text{ lower bound of the minimal evaluation cost } \mathbf{T}_{min}(\mathbf{t_l}, s). \end{aligned}$$

Obviously, the following relations are satisfied

$$L(\mathbf{t_l}, s) \leq \mathbf{T}_{min}(\mathbf{t_l}, s) \leq U(\mathbf{t_l}, s). \tag{2.81}$$

Moreover, due to (4.30), we obtain

$$L(\mathbf{t_l}, s) \leq \mathbf{T}_{min}(\mathbf{t_l}, s) \leq \mathbf{T}(\mathbf{t_l}, \mathbf{S}), \tag{2.82}$$

for any arbitrary reversal schedule $\mathbf{S}$ with $l \leq l_{max}(\mathbf{S})$ and $s_{max}(\mathbf{S}) \leq s$. In the beginning, perform a permutation of a step cost distribution $\mathbf{t_l}$ defined by

$$t_i \to t_i^p, \qquad 1 \leq i \leq l, \tag{2.83}$$

so that $t_{i-1}^p \geq t_i^p$, $2 \leq i \leq l$. Therefore, a lower bound $L(\mathbf{t_l}, s)$ for the minimal evaluation cost $\mathbf{T}_{min}(\mathbf{t_l}, s)$ is evaluated by

$$L(\mathbf{t_l}, s) = \sum_{j=0}^{r_{bin}(s,l)} \sum_{i=1}^{m_j(l,s)} j \cdot t_k^p, \tag{2.84}$$

with $k = i + \sum_{u=0}^{j-1} m_u(l,s) = i + \sum_{u=0}^{j-1} \beta(s-1,u) = i + \beta(s,j-1)$ and $m_j(l,s),\ 0 \leq j \leq r_{bin}(s,l)$, being the frequency numbers of the binomial reversal schedule $\mathbf{S}_{bin}(l,s)$.

In order to evaluate $L(\mathbf{t_l}, s)$, the $m_0(l,s)$ most expensive time steps are chosen; their costs are multiplied by 0. Due to Definition 2.6, $m_0(l,s) = 1$. Therefore, the most expensive time step is ignored during the evaluation of the lower bound $L(\mathbf{t_l}, s)$. Then, the next $m_1(l,s)$ most expensive steps are chosen. Their costs are multiplied by 1 and added to the previous value of $L(\mathbf{t_l}, s)$, and so on. Finally, the $m_{r_{bin}(s,l)}(l,s)$ least expensive time steps are chosen, their costs are multiplied by $r_{bin}(s,l)$, and added to the overall sum $L(\mathbf{t_l}, s)$. Hence, the resulting sum corresponds to a lower bound $L(\mathbf{t_l}, s)$ of the minimal evaluation cost $\mathbf{T}_{min}(\mathbf{t_l}, s)$.

The evaluation of the upper bound $U(\mathbf{t_l}, s)$ proceeds in the same manner as the evaluation of the lower bound $L(\mathbf{t_l}, s)$, i.e. using frequency numbers $m_j(l,s)$. The difference is that here the summation proceeds at first over the least expensive and then over the most expensive time steps. The upper bound $U(\mathbf{t_l}, s)$ for the minimal evaluation cost $\mathbf{T}_{min}(\mathbf{t_l}, s)$ is determined by

$$U(\mathbf{t_l}, s) = \sum_{j=0}^{r_{bin}(s,l)} \sum_{i=1}^{m_j(l,s)} j \cdot t_k^p, \tag{2.85}$$

with $k = \left(l - i - \sum_{u=0}^{j-1} m_u(l,s) \right) + 1 = (l - i - \beta(s, j-1)) + 1$. The construction of the upper bound $U(\mathbf{t_l}, s)$ can be interpreted in the following way: the binomial reversal schedule $\mathbf{S}_{bin}(l,s)$ is applied to the reversal of a sequence of l time steps with non-uniform step costs $\mathbf{t_l}$, so that the step cost distribution is extremely inefficient, i.e. expensive steps are frequently evaluated but inexpensive steps are evaluated rarely. Thus, the following inequality is satisfied

$$\mathbf{T}_{min}(\mathbf{t_l}, s) \leq U(\mathbf{t_l}, s). \tag{2.86}$$

Note that in contrast to the lower bound $L(\mathbf{t_l}, s)$, the relation (2.86) must not be satisfied for the evaluation cost $\mathbf{T}(\mathbf{t_l}, \mathbf{S})$ of any arbitrary reversal schedule $\mathbf{S}$.

Recalling Corollary 2.1 and its conclusions, we obtain that the formulae (2.84) and (2.85) for the lower $L(\mathbf{t_l}, s)$ and upper $U(\mathbf{t_l}, s)$ bounds can be simplified to

$$(2.87) \qquad L(\mathbf{t_l}, s) = \sum_{i=1}^{l} t_i^p r_{bin}(s,i), \qquad U(\mathbf{t_l}, s) = \sum_{i=1}^{l} t_i^p r_{bin}(s, l-i+1).$$

An example for lower and upper bounds $L(\mathbf{t_l}, s)$ and $U(\mathbf{t_l}, s)$, evaluated by (2.84) and (2.85) or (2.87), is illustrated in Figure 2.17. Three checkpoints are available. A corresponding step cost distribution $\mathbf{t_l}$ is shown in Figure 2.16.

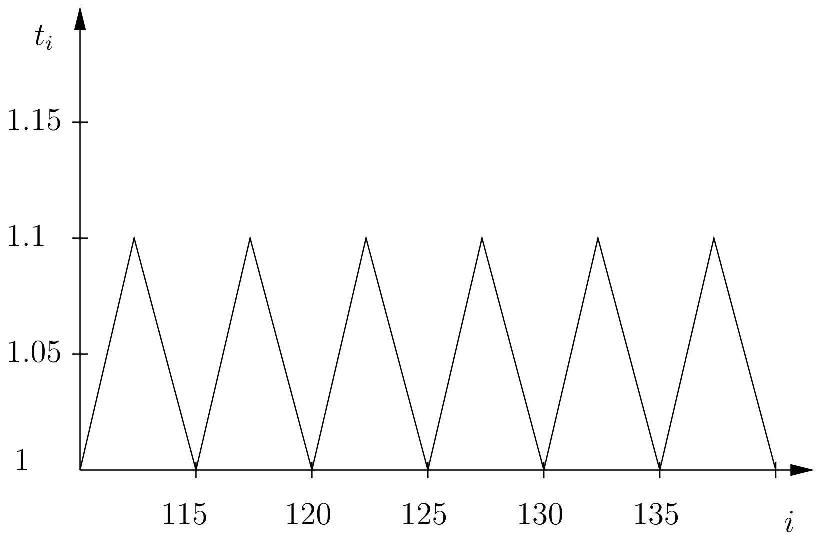

Figure 2.16: Example of a non-uniform step cost distribution $\mathbf{t_l}$

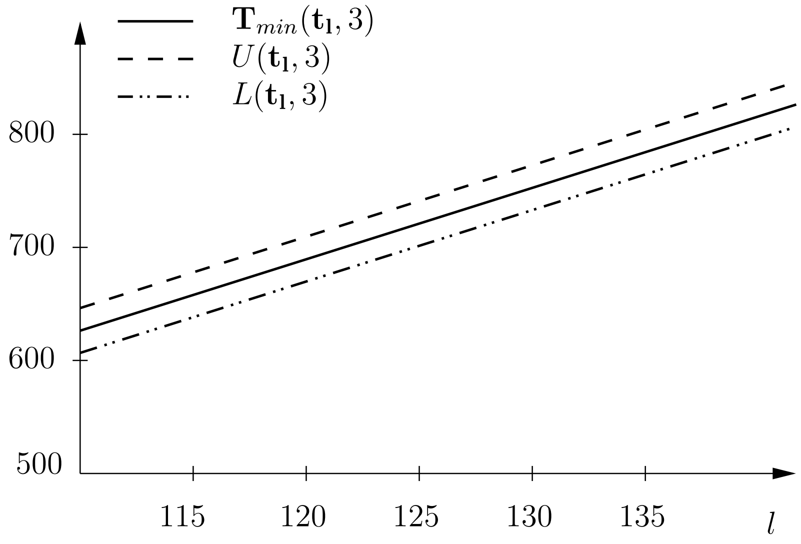

Figure 2.17: Example for $U(\mathbf{t_l}, 3)$ and $L(\mathbf{t_l}, 3)$ with the step cost distribution $\mathbf{t_l}$ in Figure 2.16

Chapter 3

Survey of Numerical Methods for Separated BVPs in ODEs

3.1 Euler-Lagrange Equations

Consider the following unconstrained primal control problem of an initial value ordinary differential equation, with Bolza objective

$$\min_{\mathbf{x},\mathbf{u}} \Phi(\mathbf{x},\mathbf{u}) = \min_{\mathbf{x},\mathbf{u}} \phi(\mathbf{x}(T)) + \int_0^T L(\mathbf{x}(t),\mathbf{u}(t),t)dt, \tag{3.1}$$

where the system is described by

$$\dot{\mathbf{x}} = f(\mathbf{x}(t),\mathbf{u}(t),t), \quad \mathbf{x}(0) = \mathbf{x}_0. \tag{3.2}$$

Here, $\mathbf{x} : [0,T] \to \mathbf{R}^n$, $\mathbf{u} : [0,T] \to \mathbf{R}^m$, $L, f : \mathbf{R}^n \times \mathbf{R}^m \times [0,T] \to \mathbf{R}$, $\phi : \mathbf{R}^n \to \mathbf{R}$. The control function $\mathbf{u}(t)$ is assumed to be an essentially bounded function, i.e., an element of the Banach space $L^\infty(0,T;\, \mathbf{R}^m)$ which is endowed with the norm $\|\mathbf{u}\|_\infty \equiv \textit{ess sup}\,\{\mid \mathbf{u}(t) \mid, 0 \le t \le T\}$. Here, $\mid \cdot \mid$ stands for the Euclidean norm in $\mathbf{R}^m$. The state function $\mathbf{x}(t)$ is an element of the space $W^{1,\infty}(0,T;\, \mathbf{R}^n)$ of uniformly Lipschitz-continuous functions which is a Banach space under the norm $\|\mathbf{x}\|_{1,\infty} \equiv \mid \mathbf{x}(0) \mid + \|\dot{\mathbf{x}}\|_\infty$.

The problem is to find the function $\mathbf{u}(t)$ that minimises (3.1). Adjoin the system of differential equations (3.2) to (3.1) with multiplier function $\bar{\mathbf{x}}(t)$:

$$\mathbf{J} = \phi(\mathbf{x}(T)) + \int_0^T (L(\mathbf{x}(t),\mathbf{u}(t),t) + \bar{\mathbf{x}}^\top(t)\,(f(\mathbf{x}(t),\mathbf{u}(t),t) - \dot{\mathbf{x}}))dt. \tag{3.3}$$

The scalar function H (the **Hamiltonian**) can be written as

$$H(\mathbf{x}(t),\mathbf{u}(t),\bar{\mathbf{x}}(t),t) = L(\mathbf{x}(t),\mathbf{u}(t),t)) + \bar{\mathbf{x}}^\top(t)\,f(\mathbf{x}(t),\mathbf{u}(t),t). \tag{3.4}$$

After integrating the last term of (3.3) by parts we obtain:
(3.5)
$$\mathbf{J} = \phi(\mathbf{x}(T)) - \bar{\mathbf{x}}^\top(T)\mathbf{x}(T) + \bar{\mathbf{x}}^\top(0)\mathbf{x}(0) + \int_0^T \{H(\mathbf{x}(t), \mathbf{u}(t), t)) + \dot{\bar{\mathbf{x}}}^\top(t)\mathbf{x}(t)\} dt.$$

Now, consider the variation in J due to variations in the control vector $\mathbf{u}(t)$ for fixed times $t_0 = 0$ and $t_f = T$, i.e.
(3.6)
$$\delta\mathbf{J} = \left[\left(\frac{\partial\phi}{\partial\mathbf{x}} - \bar{\mathbf{x}}^\top\right)\delta\mathbf{x}\right]_{t=T} + [\bar{\mathbf{x}}^\top\delta\mathbf{x}]_{t=0} + \int_0^T \left[\left(\frac{\partial H}{\partial\mathbf{x}} + \dot{\bar{\mathbf{x}}}^\top\right)\delta\mathbf{x} + \frac{\partial H}{\partial\mathbf{u}}\,\delta\mathbf{u}\right] dt.$$

We choose the multiplier function $\bar{\mathbf{x}}(t)$ to cause the coefficients of $\delta\mathbf{x}$ in (3.6) vanish:

$$\dot{\bar{\mathbf{x}}} = -H_{\mathbf{x}}^\top = -L_{\mathbf{x}}^\top - f_{\mathbf{x}}^\top\,\bar{\mathbf{x}}, \tag{3.7}$$

with boundary conditions

$$\bar{\mathbf{x}}(T) = \phi_{\mathbf{x}}^\top(\mathbf{x}(T)). \tag{3.8}$$

Note that here $\dot{\bar{\mathbf{x}}}$ represents the total time derivative of $\bar{\mathbf{x}}$. At each point along the solution path the Hamiltonian function must be minimal with respect to the control value $\mathbf{u}(t)$. Therefore, we obtain the **First Order Necessary Optimality Condition**

$$\left(\frac{\partial H}{\partial\mathbf{u}}\right)^\top = 0, \qquad 0 \le t \le T, \tag{3.9}$$

for the optimal control problem (3.1) - (3.2).

Equations (3.2), (3.7), (3.8), and (3.9) are known as the **Euler-Lagrange equations** in the calculus of variations. In summary, to find a control vector function $\mathbf{u}(t)$ that produces a stationary value of the performance index J, we must solve the following differential equations:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, t), \tag{3.10}$$

$$\dot{\bar{\mathbf{x}}} = -L_{\mathbf{x}}^\top - f_{\mathbf{x}}^\top\,\bar{\mathbf{x}}, \tag{3.11}$$

where $\mathbf{u}(t)$ is determined by

$$\left(\frac{\partial H}{\partial\mathbf{u}}\right)^\top = 0, \qquad \text{or} \qquad f_{\mathbf{u}}^\top\,\bar{\mathbf{x}} + L_{\mathbf{u}}^\top = 0. \tag{3.12}$$

The boundary conditions for (3.10) and (3.11) are separated; one part of them are given for $t = 0$, the other one for $t = T$:

$$\mathbf{x}(0) = \mathbf{x}_0, \tag{3.13}$$

$$\bar{\mathbf{x}}(T) = \phi_{\mathbf{x}}^{\top}(\mathbf{x}(T)). \tag{3.14}$$

Thus, we are facing with a **two-point boundary-value problem**.

For the sake of simplicity, we will consider control problems with $L \equiv 0$ for all $0 \le t \le T$. This is because the initial optimal control problem (3.1) - (3.2) can be transformed into a simplified one by introducing a new state variable

$$\mathbf{x}_{n+1}(t) = \int_0^t L(\mathbf{x}(\tilde{t}), \mathbf{u}(\tilde{t}), \tilde{t}) d\tilde{t}. \tag{3.15}$$

For the derivative of $\mathbf{x}_{n+1}(t)$ we obtain:

$$\dot{\mathbf{x}}_{n+1}(t) = L(\mathbf{x}(t), \mathbf{u}(t), t), \qquad \mathbf{x}_{n+1}(0) = 0. \tag{3.16}$$

The control problem (3.1) - (3.2) takes the form:

$$\min_{\mathbf{x},\mathbf{u}} \phi(\mathbf{x}(T)) + \mathbf{x}_{n+1}(T) = \min_{\mathbf{X},\mathbf{u}} \tilde{\phi}(\mathbf{X}(T)), \tag{3.17}$$

where the system is described by

$$\dot{\mathbf{X}} = \tilde{f}(\mathbf{X}(t), \mathbf{u}(t), t), \quad \mathbf{X}(0) = \mathbf{X}_0 = [\mathbf{x}_0^{\top}, 0]^{\top}, \tag{3.18}$$

with $\mathbf{X} = (x_1, ..., x_n, x_{n+1})^{\top} : [0, T] \to \mathbf{R}^{n+1}$, $\tilde{f} = [f^{\top}(\mathbf{x}, \mathbf{u}, t), L(\mathbf{x}, \mathbf{u}, t)]^{\top} : \mathbf{R}^{n+1} \times \mathbf{R}^m \times [0, T] \to \mathbf{R}^{n+1}$. The Euler-Lagrange adjoint equation for the transformed problem (3.17) - (3.18) is

$$\dot{\bar{\mathbf{X}}} = -H_{\mathbf{X}}^{\top} = -\tilde{f}_{\mathbf{X}}^{\top} \bar{\mathbf{X}}, \tag{3.19}$$

with boundary and first order necessary optimality conditions

$$\bar{\mathbf{X}}(T) = \tilde{\phi}_{\mathbf{X}}^{\top}(\mathbf{X}(T)), \tag{3.20}$$

$$\left(\frac{\partial H}{\partial \mathbf{u}}\right)^{\top} = 0, \qquad 0 \le t \le T. \tag{3.21}$$

The Hamiltonian has only the last term:

$$H(\mathbf{X}(t), \mathbf{u}(t), \bar{\mathbf{X}}(t), t) = \bar{\mathbf{X}}^\top(t)\, \tilde{f}(\mathbf{X}(t), \mathbf{u}(t), t). \tag{3.22}$$

A discrete-time counterpart of the problem (3.1) - (3.2) is the problem

$$\begin{aligned} &\min_{\mathbf{x},\mathbf{u}} \sum_{i=0}^{l-1} L_i(\mathbf{x}_i, \mathbf{u}_i) + \phi(\mathbf{x}_l), \\ \mathbf{x}_{i+1} \;&=\; f_i(\mathbf{x}_i, \mathbf{u}_i), \quad i = 0, ..., l-1, \qquad \mathbf{x}_0 \text{ fixed}, \end{aligned} \tag{3.23}$$

which has the standard form (3.17) - (3.18) by introducing a new component $v_i \in \mathbf{R}$, defined by

$$v_{i+1} = v_i + L_i(\mathbf{x}_i, \mathbf{u}_i), \qquad v_0 = 0. \tag{3.24}$$

Then, $\sum_{i=0}^{l-1} L_i(\mathbf{x}_i, \mathbf{u}_i) + \phi(\mathbf{x}_l) = v_l + \phi(\mathbf{x}_l)$. Therefore, in the sequel of these thesis consider the following unconstrained primal control problem of an initial value ordinary differential equation

$$\min_{\mathbf{x},\mathbf{u}} \phi(\mathbf{x}(T)), \tag{3.25}$$

where the system is described by

$$\dot{\mathbf{x}} = f(\mathbf{x}(t), \mathbf{u}(t), t), \quad \mathbf{x}(0) = \mathbf{x}_0. \tag{3.26}$$

Now, we need two assumptions in order to formulate the **Second Order Sufficient Conditions** for unconstrained control problems.

Assumption 3.1 Strict Legendre-Clebsch Condition
The relation

$$H_{\mathbf{uu}}(t) > 0 \tag{3.27}$$

is valid on the whole interval $[0, T]$.

Assumption 3.2 Coercivity Condition
There exists a solution of the Riccati equation

$$\dot{Q} = -Qf_{\mathbf{x}}(t) - f_{\mathbf{x}}^T(t)Q - H_{\mathbf{xx}}(t) + (H_{\mathbf{xu}}(t) + Qf_{\mathbf{u}}(t))\, H_{\mathbf{uu}}^{-1}(t)\, (H_{\mathbf{xu}}(t) + Qf_{\mathbf{u}}(t))^\top, \tag{3.28}$$

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

$$Q(T) = \phi_{\mathbf{xx}}^\top(\mathbf{x}(T)). \tag{3.29}$$

The next theorem summarises the SSC for a weak local minimum which are to be found in [63, 66, 75, 98].

Theorem 3.1 SSC for Control Problems
Let $(\mathbf{x}_0, \mathbf{u}_0)$ *be admissible for the problem (3.1) - (3.2). Suppose, there exist multipliers* $\bar{\mathbf{x}} \in W^{1,\infty}(0, T;\, \mathbf{R}^n)$*, so that the Euler-Lagrange equations (3.2), (3.7) - (3.9) and Assumptions 3.1 and 3.2 are satisfied. Then, there exist* $c > 0$ *and* $\alpha > 0$*, so that*

$$\Phi(\mathbf{x}, \mathbf{u}) \geq \Phi(\mathbf{x}_0, \mathbf{u}_0) + c\left\{\|\mathbf{x} - \mathbf{x}_0\|_{1,2}^2 + \|\mathbf{u} - \mathbf{u}_0\|_2^2\right\} \tag{3.30}$$

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 (3.1) - (3.2).*

3.1.1 Solution Approaches Classification

There are some methods for solving the optimal control problem (3.1) - (3.2). They can be divided in two classes: **direct** and **indirect** methods.

The indirect methods can be characterised by solving the boundary-value problem, which is a result of first order necessary conditions. This boundary-value problem is solved with respect to the state and adjoint variables. These methods yield highly accurate results, but they have some drawbacks as well. First of them is that sometimes it is not possible to construct the boundary-value problem explicitly. It means that we have to express the control function $\mathbf{u}$ in terms of $\mathbf{x}$ and $\bar{\mathbf{x}}$ from the relation (3.9). The second disadvantage of these methods is that often we have to find a very good initial guess or good estimations for the adjoint variables in order to achieve the required convergence to the continuous solution.

Direct methods have been proposed to avoid the drawbacks of the indirect approach. Here, the optimal control problem is discretised using a suitable discretisation. The obtained discretised problem is treated as a non-linear programming problem (**NLP**) with respect to the discretised state and control variables. We have two types of discretisation: **full**, where both, state and control functions, are discretised, and the **partially** discretisation, where the discretisation is proceeded only with respect to the control functions and the discrete state variables are evaluated by the integration of the state equations. The direct approach has an advantage compared to the indirect approach, since it does not need any estimation for the values of the adjoint variables.

In the next section an overview for the indirect methods for solving optimal control problems is given.

3.2 Initial Value Methods for Linear Problems

There are close theoretical relationships between BVPs and IVPs. The basic theorems of solution existence and representation for a BVP are given in terms of fundamental solutions, which are solutions of corresponding IVPs. As described in [3], it makes sense to construct a numerical method for a given BVP by relating the problem to associated IVPs, and solving the latter numerically. The present section describes methods for BVPs, which are based on the IVPs approaches. The following description closely follows the description in [3].

3.2.1 Single Shooting for Linear Problems

The simplest initial value method for BVPs is the **single shooting** method. Unfortunately, it can also have stability drawbacks. Consider the general linear two-point BVP

(3.31) $$\dot{y} = A(t)y + q(t), \qquad a < t < b,$$

(3.32) $$B_a y(a) + B_b y(b) = \beta.$$

$A(t), B_a, B_b \in \mathbf{R}^{n\times n}$, $y(t), y(a), y(b), \beta \in \mathbf{R}^n$. The solution of this BVP is represented as a linear combination of solutions to associated IVPs. This method is called the **superposition method**. The general solution of (3.31) can be represented as

(3.33) $$y(t) = Y(t)s + v(t), \qquad a \leq t \leq b,$$

where $Y(t)$ is a fundamental solution, s is a parameter vector ($s \in \mathbf{R}^n$), and $v(t)$ is a particular solution. Let $Y(t) \equiv Y(t;\, a)$ be the fundamental solution satisfying

(3.34) $$\dot{Y} = A(t)Y, \qquad Y(a) = I, \qquad a \leq t \leq b.$$

The particular solution $v(t)$ can be determined as the solution of the IVP

(3.35) $$\dot{v} = A(t)v + q(t), \qquad v(a) = \alpha,$$

for some vector α (e.g. $\alpha = 0$). Thus, the n columns of $Y(t)$ and the vector $v(t)$ can be computed as the solution of $n+1$ IVPs. The BVP solution $y(t)$

is given as their superposition in (3.33). To determine the parameters s we substitute (3.33) in (3.32) to obtain

(3.36)
$$\beta = B_a[Y(a)s+v(a)]+B_b[Y(b)s+v(b)] = [B_aI+B_bY(b)]s+B_av(a)+B_bv(b),$$

or

$$Qs = \hat{\beta}, \tag{3.37}$$

with

$$Q = B_a + B_bY(b), \tag{3.38}$$

$$\hat{\beta} = \beta - B_av(a) - B_bv(b). \tag{3.39}$$

Then, the solution for the BVP is given by (3.33). The practical implementation of the superposition method is a numerical approximation to the quantities appearing in (3.33) - (3.39).

The shooting principle can be extended to non-linear problems as well. Our aim is to find a correct angle or a parameter s, so that the solution of the IVP also solves the BVP (see Fig. 3.1 for $n = 1$). In general, the shooting method

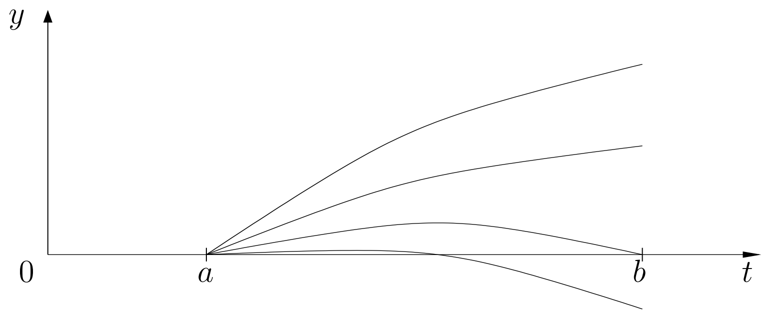

Figure 3.1: Shooting

will give appropriate global discretisation errors if the original problem (3.31) - (3.32) is well-conditioned.

Stability difficulties arise when the differential operator in (3.31) contains rapidly growing fundamental solution modes. Then, the initial value solution is very sensitive to small changes in the initial value s of (3.33). If we have no fast-decaying modes, then, the shooting method can still be made to work well if we do the IVP integrations in the **reverse** direction. That is so, because if the BVP is well-conditioned, then, a fast-growing mode from a

to b is scaled down by the BC at b. The trick of reversing the direction of integration will not work, however, if there are both fast-growing and fast-decaying modes present.

A major disadvantage of the single shooting method is the roundoff accumulation, occurring when unstable IVPs have to be integrated. In an attempt to decrease this bound, it is natural to restrict the size of domains over which IVPs are integrated. This leads to the method of **multiple shooting**.

3.2.2 Multiple Shooting for Linear Problems

Consider again the general linear two-point BVP (3.31) - (3.32) and generalise the superposition method directly. Firstly, the interval of integration $[a, b]$ is subdivided by a mesh

$$a = t_0 < t_1 < \ldots < t_{l-1} < t_l = b. \tag{3.40}$$

Then, as in single shooting, initial value integrations are performed on each subinterval $[t_i, t_{i+1}]$, $0 \le i \le l-1$ (see Figure 3.2 for $l = 5$). The resulting

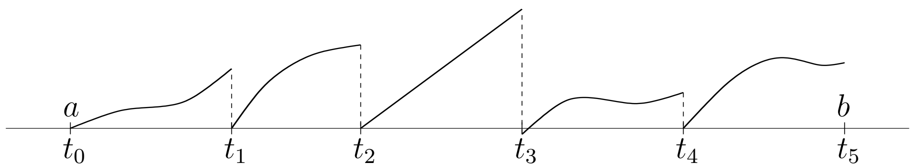

Figure 3.2: Multiple shooting

solution segments are patched up to form a continuous solution over the entire interval $[a, b]$.

On each subinterval $[t_i, t_{i+1}]$, $0 \le i \le l-1$, of the mesh (3.40), we can express the general solution of (3.31) as

$$y(t) = Y_i(t)s_i + v_i(t), \qquad t_i \le t \le t_{i+1}, \tag{3.41}$$

where $Y_i(t)$ is a fundamental solution, s_i is a parameter vector ($s_i \in \mathbf{R}^n$), and $v_i(t)$ is a particular solution. They are defined for each i by the IVPs

$$\dot{Y}_i = A(t)Y_i, \qquad Y_i(t_i) = F_i, \qquad t_i \le t \le t_{i+1}, \tag{3.42}$$

with F_i being an $n \times n$ nonsingular matrix, and

$$\dot{v}_i = A(t)v_i + q(t), \qquad v_i(t_i) = \alpha_i, \qquad t_i \le t \le t_{i+1}. \tag{3.43}$$

Most common choices for F_i and α_i are

$$F_i = I, \qquad \alpha_i = 0, \qquad 0 \le i \le l-1. \tag{3.44}$$

If the matrices F_i are chosen independently from each other, as in (3.44), then, the method is called **parallel shooting** and the problems (3.42) can be integrated parallel. Other variants involve **marching techniques**, where for each i the F_{i+1} is determined only after (3.42) has been solved. If (3.44) holds, then, we call the method **standard multiple shooting**.

The nl parameters

$$s^\top := (s_0^\top, s_1^\top, ..., s_{l-1}^\top) \tag{3.45}$$

are determined, so that the (numerical) solution is continuous over the entire interval $[a, b]$, and so that the BC (3.32) are satisfied. From (3.41) the matching conditions $y(t_{i+1}^-) = y(t_{i+1}^+)$, $0 \le i \le l-2$, are

$$Y_i(t_{i+1})s_i + v_i(t_{i+1}) = Y_{i+1}(t_{i+1})s_{i+1} + v_{i+1}(t_{i+1}), \qquad 0 \le i \le l-2. \tag{3.46}$$

Rearranging and substituting initial values, we obtain

$$F_{i+1}s_{i+1} = Y_i(t_{i+1})s_i + [v_i(t_{i+1}) - \alpha_{i+1}], \qquad 0 \le i \le l-2. \tag{3.47}$$

This recursion together with the BC

$$B_a\,[Y_0(a)s_0 + v_0(a)] + B_b\,[Y_{l-1}(b)s_{l-1} + v_{l-1}(b)] = \beta, \tag{3.48}$$

gives the linear system of nl equations

$$\mathcal{A}s = \hat{\beta}, \tag{3.49}$$

which in component form is

$$\begin{bmatrix} -Y_0(t_1) & F_1 & & & \\ & -Y_1(t_2) & F_2 & & \\ & & \cdot & \cdot & \\ & & & -Y_{l-2}(t_{l-1}) & F_{l-1} \\ B_aF_0 & & & & B_bY_{l-1}(b) \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ \cdot \\ \cdot \\ \cdot \\ s_{l-1} \end{bmatrix} = \begin{bmatrix} v_0(t_1) - \alpha_1 \\ v_1(t_2) - \alpha_2 \\ \cdot \\ \cdot \\ v_{l-2}(t_{l-1}) - \alpha_{l-1} \\ \beta - B_a\alpha_0 - B_bv_{l-1}(b) \end{bmatrix}. \tag{3.50}$$

Here, in multiple shooting, the solution is given by

$$y(t_i) = F_is_i + \alpha_i. \tag{3.51}$$

In particular, $y(t_i) = s_i$ for the standard multiple shooting method. If we solve a BVP using multiple shooting method, we have in fact **two-level**

discretisation process: first, by a coarse mesh (3.40), and then, by a finer discretisation in each subinterval for the numerical solution of the IVPs.

Since the main disadvantage of the single shooting was its instability, we want to investigate how the stability properties with respect to the roundoff errors are improved with the multiple shooting. Assume, the initial value integrations use $\hat{l}$ steps, so the combined fine mesh of the two-level multiple shooting process has

$$\tilde{l} := l\hat{l} \tag{3.52}$$

mesh points. For simplicity, let us consider the standard multiple shooting method and denote

$$K := \max_{0 \le i \le l-1} \|Y(t_{i+1};\, t_i)\|, \tag{3.53}$$

then, $\|A\| \approx K$. Thus, it can be shown (see [3]) that if the BVP is well-conditioned, and, if the shooting points are chosen, so that K in (3.53) is of moderate size (with $\tilde{l}$ not extremely large), then, the standard multiple shooting method is stable, in the sense that there is a tolerable roundoff error application.

The practical choice of shooting points is not easy. That is, the IVP integration from a shooting point t_i proceeds until $\|Y_i(t)\|$ gets large. Then, the integration stops and a new shooting point t_{i+1} is set. However, this way is not always the most efficient way to go, since the integration steps tend to decrease. Thus, longer shooting intervals may well be less desirable, as they require more integration steps. On the over hand, one have to restart the integration at each new shooting point. Then, some compromise is necessary.

Other factors are involved as well: for example the question of evaluating the computed solution. If there are many output points, where the BVP solution value is desired, one possibility is to integrate for a given output point $\hat{t}$, $t_i < \hat{t} \le t_{i+1}$, the IVP

$$\dot{y} = A(t)y + q(t), \qquad y(t_i) = s_i, \qquad t_i < t \le t_{i+1}, \tag{3.54}$$

which is similar to, but much more stable than the IVP in the last step of single shooting algorithm. Another possibility is to interpolate $y(\hat{t})$ from the given values $y(t_i) = s_i$, $0 \le i \le l-1$.

3.2.3 Compactification

In this section the **compactification method** will be explained which can be applied to solve the linear system (3.49) - (3.50). The idea of this method is very simple: use the recursion relation (3.47) to express s_{l-1} in terms of s_0; substitute this in the BC (3.48) and obtain an $n \times n$ system of linear equations for s_0; solve for s_0; finally use (3.47) again to obtain s. We express this method for a standard multiple shooting scheme.

Thus, we compute a particular solution $\{\psi_i\}_{i=0}^{l}$ and a fundamental solution $\{\Phi_i\}_{i=0}^{l}$ by

$$\psi_{i+1} = Y(t_{i+1};\ t_i)\psi_i + v_i(t_{i+1}), \qquad \psi_0 = 0, \qquad 0 \le i \le l-1, \tag{3.55}$$

$$\Phi_{i+1} = Y(t_{i+1};\ t_i)\Phi_i, \qquad \Phi_0 = I, \qquad 0 \le i \le l-1, \tag{3.56}$$

and form

$$s_i = \Phi_i s_0 + \psi_i, \qquad 0 \le i \le l-1, \tag{3.57}$$

where s_0 is obtained as the solution of the $n \times n$ linear system

$$[B_a + B_b \Phi_l] s_0 = \beta - B_b \psi_l. \tag{3.58}$$

3.2.4 Marching Techniques for Multiple Shooting

Consider again the solution of the BVP (3.31) - (3.32) by multiple shooting. A marching technique is one where the IVPs (3.42) are solved consecutively, with the initial matrix F_{i+1} for the $(i+1)$st interval being formed only after the IVP for the ith subinterval has been solved.

The motivation behind is to construct not only F_{i+1}, but also the next shooting point t_{i+1} from the following reasoning: in order to make multiple shooting algorithm stable, it is necessary to monitory the growth of the fundamental solution modes to decide where the next shooting point should be. Since these basis functions tend to resemble more and more the most dominant one, they become more and more numerically linearly dependent as the integration interval gets larger. Thus, when some tolerance of dependence is exceeded, a new shooting point t_{i+1} should be set and the columns of $Y_i(t_{i+1})$ should be reorthogonilised to restore their linear dependence.

The reorthogonalisation turns out to produce a decoupling of growing and decreasing solution components, enabling us to construct a stable version of the compactification algorithm.

3.2.5 Reorthogonalisation

We start integrating the IVPs (3.42) and (3.43) for $i = 0$, assuming that F_0 is an orthogonal matrix. For the general step i, $0 \leq i \leq l-1$, we begin the IVPs integrations with an orthogonal matrix F_i. Then, in the next shooting point t_{i+1} we perform a QU-decomposition of $Y_i(t_{i+1})$, i.e.

$$Y_i(t_{i+1}) = F_{i+1}G_i, \qquad 0 \leq i \leq l-1, \tag{3.59}$$

with F_{i+1} orthogonal and G_i upper triangular. Regarding the particular solution $v_i(t)$, we take $\alpha_i = 0$.

Consider the recursion relation (3.47). Defining

$$\tilde{\beta}_i := F_{i+1}^\top (v_i(t_{i+1}) - \alpha_{i+1}), \qquad 0 \leq i \leq l-2, \tag{3.60}$$

$$\tilde{\beta}_{l-1} := \beta - B_a\alpha_0 - B_b v_{l-1}(b), \qquad \tilde{\beta}^\top = (\tilde{\beta}_0^\top, ..., \tilde{\beta}_{l-2}^\top, \tilde{\beta}_{l-1}^\top), \tag{3.61}$$

we find

$$s_{i+1} = G_i s_i + \tilde{\beta}_i, \qquad 0 \leq i \leq l-2, \tag{3.62}$$

with G_i upper triangular. Incorporating the BC, we obtain

$$\mathcal{A}s = \tilde{\beta}, \tag{3.63}$$

$$\mathcal{A} = \begin{bmatrix} -G_0 & I & & & \\ & -G_1 & I & & \\ & & \cdot & \cdot & \\ & & & -G_{l-2} & I \\ B_aF_0 & & & & B_bF_lG_{l-1} \end{bmatrix}. \tag{3.64}$$

3.2.6 Decoupling

Let us examine the connections between G_i here and the fundamental solutions of the standard multiple shooting method. We have

$$Y(t_{i+1};\ t_i) = Y_i(t_{i+1})[Y_i(t_i)]^{-1} = F_{i+1}G_iF_i^\top, \tag{3.65}$$

so that

$$Y(t_{i+1};\ a) = Y(t_{i+1}, t_i)Y(t_i, t_{i-1})...Y(t_1, a) = F_{i+1}G_i...G_0F_0^\top, \tag{3.66}$$

(3.67) $$Y_0(t_{i+1}) = F_{i+1}G_i...G_0 = F_{i+1}\hat{G}_i.$$

Now, $\hat{G}_i = G_i...G_0$ is upper triangular. Hence, (3.67) gives the QU decomposition of $Y_0(t_{i+1})$.

Consider the linear space spanned by the first k columns of F_{i+1}. Since $\hat{G}_i$ is upper triangular, the first k columns of $Y_0(t_{i+1})$ are, by (3.67), linear combinations of the first k columns of F_{i+1}, i.e. they are in the span of these columns of F_{i+1}. Thus, if F_0 is chosen, so that its first k columns represent somehow initial values of the most dominant modes, then, the span of the first k columns of F_{i+1} represents the directions of their integrated form up to $t = t_{i+1}$. This means that the upper left $k \times k$ block of $\hat{G}_i$ has a norm of the same order of magnitude as the increment of these dominant modes.

There is a counterpart to the right lower $(n-k) \times (n-k)$ block of $\hat{G}_i$ in the case of dichotomy with an $(n-k)$-dimensional decaying solution space.

If an $n \times n$ matrix can be written as

(3.68) $$M = (M^1 \mid M^1L + M^2),$$

where M^1 has k columns and L is some $k \times (n-k)$ matrix, then, a QU decomposition of M, say $M = QR$, gives a right $(n-k) \times (n-k)$ block R^{22} of R, so that $\|R^{22}\|_2 = \|M^2\|_2$ (for the proof of this statement see [3]).

We can utilise this fact in the case of dichotomy. If the first k columns of $Y_0(t_{i+1})$ represent the dominant modes, then, the remaining columns can be written as M^1L+M^2, with M^2 representing values of decaying modes. Then, we have that the right lower $(n-k) \times (n-k)$ block of $\hat{G}_i$ has a norm of the same order of magnitude as the decaying modes.

This means that in (3.62) we can effectively decouple the recursion (3.47), where the decoupling corresponds to the splitting of growing and decaying modes. If we write

(3.69) $$G_i = \begin{pmatrix} D_i & C_i \\ 0 & E_i \end{pmatrix},$$

where D_i is $k \times k$ matrix, and partial vectors $g_i \in \mathbf{R}^n$ as

(3.70) $$g_i = \begin{pmatrix} g_i^1 \\ g_i^2 \end{pmatrix}, \qquad g_i^1 \in \mathbf{R}^k,$$

then, we can write (3.62) as

(3.71) $$s_{i+1}^1 = D_i s_i^1 + C_i s_i^2 + \tilde{\beta}_i^1, \qquad 0 \le i \le l-2,$$

$$(3.72) \qquad s_{i+1}^2 = \qquad E_i s_i^2 + \tilde{\beta}_i^2, \qquad 0 \le i \le l-2.$$

Now, the recursion (3.72) is stable in the forward direction, because $E_i ... E_0 = \hat{G}_i^{22}$, while (3.71) is stable in the backward direction because we expect $\|D_i^{-1} ... D_{l-2}^{-1}\| \approx 1$. It can be shown that it leads to a stable version of the compactification algorithm, applied to the linear system (3.63). This algorithm is much more stable as the compactification algorithm (see [3]).

One of the questions left to be dealt with, if we want to use this algorithm, is to find an appropriate initial matrix F_0, so that the first columns of F_0 span a subspace that generates the growing solutions (assuming that we have an exponential dichotomy).

Suppose, the BC are separated

$$(3.73) \qquad \dot{y} = A(t)y + q(t), \qquad a < t < b,$$

$$(3.74) \qquad \begin{bmatrix} B_{a1} \\ 0 \end{bmatrix} y(a) + \begin{bmatrix} 0 \\ B_{b2} \end{bmatrix} y(b) = \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix},$$

with $B_{a1} \in \mathbf{R}^{(n-k)\times n}$ and $B_{b2} \in \mathbf{R}^{k\times n}$, where k corresponds to the k growing modes of the matrix $A(t)$. Moreover, these matrices have the following structure

$$B_{a1} = [0 \quad \times] \qquad B_{b2} = [\times \quad 0]\,,$$

where $\times$ denotes a quadratic nonzero part of the matrices. Then, the algorithm **Compactification for a Decoupled System** can be written in the following way.

Note, the **Reversal Initialisation** in Algorithm 3.1 can be carried out without knowing all values of Φ_0 and ξ_0. This is because of the separated BC given by (3.74), i.e. B_{a1} controls $(n-k)$ decreasing modes and B_{b2} controls k increasing modes of $A(t)$.

From Algorithm 3.1 we can see that the compactification for a decoupled system consists of two sweeps through the time window $[a, b]$. The simplest strategy is to implement Algorithm 3.1 by 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 a and b. However, this approach can be realised only when there is a sufficiently large amount of memory available. If this is not the case, then, we can apply checkpointing techniques from Chapter 2. The routine implementing Algorithm 3.1 was coded using the

Algorithm 3.1 Compactification for a Decoupled System
Original Initialisation:
$F_0 = I$, $\alpha_i = 0$ *for all* i,
$\xi_0^2 = 0$, $\xi_l^1 = 0$, $\Phi_0^{22} = I_{n-k}$, $\Phi_l^{11} = I_k$, $\Phi_l^{12} = 0$, $\Phi_i^{21} = 0$ *for all* i.
For $i = 0, 1, ..., l-1$ **DO**
  Original Step:
    Solve $\dot{Y}_i = AY_i, \qquad t_i < t < t_{i+1}, \qquad Y_i(t_i) = F_i$.
    Solve $\dot{v}_i = Av_i + q_i, \qquad t_i < t < t_{i+1}, \qquad v_i(t_i) = \alpha_i$.
    Find F_{i+1} *and* G_i *from* $Y_i(t_{i+1}) = F_{i+1}G_i$, $G_i = \begin{pmatrix} D_i & C_i \\ 0 & E_i \end{pmatrix}$.
    Find $\tilde{\beta}_i = \begin{pmatrix} \tilde{\beta}_i^1 \\ \tilde{\beta}_i^2 \end{pmatrix} = F_{i+1}^\top v_i(t_{i+1})$.
    Find ξ_{i+1}^2 *and* Φ_{i+1}^{22} *from* $\xi_{i+1}^2 = E_i\xi_i^2 + \tilde{\beta}_i^2$, $\Phi_{i+1}^{22} = E_i\Phi_i^{22}$.
Reversal Initialisation:
  $M := B_aF_0\Phi_0 + B_bF_l\Phi_l$.
  $\mu := \beta - B_aF_0\xi_0 - B_bF_lG_{l-1}\xi_{l-1} - B_bv_{l-1}(b)$.
  Solve $M\eta = \mu$ *for* η.
For $i = l-1, l-2, ..., 0$ **DO**
  Reverse Step:
    $\xi_i^1 = D_i^{-1}(\xi_{i+1}^1 - C_i\xi_i^2 - \tilde{\beta}_i^1)$.
    $\Phi_i^{11} = D_i^{-1}\Phi_{i+1}^{11}$.
    $\Phi_i^{12} = D_i^{-1}(\Phi_{i+1}^{12} - C_i\Phi_i^{22})$.
    $y(t_i) = F_i\Phi_i\eta + F_i\xi_i$.

C++ programming language environment. This obtained routine was coupled with the existing routine **revolve** (see [39]) for the memory reduced implementation of Algorithm 3.1. Various run-time tests were performed. For a numerical test see the following Example 3.1.

Example 3.1 *This example is taken from [3]. Consider the BVP*

$$\dot{y} = \begin{bmatrix} \psi(t) & 0 \\ 2\psi(t) & -\psi(t) \end{bmatrix} y + \begin{bmatrix} (1-\psi(t))\exp(t) \\ 2\exp(t) \end{bmatrix}, \tag{3.75}$$

with $\psi(t) = 20\sin(t) + 20t\cos(t)$, $a = 0$, $b = 1$, *and the BC*

$$\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} y(a) + \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} y(b) = \begin{bmatrix} 2 \\ \exp(1) \end{bmatrix}. \tag{3.76}$$

The BVP (3.75) - (3.76) has the solution $y(t) = \begin{bmatrix} \exp(t) \\ 2\exp(t) \end{bmatrix}$, *although the ODE has modes growing like* $\exp(20t\sin(t))$ *and* $\exp(-20t\sin(t))$. *Hence, the BVP (3.75) - (3.76) is fitted into the set of BVPs to which Algorithm 3.1 can be applied. Therefore, the BVP (3.75) - (3.76) is solved using the compactification approach for a decoupled system, according to Algorithm 3.1. For the memory efficient implementation of Algorithm 3.1, the checkpointing techniques in Chapter 2 are applied. The time interval* $[a, b] = [0, 1]$ *is uniformly discretised in* l *time steps with* $l = 200, 500, 1000$. *Therefore, binomial reversal schedules, which are optimal reversal schedules for uniform cases, are applied to the implementation of Algorithm 3.1.*

Figure 3.3 illustrates the run-time requirement, needed to implement Algorithm 3.1, using binomial reversal schedules with various numbers of checkpoints. As one can see, the memory requirement, i.e. the number of available checkpoints, can be pushed far below ten before any impact on the run-time becomes apparent. Only when the number of checkpoints becomes lower than three, we observe a significant increase in run-time. These threshold results are not constant and will decline as the total number of time steps, i.e. the discretisation refinement, grows.

□

3.2.7 Reduced Compactification

Consider the linear BVP with separated BC

$$\dot{y} = A(t)y + q(t) \qquad a < t < b, \tag{3.77}$$

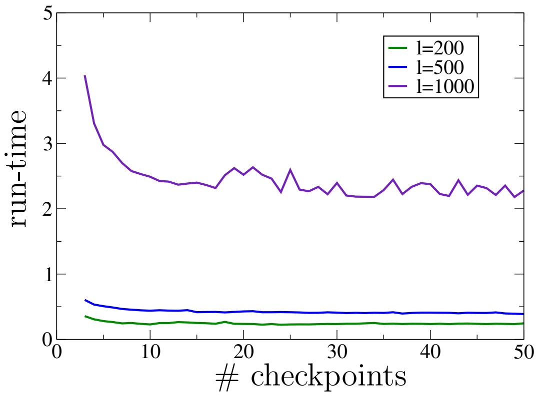

Figure 3.3: Run-time behaviour obtained from the implementation of Algorithm 3.1 for the solution of the BVP (3.75) - (3.76), using binomial reversal schedules with various numbers of checkpoints

$$\begin{bmatrix} B_{a1} \\ 0 \end{bmatrix} y(a) + \begin{bmatrix} 0 \\ B_{b2} \end{bmatrix} y(b) = \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix}, \tag{3.78}$$

with $B_{a1} \in R^{(n-k)\times n}$, $rank(B_{a1}) = n-k$ and $B_{b2} \in R^{k\times n}$, $rank(B_{b2}) = k$. Given a mesh (3.40) of shooting points, the solution $y(t)$ can be expressed in the form

$$y(t) = \bar{Y}_i(t)\bar{s}_i + v_i(t), \qquad t_i \le t \le t_{i+1}, \qquad 0 \le i \le l-1, \tag{3.79}$$

where each $\bar{Y}_i(t)$ is now an $n \times k$, not $n \times n$, matrix of fundamental solutions, satisfying

$$\dot{\bar{Y}}_i = A\dot{\bar{Y}}_i, \qquad t_i < t < t_{i+1}, \qquad \bar{Y}_i(t_i) = \bar{F}_i, \qquad 0 \le i \le l-1, \tag{3.80}$$

and each $\bar{F}_i$ is an $n \times k$ matrix or $rank\ k$. We choose $v_0(a)$ and $\bar{F}_0$, so that

$$B_{a1} v_0(a) = \beta_1, \tag{3.81}$$

$$B_{a1} \bar{F}_0 = 0. \tag{3.82}$$

To obtain these initial values, we construct at first the QU decomposition of $B_{a1}^\top$. Thus, let H be an orthogonal $n \times n$ matrix and R a lower triangular $(n-k) \times (n-k)$ matrix, so that

$$H B_{a1}^\top = \begin{bmatrix} R^\top \\ 0 \end{bmatrix}. \tag{3.83}$$

Now, let $\bar{F}_0$ be the last k columns of $H^\top$. Then, transposing (3.83) and considering the last k columns, the (3.82) will be satisfied. Writing

$$H^\top = \left[\hat{Y}(a) \mid \bar{Y}(a)\right] = \left[\hat{F}_0 \mid \bar{F}_0\right], \tag{3.84}$$

and defining

$$v_0(a) := \hat{Y}(a)R^{-1}\beta_1, \tag{3.85}$$

we see that (3.81) is satisfied as well.

For the general step i, we integrate a particular solution $v_i(t)$ and a reduced fundamental solution $\bar{Y}_i(t)$ until the next shooting point $t = t_{i+1}$. Then, we have to make sure that the new reduced fundamental solution $\bar{Y}_{i+1}(t)$ is so that $range(\bar{Y}_{i+1}(t_{i+1})) = range(\bar{Y}_i(t_{i+1}))$ and that the new particular solution $v_{i+1}(t)$ is in $range(\bar{Y}_i(t_{i+1})) + v_i(t_{i+1})$. This is done as follows: first, we orthogonalised the columns of $\bar{Y}_i(t_{i+1})$, using Householder transformation. This gives

$$\bar{Y}_i(t_{i+1}) = \bar{F}_{i+1}\bar{G}_i, \tag{3.86}$$

with $\bar{G}_i$ being an upper triangular $k \times k$ matrix and $\bar{F}_{i+1}$ being an $n \times k$ matrix having orthogonal columns. Then, we choose the initial values α_{i+1} for $v_{i+1}(t_{i+1})$ by

$$\alpha_{i+1} := (I - \bar{F}_{i+1}\bar{F}_{i+1}^\top)v_i(t_{i+1}). \tag{3.87}$$

Matching the solution segments of (3.79) to obtain a continuous $y(t)$, we have

$$\bar{Y}_i(t_{i+1})\bar{s}_i + v_i(t_{i+1}) = \bar{F}_{i+1}\bar{s}_{i+1} + \alpha_{i+1}. \tag{3.88}$$

Then, using (3.86), (3.87), and, multiplying through by $\bar{F}_{i+1}^\top$, yields

$$\bar{s}_{i+1} = \bar{G}_i\bar{s}_i + \bar{F}_{i+1}^\top v_i(t_{i+1}), \qquad 0 \le i \le l-2. \tag{3.89}$$

The BC now read

$$B_{b2}\left[\bar{Y}_{l-1}(b)\bar{s}_{l-1} + v_{l-1}(b)\right] = \beta_2. \tag{3.90}$$

The relations (3.89) and (3.90) give a linear system of order kl for $\bar{s}^\top = (\bar{s}_0^\top, ..., \bar{s}_{l-1}^\top)$,

$$\bar{\mathcal{A}}\bar{s} = \bar{\beta}, \tag{3.91}$$

with

$$
\bar{\mathcal{A}} = \begin{bmatrix} -\bar{G}_0 & I & & & \\ & -\bar{G}_1 & I & & \\ & . & . & . & \\ & & & -\bar{G}_{l-2} & I \\ & & & & B_{b2}\bar{F}_l\bar{G}_{l-1} \end{bmatrix}. \tag{3.92}
$$

The construction of the right-hand-side vector $\bar{\beta}$ is straightforward.

In the (3.92) we recognise a system like (3.50) and (3.64), but of a reduced order. Hence, the implementation is more efficient with respect to the number of IVPs to be integrated and with respect to the algebraic system to be solved. $\bar{\mathcal{A}}$ in (3.91) and (3.92) is a simple block upper triangular matrix, suggesting a computation of $\bar{s}$ by block back-substitution. The orthogonalisation process here may be viewed as a special case of the process described in Section 3.2.5. The Stabilised March Algorithm takes the following form.

Algorithm 3.2 Stabilised March
Initialisation:
Determine $\bar{F}_0$ *by* $B_{a1}\bar{F}_0 = 0$.
Determine α_0 *by* $B_{a1}\alpha_0 = \beta_1$.
For $i = 0, 1, ..., l-1$ **DO**
  Original Step:
    Solve $\dot{\bar{Y}}_i = A\bar{Y}_i, \qquad t_i < t < t_{i+1}, \qquad \bar{Y}_i(t_i) = \bar{F}_i$.
    Solve $\dot{v}_i = Av_i + q_i, \qquad t_i < t < t_{i+1}, \qquad v_i(t_i) = \alpha_i$.
    Obtain $\bar{F}_{i+1}$ *from* $\bar{Y}_i(t_{i+1}) = \bar{F}_{i+1}\bar{G}_i$.
    Find $\alpha_{i+1} = (I - \bar{F}_{i+1}\bar{F}_{i+1}^{\top})v_i(t_{i+1})$.
Reversal Initialisation:
  Solve $B_{b2}\left[\bar{Y}_{l-1}(b)\bar{s}_{l-1} + v_{l-1}(b)\right] = \beta_2$ *for* $\bar{s}_{l-1}$.
  Find $y(t_{l-1}) = \bar{F}_{l-1}\bar{s}_{l-1} + \alpha_{l-1}$.
For $i = l-2, l-3, ..., 0$ **DO**
  Reverse Step:
    Solve $\bar{s}_{i+1} = \bar{G}_i\bar{s}_i + \bar{F}_{i+1}^{\top}v_i(t_{i+1})$ *for* $\bar{s}_i$.
    Find $y(t_i) = \bar{F}_i\bar{s}_i + \alpha_i$.

From Algorithm 3.2 we can see that the stabilised march consists of two sweeps through the time window $[a, b]$. Therefore, for a memory efficient implementation of Algorithm 3.2 checkpointing techniques from Chapter 2 can be applied. The routine implementing Algorithm 3.2 was coded using the C++ programming language environment. This obtained routine was coupled with the existing routine **revolve** (see [39]) for the memory reduced implementation of Algorithm 3.2. Various run-time tests were performed. For a numerical test see Example 3.2.

Example 3.2 *This example is taken from [1]. Consider the BVP*

$$\dot{y} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{t}{\epsilon} \end{bmatrix} y + \begin{bmatrix} 0 \\ \frac{-\epsilon\pi^2 \cos(\pi t) - \pi t \sin(\pi t)}{\epsilon} \end{bmatrix}, \tag{3.93}$$

with $erf(t) = \frac{2}{\sqrt{\pi}} \int_0^t e^{-v^2} dv$, $a = -1$, $b = 1$, *and the BC*

$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} y(a) + \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} y(b) = \begin{bmatrix} -2 \\ \frac{\sqrt{2}\, e^{-\frac{1}{2\epsilon}}}{\sqrt{\pi\epsilon}\, erf(\frac{1}{\sqrt{2\epsilon}})} \end{bmatrix}. \tag{3.94}$$

The BVP has the solution

$$y(t) = \begin{bmatrix} \cos(\pi t) + \frac{erf(t/\sqrt{2\epsilon})}{erf(1/\sqrt{2\epsilon})} \\ -\pi \sin(\pi t) + \frac{\sqrt{2}\, e^{-(t/\sqrt{2\epsilon})^2}}{\sqrt{\pi\epsilon}\, erf(1/\sqrt{2\epsilon})} \end{bmatrix}.$$

The solution has a shock layer in the turning point region near $t = 0$ (see Figure 3.4). The BVP (3.93) - (3.94) is solved using stabilised march for

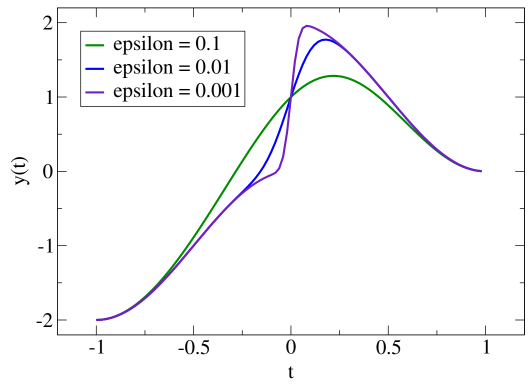

Figure 3.4: $y(t)$ versus t for various ϵ

decoupled systems, according to Algorithm 3.2. For the memory efficient implementation of Algorithm 3.2, the checkpointing techniques in Chapter 2 are *applied. The time interval* $[a, b] = [-1, 1]$ *is uniformly discretised in* l *time steps with* $l = 200, 500, 1000$*. Therefore, binomial reversal schedules, being optimal reversal schedules for uniform cases, are applied to the implementation of Algorithm 3.2.*

Figure 3.5 illustrates the run-time requirement, needed to implement Algorithm 3.2, using binomial reversal schedules with various numbers of checkpoints.

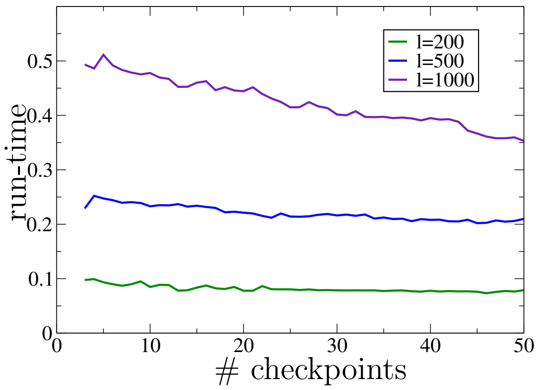

Figure 3.5: Run-time behaviour obtained from the implementation of Algorithm 3.2 for the solution of the BVP (3.93) - (3.94), using binomial reversal schedules with various numbers of checkpoints

The results are similar to Figure 3.3 in Example 3.1, namely, the memory requirement, i.e. the number of available checkpoints, can be pushed far below ten before any impact on the run-time becomes apparent. Only when the number of checkpoints becomes lower than three, we observe a significant increase in run-time. These threshold results are not constant and will decline as the total number of time steps, i.e. the discretisation refinement, grows. This implies that the utilisation of checkpointing techniques for this example results in a slight increase of run-time if the number of checkpoints is less than ten. At the same time, the memory requirement is reduced drastically, compared to the approach where all intermediate states are stored.

□

3.2.8 Riccati Transformation for Linear BVPs

The common feature to all of the methods discussed so far is that the discretisation was first applied to the original BVP. Then, some transformations were applied to obtain a stable solution. The method, which will be discussed in the present section, is related to what has been called **invariant embedding**. Its name is the **Riccati method**. Here, the ODEs are transformed at first. Then, the discretisation is performed. What is to be achieved is that the transformed system of ODEs contains IVPs which are stable. Hence, they can be stably and efficiently integrated.

For a given ODE

$$\dot{y} = A(t)y + q(t), \tag{3.95}$$

let $T(t)$ be a linear transformation of the form

$$T(t) = \begin{pmatrix} I & 0 \\ K(t) & I \end{pmatrix}, \tag{3.96}$$

with $K(t)$ being an $(n-k) \times k$ matrix. $K(t)$ will be defined later. Define

$$w(t) := T^{-1}(t)y(t). \tag{3.97}$$

Then, $w(t)$ satisfies

$$\dot{w} = U(t)w + g(t). \tag{3.98}$$

$U(t)$ and $T(t)$ satisfy the Lyapunov equation

$$\dot{T} = AT - TU, \tag{3.99}$$

and

$$g(t) := T^{-1}(t)q(t). \tag{3.100}$$

Now, we require that U be 'block upper triangular' in the sense that its lower left block, corresponding in dimension $K(t)$, is zero. Thus, we introduce decoupling. We write

$$U(t) = \begin{pmatrix} U^{11}(t) & U^{12}(t) \\ U^{21}(t) & U^{22}(t) \end{pmatrix}, \qquad g(t) = \begin{pmatrix} g^1(t) \\ g^2(t) \end{pmatrix}, \tag{3.101}$$

with U^{11} and U^{22} square submatrices, and use a similar partition notation for other matrices in the sequel. Our requirement for $U(t)$ is that $U^{21}(t) = 0$.

This yields, upon substitution in (3.99), the **Riccati differential equation** for $K(t)$,

$$\dot{K} = A^{21} + A^{22}K - KA^{11} - KA^{12}K, \tag{3.102}$$

and the block form for $U(t)$

$$U(t) = \begin{pmatrix} A^{11} + A^{12}K & A^{12} \\ 0 & A^{22} - KA^{12} \end{pmatrix}. \tag{3.103}$$

The transformed ODEs (3.98) can be written in the decoupled form

$$\dot{w}^1 = U^{11}(t)w^1 + U^{12}(t)w^2 + g^1(t), \tag{3.104}$$

$$\dot{w}^2 = U^{22}(t)w^2 + g^2(t). \tag{3.105}$$

3.2.9 Riccati Transformation for Discretised BVPs

So far, dichotomy and decoupling are discussed at the ODE level. Now, consider the same concepts at the discrete level. Consider a one-step recursion of the form

$$R_i y_{i+1} = S_i y_i + q_i, \qquad 0 \le i \le l-1, \tag{3.106}$$

$$B_a y_0 + B_b y_l = \beta. \tag{3.107}$$

The recursion (3.106) - (3.107) arises when one is discretising a BVP. Hence, we refer to (3.106) - (3.107) as a **discrete BVP**. Suppose that both R_i and S_i are nonsingular and well-conditioned matrices.

An analogue to the continuous Lyapunov equation (3.99) reads in the discrete case

$$S_i T_i = R_i T_{i+1} U_i, \qquad 0 \le i \le l-1, \tag{3.108}$$

with T_i required to be nonsingular and U_i required to be block upper triangular. Introducing a transformed variable

$$w_i := T_i^{-1} y_i, \qquad 0 \le i \le l, \tag{3.109}$$

we find the recursion (3.106) - (3.107) reads after transformation

$$w_{i+1} = U_i w_i + (R_i T_{i+1})^{-1} q_i, \qquad 0 \le i \le l-1, \tag{3.110}$$

$$B_a T_0 w_0 + B_b T_l w_l = \beta. \tag{3.111}$$

Once we have found transformation matrices T_i, a decoupled recursion can be defined. Consequently, we can write the (3.110) - (3.111) in the following form

$$w_{i+1}^1 = U_i^{11} w_i^1 + U_i^{12} w_i^2 + g_i^1, \tag{3.112}$$

$$w_{i+1}^2 = U_i^{22} w_i^2 + g_i^2, \tag{3.113}$$

with g_i defined by

$$g_i = T_{i+1}^{-1} R_i^{-1} q_i. \tag{3.114}$$

In the discrete case, there are many ways to achieve decoupling (note that the first k columns of T_i have to span the growing modes of the BVP). We should factorise the matrix $S_i T_i$ into a product of two matrices of which the second one is block upper triangular. For simplicity of notation, we take

$$R_i = I. \tag{3.115}$$

Otherwise, we can multiply (3.106) by R_i^{-1} for each i.

One way to proceed is the following. Since S_i is nonsingular, we may perform an LU-decomposition of $S_i T_i$, i.e.

$$S_i T_i = L_i U_i. \tag{3.116}$$

Then, to satisfy (3.108) we set

$$T_{i+1} := L_i. \tag{3.117}$$

Note that T_{i+1} is a lower triangular nonsingular matrix. As usual, an advantage of an LU-decomposition is its speed, while a disadvantage is its poor worst-case stability bounds.

Another variant is to perform a QU-decomposition. Once we have computed

$$S_i T_i = Q_i U_i, \tag{3.118}$$

then, T_{i+1} is defined by

$$T_{i+1} := Q_i. \tag{3.119}$$

The main advantage of this extensive variant is that T_{i+1} is orthogonal and, hence, ideally conditioned.

Another approach to achieve decoupling is to straightforwardly integrate the discrete Riccati equation (3.108). One more time we write the initial recursion in the form

$$\begin{pmatrix} y_{i+1}^1 \\ y_{i+1}^2 \end{pmatrix} = \begin{pmatrix} S_i^{11} & S_i^{12} \\ S_i^{21} & S_i^{22} \end{pmatrix} \begin{pmatrix} y_i^1 \\ y_i^2 \end{pmatrix} + \begin{pmatrix} q_i^1 \\ q_i^2 \end{pmatrix}. \tag{3.120}$$

Introducing a transformed variable w_i in (3.109) with $T_i = \begin{pmatrix} I & 0 \\ K_i & I \end{pmatrix}$, we obtain

$$\begin{pmatrix} I & 0 \\ K_{i+1} & I \end{pmatrix} \begin{pmatrix} w_{i+1}^1 \\ w_{i+1}^2 \end{pmatrix} = \begin{pmatrix} S_i^{11} & S_i^{12} \\ S_i^{21} & S_i^{22} \end{pmatrix} \begin{pmatrix} I & 0 \\ K_i & I \end{pmatrix} \begin{pmatrix} w_i^1 \\ w_i^2 \end{pmatrix} + \begin{pmatrix} q_i^1 \\ q_i^2 \end{pmatrix}, \tag{3.121}$$

or

$$\begin{pmatrix} w_{i+1}^1 \\ K_{i+1}w_{i+1}^1 + w_{i+1}^2 \end{pmatrix} = \begin{pmatrix} S_i^{11} + S_i^{12}K_i & S_i^{12} \\ S_i^{21} + S_i^{22}K_i & S_i^{22} \end{pmatrix} \begin{pmatrix} w_i^1 \\ w_i^2 \end{pmatrix} + \begin{pmatrix} q_i^1 \\ q_i^2 \end{pmatrix}. \tag{3.122}$$

After substituting the expression for w_{i+1}^1 in the lower equation of (3.122) we obtain

$$\begin{pmatrix} w_{i+1}^1 \\ w_{i+1}^2 \end{pmatrix} = \begin{pmatrix} S_i^{11} + S_i^{12}K_i & S_i^{12} \\ U^{21} & S_i^{22} - K_{i+1}S_i^{12} \end{pmatrix} \begin{pmatrix} w_i^1 \\ w_i^2 \end{pmatrix} + \begin{pmatrix} q_i^1 \\ q_i^2 - K_{i+1}q_i^1 \end{pmatrix}, \tag{3.123}$$

with $U^{21} = S_i^{21} + S_i^{22}K_i - K_{i+1}(S_i^{11} + S_i^{12}K_i)$. Obviously, the recursion (3.123) corresponds to the recursion (3.110). From the requirement $U_i^{21} = 0$ we obtain the discrete case of the Riccati equation (3.102) in the form

$$S_i^{21} + S_i^{22}K_i - K_{i+1}(S_i^{11} + S_i^{12}K_i) = 0. \tag{3.124}$$

Now, we write the discrete Riccati transformation in a matrix form. Firstly, we can write a discrete BVP (3.106) - (3.107) in this form:

$$Sy = \begin{pmatrix} -S_0 & R_0 & 0 & & & \\ 0 & -S_1 & R_1 & 0 & & \\ & & . & . & & \\ & & & 0 & -S_{l-1} & R_{l-1} \\ B_a & & & & 0 & B_b \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{l-1} \\ y_l \end{pmatrix} = \begin{pmatrix} q_0 \\ q_1 \\ \vdots \\ q_{l-1} \\ \beta \end{pmatrix} = q. \tag{3.125}$$

Then, introducing a transformed variable $w_i,\ 0 \le i \le l$, in (3.109), we obtain
(3.126)

$$w = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_{l-1} \\ w_l \end{pmatrix} = \begin{pmatrix} T_0^{-1} & 0 & & & \\ 0 & T_1^{-1} & 0 & & \\ & & \cdot & \cdot & \\ & & 0 & T_{l-1}^{-1} & 0 \\ & & & 0 & T_l^{-1} \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{l-1} \\ y_l \end{pmatrix} = T^{-1} y,$$

or

$$y = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{l-1} \\ y_l \end{pmatrix} = \begin{pmatrix} T_0 & 0 & & & \\ 0 & T_1 & 0 & & \\ & & \cdot & \cdot & \\ & & 0 & T_{l-1} & 0 \\ & & & 0 & T_l \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_{l-1} \\ w_l \end{pmatrix} = Tw. \tag{3.127}$$

After the substitution of (3.127) into (3.125) we obtain

$$STw = q, \tag{3.128}$$

or
(3.129)

$$\begin{pmatrix} -S_0T_0 & R_0T_1 & 0 & & \\ 0 & -S_1T_1 & R_1T_2 & 0 & \\ & & \cdot & \cdot & \\ & & 0 & -S_{l-1}T_{l-1} & R_{l-1}T_l \\ B_a & & & 0 & B_bT_l \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_{l-1} \\ w_l \end{pmatrix} = \begin{pmatrix} q_0 \\ q_1 \\ \vdots \\ q_{l-1} \\ \beta \end{pmatrix}.$$

Therefore, the recursion for $w_i,\ 0 \le i \le l-1$, can be written in the form

$$R_iT_{i+1}w_{i+1} = S_iT_iw_i + q_i, \tag{3.130}$$

or

$$w_{i+1} = (R_iT_{i+1})^{-1}S_iT_iw_i + (R_iT_{i+1})^{-1}q_i, \tag{3.131}$$

with $(R_iT_{i+1})^{-1}S_iT_i = U_i$ a discrete case of the Lyapunov equation (recall (3.108)). Therefore, U_i has the desired form $U_i = \begin{pmatrix} U_i^{11} & U_i^{12} \\ 0 & U_i^{22} \end{pmatrix}$, provided the transformation matrix T_i is evaluated using one of the methods described above. Thus, the recursion (3.131) is decoupled.

It is left to be shown how to straightforwardly integrate the discrete Riccati equation (3.124). Provided that the BC are separated, i.e.

$$\begin{bmatrix} B_{a1} \\ 0 \end{bmatrix} y(a) + \begin{bmatrix} 0 \\ B_{b2} \end{bmatrix} y(b) = \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix}. \tag{3.132}$$

B_{a1} controls the last $(n-k)$ decreasing modes of $y(t)$, B_{b2} controls the first k increasing modes of $y(t)$. Now, we describe how to choose the initial values T_0, w_0^2, and w_l^1. Using (3.132), we obtain

$$B_{a1} y_0 = \beta_1, \qquad B_{b2} y_l = \beta_2, \tag{3.133}$$

with

$$B_{a1} = (C \mid D). \tag{3.134}$$

D is a nonsingular $(n-k) \times (n-k)$ matrix. Using (3.109), w_0^2 can be defined by

$$w_0^2 = -K_0 y_0^1 + y_0^2. \tag{3.135}$$

Therefore, the choice

$$K_0 := -D^{-1} C \tag{3.136}$$

yields

$$w_0^2 = D^{-1} \beta_1. \tag{3.137}$$

Then, $K_{i+1},\ 0 \le i \le l-1$, and $w_{i+1}^2,\ 0 \le i \le l-1$, can be determined by (3.124) and (3.113), respectively. After finding K_l and w_l^2, we evaluate w_l^1 using (3.133) in the form

$$B_{b2} T_l w_l = \beta_2. \tag{3.138}$$

Then, $w_i^1,\ 0 \le i \le l-1$, can be determined by (3.112). This approach leads to the following conceptual algorithm (see Algorithm 3.3).

From Algorithm 3.3 we can see that the Riccati method consists of two sweeps through the time window $[a, b]$. Therefore, for a memory efficient implementation of Algorithm 3.3 checkpointing techniques from Chapter 2 can be applied. The routine implementing Algorithm 3.3 was coded using the C++ programming language environment. This obtained routine was coupled with the existing routine **revolve** (see [39]) for the memory reduced implementation of Algorithm 3.3. Various run-time tests were performed. For a numerical test see the following Example 3.3.

Algorithm 3.3 Riccati Method
Initialisation:
Determine K_0 *by* $K_0 := -D^{-1}C$.
Determine w_0^2 *by* $w_0^2 = D^{-1}\beta_1$.
For $i = 0, 1, ..., l-1$ **DO**
  Original Step:
    Evaluate T_{i+1} *and* U_i *using one of the available methods.*
    Evaluate $w_{i+1}^2 = U_i^{22} w_i^2 + g_i^2$.
Reversal Initialisation:
  Solve $B_{b2} T_l w_l = \beta_2$ *for* w_l^1.
  Evaluate $y_l = T_l w_l$.
For $i = l-1, l-2, ..., 0$ **DO**
  Reverse Step:
    Solve $w_{i+1}^1 = U_i^{11} w_i^1 + U_i^{12} w_i^2 + g_i^1$ *for* w_i^1.
    Evaluate $y_i = T_i w_i$.

Example 3.3 *This example is taken from [1]. Consider the BVP*

$$\dot{y} = \begin{bmatrix} 0 & 1 \\ 1/\epsilon & t/\epsilon \end{bmatrix} y + \begin{bmatrix} 0 \\ -((1+\epsilon\pi^2)\cos(\pi t) + \pi t \sin(\pi t))/\epsilon \end{bmatrix}, \tag{3.139}$$

with $a = -1$, $b = 1$, *and the BC*

$$\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} y(a) + \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} y(b) = \begin{bmatrix} 0 \\ -1 \end{bmatrix}. \tag{3.140}$$

The BVP has a solution

$$y(t) = \begin{bmatrix} \cos(\pi t) \\ -\pi \sin(\pi t) \end{bmatrix}.$$

The problem has a turning point at $t = 0$ *but the solution is smooth (see Figure 3.6). The BVP (3.139) - (3.140) is solved using the Riccati method, according to Algorithm 3.3. For the memory efficient implementation of Algorithm 3.3, the checkpointing techniques from Chapter 2 are applied. The time interval* $[a, b] = [-1, 1]$ *is uniformly discretised in* l *time steps with* $l = 200, 500, 1000$. *Therefore, binomial reversal schedules, which are optimal reversal schedules for uniform cases, are applied to the implementation of Algorithm 3.3. In order to use Algorithm 3.3, i.e. to obtain a discrete one-step recursion of the form (3.106), the BVP (3.139) - (3.140) has to be discretised applying for instance the* **trapezoidal** *scheme*

$$\frac{y_{i+1} - y_i}{h_i} = \frac{1}{2}[A(t_{i+1})y_{i+1} + A(t_i)y_i] + \frac{1}{2}[q(t_{i+1}) + q(t_i)], \qquad 0 \le i \le l-1.$$

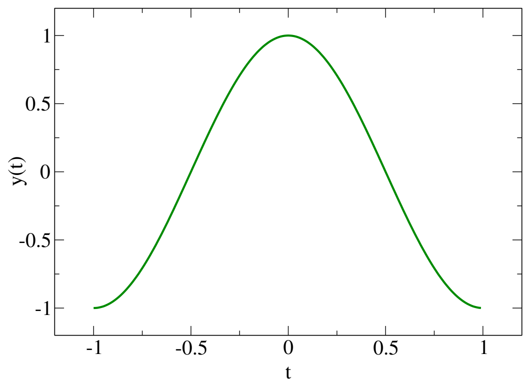

Figure 3.6: $y(t)$ versus t for all ϵ

Figure 3.7 illustrates the run-time requirement, needed to implement Algorithm 3.3 using binomial reversal schedules with various numbers of checkpoints.

As one can see in Figure 3.7, the utilisation of checkpointing techniques for this example does not result in a increase of run-time. At the same time, the memory requirement is reduced drastically, compared to the approach where all intermediate states are stored.

□

Example 3.3 is an instance where the Riccati transformation is a useful tool for decoupling increasing from decreasing modes for BVPs with separated BC. The resulting system of ODEs is so that the IVPs can be stably and efficiently integrated.

3.3 Initial Value Methods for Non-Linear Problems

In this section we consider a general non-linear BVP

$$\dot{y} = f(t, y), \qquad a < t < b, \tag{3.141}$$

$$g(y(a), y(b)) = 0. \tag{3.142}$$

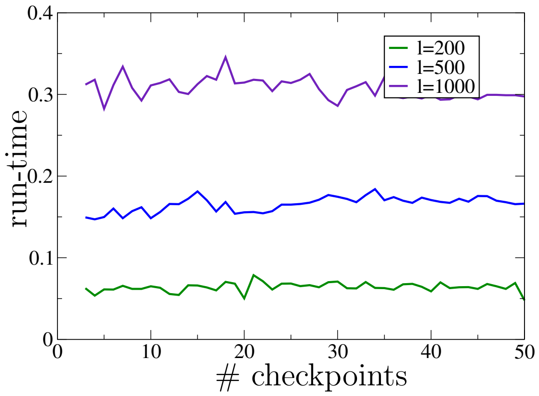

Figure 3.7: Run-time behaviour obtained from the implementation of Algorithm 3.3 for the solution of the BVP (3.139) - (3.140), using binomial reversal schedules with various numbers of checkpoints

If we apply single or multiple shooting discretisation to (3.141) - (3.142), we obtain a non-linear system of algebraic equations. Newton's method can be applied to solve these equations. At each Newton iteration we use the linear methods considered in the previous sections. Single shooting in the non-linear case has similar drawbacks as one in the linear case, but multiple shooting avoids them here, as well as it was in the linear case.

3.3.1 Single Shooting for Non-Linear Problems

For any initial vector s, let $y(t, s)$ satisfy

$$\frac{dy}{dt}(t,s) = f(t, y(t,s)), \qquad a < t < b, \tag{3.143}$$

$$y(a, s) = s. \tag{3.144}$$

The required $s = s^*$ is that for which the BC (3.142) are satisfied, i.e.

$$g(s^*, y(b, s^*)) = 0. \tag{3.145}$$

In general, s^* is not known. Therefore, we may try to find it by an iterative scheme like Newton's method. We have to obtain a solution of

$$F(s) = 0, \tag{3.146}$$

with

$$F(s) := g(s, y(b, s)). \tag{3.147}$$

Given an initial guess s^0, Newton's method generates a sequence of iterates $s^1, s^2, ..., s^j, ...,$ converging to a solution s^* of (3.146) by

$$s^{j+1} := s^j + \xi, \tag{3.148}$$

where ξ solves the linear system

$$\dot{F}(s^j)\xi = -F(s^j). \tag{3.149}$$

$\dot{F}$ is the $n \times n$ Jacobian matrix defined by

$$\dot{F}(s) = \frac{\partial F(s)}{\partial s}. \tag{3.150}$$

Using (3.147), we obtain

$$\dot{F}(s) = B_a + B_b Y(b), \tag{3.151}$$

with

$$B_a = \frac{\partial g(u,v)}{\partial u}, \quad B_b = \frac{\partial g(u,v)}{\partial v} \; at \; u = s, \; v = y(b,s), \tag{3.152}$$

and $Y(t) = \frac{\partial y(t,s)}{\partial s}$ is the $n \times n$ fundamental solution defined by

$$\dot{Y} = A(t)Y, \qquad a < t < b, \tag{3.153}$$

$$Y(a) = I, \tag{3.154}$$

with

$$A(t) = A(t,s) \equiv \frac{\partial f}{\partial y}(t, y(t,s)). \tag{3.155}$$

In general, B_a, B_b, and $Y(t)$ all depend on s. In actual computations, $y(t,s)$ and $Y(t)$ are replaced by their numerical approximations obtained by solving the initial value problems.

3.3.2 Multiple Shooting for Non-Linear Problems

We subdivide the interval $[a, b]$ by a mesh $a = t_0 < t_1 < ... < t_l = b$ and consider the initial value problems for $0 \le i \le l-1$,

$$\dot{y} = f(t, y), \qquad t_i < t < t_{i+1}, \tag{3.156}$$

$$y(t_i) = s_i. \tag{3.157}$$

The solution of (3.156) - (3.157) is denoted $y_i(t, s_i)$.

The unknowns to be found are the nl parameters

$$s^\top = (s_0^\top, s_1^\top, ..., s_{l-1}^\top). \tag{3.158}$$

They are to be determined, so that the solution is continuous over the entire interval $[a, b]$. Moreover, the boundary conditions (3.142) are satisfied. Thus, the desired solution is defined by

$$y(t) := y_i(t, s_i), \qquad t_i \le t \le t_{i+1}, \qquad 0 \le i \le l-1, \tag{3.159}$$

where the requirements on s are

$$y_i(t_{i+1}, s_i) = s_{i+1}, \qquad 0 \le i \le l-2, \tag{3.160}$$

$$g(s_0, y_{l-1}(b, s_{l-1})) = 0. \tag{3.161}$$

So, defining

$$F(s) = \begin{bmatrix} s_1 - y_0(t_1, s_0) \\ s_2 - y_1(t_2, s_1) \\ . \\ . \\ s_{l-1} - y_{l-2}(t_{l-1}, s_{l-2}) \\ g(s_0, y_{l-1}(b, s_{l-1})) \end{bmatrix}, \tag{3.162}$$

we are left to solve a set of nl non-linear algebraic equations

$$F(s) = 0. \tag{3.163}$$

Suppose that Newton's method is used to solve (3.163). The process is exactly the same as described for single shooting in (3.148) - (3.150), except that the matrix $\dot{F}(s)$ now has the form similar to $\mathcal{A}$ in (3.50), i.e.

$$\dot{F}(s) = \begin{bmatrix} -Y_0(t_1) & I & & & \\ & -Y_1(t_2) & I & & \\ & & . & . & \\ & & & -Y_{l-2}(t_{l-1}) & I \\ B_a & & & & B_b Y_{l-1}(b) \end{bmatrix}. \tag{3.164}$$

Here, $Y_i = Y_i(t)$ is the $n \times n$ fundamental solution defined by

$$\dot{Y} = A(t)Y_i, \qquad t_i < t < t_{i+1}, \tag{3.165}$$

$$Y_i(t_i) = I, \qquad 0 \leq i \leq l-1, \tag{3.166}$$

with $A(t) = A(t,s)$, $B_a = B_a(s)$, and $B_b = B_b(s)$ defined, as in single shooting, by (3.152) - (3.155).

Compactification, marching techniques, and Riccati transformation can be applied to non-linear problems as well. For the numerical implementation we use one of available routines implementing Algorithm 3.1, Algorithm 3.2, or Algorithm 3.3 coupled with the existing routine **revolve** (see [39]) for the memory reduced implementation of the corresponding algorithm. Various run-time tests were performed. For a numerical test see the following Example 3.4.

Example 3.4 *This example is taken from [1]. Consider the BVP*

$$\dot{y} = \begin{bmatrix} y_2 \\ \frac{y_1 + y_1^2 - e^{-\frac{2t}{\sqrt{\epsilon}}}}{\epsilon} \end{bmatrix} = f(y,t), \qquad 0 < t < 1, \tag{3.167}$$

and the BC

$$\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} y(a) + \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} y(b) = \begin{bmatrix} -\frac{1}{\sqrt{\epsilon}} \\ e^{-\frac{1}{\sqrt{\epsilon}}} \end{bmatrix}. \tag{3.168}$$

The matrices $A(t) = A(t,s)$, $B_a = B_a(s)$, and $B_b = B_b(s)$ are defined by

$$A(t) = \begin{bmatrix} 0 & 1 \\ \frac{1+2y_1}{\epsilon} & 0 \end{bmatrix}, \qquad B_a = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \qquad B_b = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}.$$

The BVP (3.167) - (3.168) has the solution $y(t) = e^{-\frac{t}{\sqrt{\epsilon}}}$. The solution has a boundary layer of width $\mathcal{O}(\sqrt{\epsilon})$ at $t = 0$ (see Figure 3.8).

The BVP (3.167) - (3.168) is solved using Riccati method, according to Algorithm 3.3. For the memory efficient implementation of Algorithm 3.3, the checkpointing techniques from Chapter 2 are applied. The time interval $[a,b] = [0,1]$ is uniformly discretised in l time steps with $l = 200, 500, 1000$. Therefore, binomial reversal schedules, which are optimal reversal schedules for uniform cases, are applied to the implementation of Algorithm 3.3.

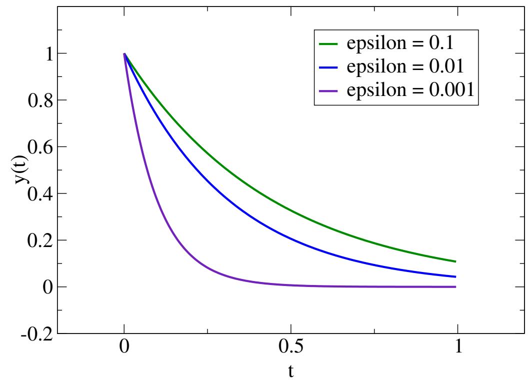

Figure 3.8: $y(t)$ versus t for various ϵ

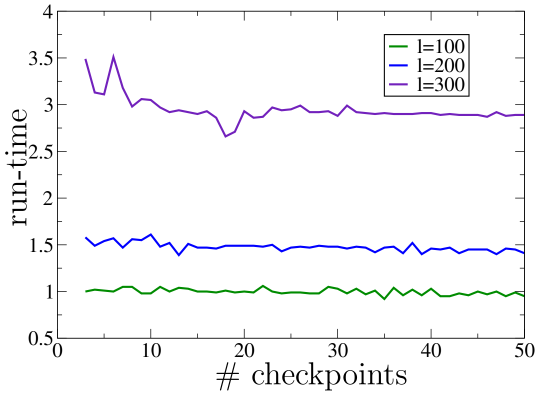

Figure 3.9: Run-time behaviour obtained from the implementation of Algorithm 3.3 for the solution of the non-linear BVP (3.167) - (3.168), using binomial reversal schedules with various numbers of checkpoints

Figure 3.9 illustrates the run-time requirement, needed to implement Algorithm 3.3, using binomial reversal schedules with various numbers of checkpoints.

The results are similar to Figure 3.3 in Example 3.1, namely, the memory requirement, i.e. the number of available checkpoints, can be pushed far below ten before any impact on the run-time becomes apparent. Only when the number of checkpoints becomes lower than three, we observe a significant increase in run-time. These threshold results are not constant and will decline as the total number of time steps, i.e. the discretisation refinement, grows. This implies that the utilisation of checkpointing techniques for this example results in a slight increase of run-time if the number of checkpoints is less than ten. At the same time, the memory requirement is reduced drastically, compared to the approach where all intermediate states are stored.

□

3.3.3 Example for Solving an Optimal Control Problem Using Multiple Shooting Method

The present section gives an example for solving an optimal control problem using multiple shooting method. Recall an optimal control problem of an initial value ordinary differential equation

$$\min_{\mathbf{x},\mathbf{u}} \phi(\mathbf{x}(T)), \tag{3.169}$$

where the system is described by

$$\dot{\mathbf{x}} = f(\mathbf{x}(t), \mathbf{u}(t), t), \quad \mathbf{x}(0) = \mathbf{x}_0. \tag{3.170}$$

A corresponding BVP, obtained from the Euler-Lagrange equations for the optimal control problem (3.169) - (3.170), is

$$\dot{\mathbf{x}} = f(\mathbf{x}(t), \mathbf{u}(t), t), \quad \mathbf{x}(0) = \mathbf{x}_0, \tag{3.171}$$

$$\dot{\bar{\mathbf{x}}} = -f_{\mathbf{x}}^{\top}(\mathbf{x}(t), \mathbf{u}(t), t)\, \bar{\mathbf{x}}, \quad \bar{\mathbf{x}}(T) = \phi_{\mathbf{x}}^{\top}(\mathbf{x}(T)), \tag{3.172}$$

$$\left(\frac{\partial H}{\partial \mathbf{u}}\right)^{\top} = f_{\mathbf{u}}^{\top}(\mathbf{x}(t), \mathbf{u}(t), t)\, \bar{\mathbf{x}} = 0, \qquad 0 \le t \le T, \tag{3.173}$$

with

$$H(\mathbf{x}(t), \mathbf{u}(t), \bar{\mathbf{x}}(t), t) = \bar{\mathbf{x}}^{\top}(t)\, f(\mathbf{x}(t), \mathbf{u}(t), t). \tag{3.174}$$

Suppose, we can express $\mathbf{u}(t)$ from the first order necessary condition (3.173), i.e. $\mathbf{u} = \mathbf{u}(\mathbf{x}(t), \bar{\mathbf{x}}(t), t)$, and substitute it into (3.171) - (3.172). Denoting $f(\mathbf{x}(t), \mathbf{u}(\mathbf{x}(t), \bar{\mathbf{x}}(t), t), t) = \tilde{f}(\mathbf{x}(t), \bar{\mathbf{x}}(t), t)$, we obtain the following BVP

$$\dot{\mathbf{x}} = f(\mathbf{x}(t), \bar{\mathbf{x}}(t), t), \quad \mathbf{x}(0) = \mathbf{x}_0, \tag{3.175}$$

$$\dot{\bar{\mathbf{x}}} = -\tilde{f}_{\mathbf{x}}^{\top}(\mathbf{x}(t), \bar{\mathbf{x}}(t), t)\, \bar{\mathbf{x}}, \quad \bar{\mathbf{x}}(T) = \phi_{\mathbf{x}}^{\top}(\mathbf{x}(T)). \tag{3.176}$$

The BVP (3.175) - (3.176) can be solved using multiple shooting techniques in Section 3.3.2. Firstly, we rewrite the BVP (3.175) - (3.176) and its BC in the following form

$$\dot{y} = \psi(y, t), \qquad 0 < t < T, \tag{3.177}$$

$$\begin{bmatrix} \mathbf{x}(0) - \mathbf{x}_0 \\ \bar{\mathbf{x}}(T) - \phi_{\mathbf{x}}^{\top}(\mathbf{x}(T)) \end{bmatrix} = 0, \tag{3.178}$$

with $y(t) = \begin{bmatrix} \mathbf{x}(t) \\ \bar{\mathbf{x}}(t) \end{bmatrix} \in \mathbf{R}^{2n}$, $\psi = \begin{bmatrix} f \\ -\tilde{f}_{\mathbf{x}}^{\top}\bar{\mathbf{x}} \end{bmatrix}$.

The BVP (3.177) - (3.178) contains both fast-growing and fast-decaying modes. This is because the primal problem (3.175) is stable by integrating in the forward direction over the subspace spanned by $\mathbf{x}(t)$, the adjoint problem (3.176) is stable by integrating in the reverse direction over the subspace spanned by $\bar{\mathbf{x}}(t)$. The BC (3.178) control the n fast-decaying modes at the beginning, and n fast-growing modes at the end of the time interval $[0, T]$. Therefore, one of the decoupling methods from Section 3.2.2 can be used for a stable integration of the BVP (3.177) - (3.178). For instance, reduced compactification techniques (see Section 3.2.7) can be applied.

$A(t)$, B_a, and B_b, corresponding to (3.152) and (3.155), take the form

$$A(t) = \begin{bmatrix} f_{\mathbf{x}} & f_{\bar{\mathbf{x}}} \\ -\tilde{f}_{\mathbf{xx}}^{\top}\bar{\mathbf{x}} & -[\tilde{f}_{\mathbf{x}}^{\top} + \tilde{f}_{\mathbf{x}\bar{\mathbf{x}}}^{\top}\bar{\mathbf{x}}] \end{bmatrix}, \tag{3.179}$$

$$B_a = \begin{bmatrix} I_n & 0 \\ 0 & 0 \end{bmatrix} \qquad B_b = \begin{bmatrix} 0 & 0 \\ -\phi_{\mathbf{xx}}^{\top} & I_n \end{bmatrix}. \tag{3.180}$$

The interval of integration $[0, T]$ is subdivided by the mesh

$$0 = t_0 < t_1 < \ldots < t_{l-1} < t_l = T. \tag{3.181}$$

Thus, the desired solution is defined by

$$(3.182) \qquad y(t) := y_i(t, s_i), \qquad t_i \leq t \leq t_{i+1}, \qquad 0 \leq i \leq l-1,$$

where the requirements for s are

$$(3.183) \qquad y_i(t_{i+1}, s_i) = s_{i+1}, \qquad 0 \leq i \leq l-2,$$

$$(3.184) \qquad g(s_0, y_{l-1}(b, s_{l-1})) = 0.$$

So, defining

$$(3.185) \qquad F(s) = \begin{bmatrix} s_1 - y_0(t_1, s_0) \\ s_2 - y_1(t_2, s_1) \\ \cdot \\ \cdot \\ s_{l-1} - y_{l-2}(t_{l-1}, s_{l-2}) \\ g(s_0, y_{l-1}(b, s_{l-1})) \end{bmatrix},$$

with $g(y(0), y(T)) = \begin{bmatrix} \mathbf{x}(0) - \mathbf{x}_0 \\ \bar{\mathbf{x}}(T) - \phi_{\mathbf{x}}^\top \end{bmatrix}$, $s^\top = (s_0^\top, ..., s_{l-1}^\top), s_i \in \mathbf{R}^{2n}$, $0 \leq i \leq l-1$, we are left to solve a set of $2nl$ non-linear algebraic equations

$$(3.186) \qquad F(s) = 0.$$

Suppose, we are given an initial guess $y^0(0) = \begin{bmatrix} \mathbf{x}_0 \\ \bar{\mathbf{x}}^0(0) \end{bmatrix}$. Then, we generate a sequence of iterates $y^1, y^2, ..., y^j, ...$, with

$$(3.187) \qquad y^{j+1} = y^j + \tilde{F}\bar{\xi}^j, \qquad \text{with} \qquad \tilde{F} = \begin{bmatrix} \bar{F}^1 & 0 & \cdots & & \\ 0 & \bar{F}^2 & 0 & \cdots & \\ & \cdots & \cdots & & \\ & 0 & \bar{F}^{l-1} & 0 & \\ & \cdots & 0 & \bar{F}^l & \end{bmatrix},$$

using Newton's method for finding $\bar{\xi}^j$ by

$$(3.188) \qquad \dot{F}(s^j)\bar{\xi}^j = -F(s^j),$$

with $\dot{F}(s^j) = \bar{\mathcal{A}} = \begin{bmatrix} -\bar{Y}_0(t_1) & \bar{F}^1 & & & \\ & -\bar{Y}_1(t_2) & \bar{F}^2 & & \\ & \cdot & \cdot & \cdot & \\ & & & -\bar{Y}_{l-2}(t_{l-1}) & \bar{F}^{l-1} \\ & & & & B_{b2}\bar{Y}_{l-1} \end{bmatrix},$

$$(3.189) \qquad \dot{\bar{Y}}_i = A\bar{Y}_i, \qquad t_i < t < t_{i+1}, \qquad \bar{Y}_i(t) = \bar{F}_i, \qquad 0 \leq i \leq l-1,$$

with A from (3.179) and each $\bar{F}_i$ is a $2n \times n$ matrix of *rank* n. We choose $\bar{F}_0$, so that

$$B_{a1}\bar{F}_0 = 0. \tag{3.190}$$

How this is to be done see Section 3.2.7. For the general step i, we integrate a particular solution $y_i(t)$ by

$$\dot{y}_i = \psi(t), \qquad t_i < t < t_{i+1}, \qquad y_i(t) = s_i, \qquad 0 \le i \le l-1, \tag{3.191}$$

and a reduced fundamental solution $\bar{Y}_i(t)$ in (3.189) until the next shooting point is reached, i.e. $t = t_{i+1}$. The details are summarised to the following Algorithm 3.4.

Algorithm 3.4 Multiple Shooting Method Versus Reduced Compactification for Optimal Control Problems

Choose the mesh $\pi : 0 = t_0 < t_1 < ... < t_{l-1} < t_l = T$.
$j = -1$, $\bar{\xi} = 0$, *initial guess* $y^0(0)$.
Do:
  Initialisation:
  $j = j + 1$.
  If $(j = 0)$
    $s_0^0 = y^0(0)$.
    Determine $\bar{F}_0^j$ *by* $B_{a1}\bar{F}_0^j = 0$.
  Else
    $s_0^j = y^{j-1}(0)$, $\bar{F}_0^j = \bar{F}_0^{j-1}$.
  For $i = 0, 1, ..., l-1$ **DO**
    Original Step:
      Solve $\dot{\bar{Y}}_i^j = A\bar{Y}_i^j, \quad t_i < t < t_{i+1}, \quad \bar{Y}_i^j(t) = \bar{F}_i^j$.
      Solve $\dot{y}_i^j = \psi(t), \quad t_i < t < t_{i+1}, \quad y_i^j(t) = s_i^j$.
      Obtain $\bar{F}_{i+1}^j$ *from* $\bar{Y}_i^j(t_{i+1}) = \bar{F}_{i+1}^j \bar{G}_i^j$.
      Find $s_{i+1}^j := (I - \bar{F}_{i+1}^j(\bar{F}_{i+1}^j)^\top) y_i(t_{i+1})$.
  Reversal Initialisation:
    Solve $B_{b2}\bar{Y}_l(b)\bar{\xi}_l^j = \bar{\mathbf{x}}(T) - \phi_{\mathbf{x}}^\top$ *for* $\bar{\xi}_l^j$.
    Find $y^j(t_l) = \bar{F}_l^j \bar{\xi}_l^j + s_l^j$.
  For $i = l-1, l-2, ..., 0$ **DO**
    Reverse Step:
      Solve $\bar{F}_{i+1}^j \bar{\xi}_{i+1}^j - \bar{Y}_i(t_{i+1})\bar{\xi}_i^j = -(s_{i+1} - y_i(t_{i+1}, s_i))$ *for* $\bar{\xi}_i^j$.
      Find $y^j(t_i) = \bar{F}_i^j \bar{\xi}_i^j + s_i^j$.
While: $\max\limits_{0 \le i \le l} \|\bar{F}_i^j \bar{\xi}_i^j\| \ge TOL$ *and* $j <$ *MAX_ITER*.

Algorithm 3.4 consists of two sweeps through the time window $[0, T]$. Therefore, for a memory efficient implementation of Algorithm 3.4, checkpointing techniques from Chapter 2 can be applied. Moreover, Algorithm 3.4 is stable since the integration of the IVPs corresponding to the BVP (3.177) - (3.178) is stable in the forward direction over the subspace spanned by $\mathbf{x}(t)$. The multiple shooting method versus reduced compactification for optimal control problems, according to Algorithm 3.4, corresponds to a method in [25], which is called the indirect method there. Furthermore, it was described in [25] how an algorithm similar to Algorithm 3.4 can be efficiently implemented using AD techniques. The routine implementing Algorithm 3.4 was coded using the C++ programming language environment. This obtained routine was coupled with the existing routine **revolve** (see [39]) for the memory reduced implementation of Algorithm 3.4. Various run-time tests were performed. For a numerical test see the following Example 3.5.

Example 3.5 *This example is taken from [65]. Consider the electric circuit (tunnel-diode oscillator) shown in Figure 3.10 where L denotes inductivity, C capacity, R resistance, I electric current, and where D is the diode. The state variable $\mathbf{x}(t)$ represents the electric current I at time t. The voltage*

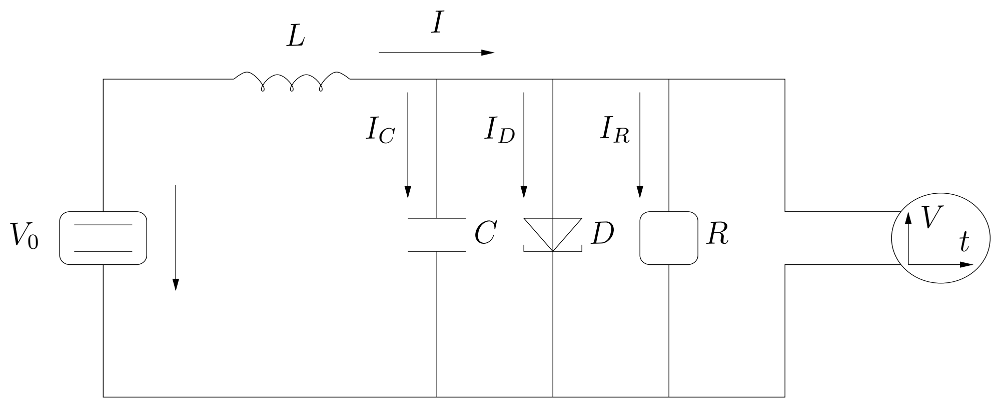

Figure 3.10: Tunnel-diode oscillator

$v_0(t)$ at the generator is regarded as a control function. After a suitable transformation of $v_0(t)$ we arrive at the following specific Rayleigh equation with a scalar control $\mathbf{u}(t)$:

$$\ddot{\mathbf{x}}(t) = -\mathbf{x}(t) + \dot{\mathbf{x}}(t)(1.4 - p\dot{\mathbf{x}}^2(t)) + 4\mathbf{u}(t). \tag{3.192}$$

The scalar value $p > 0$ in this equation is considered as a perturbation parameter for which we choose the nominal value $p_0 = 0.14$ and zero control $\mathbf{u}(t) \equiv 0$ has a limit cycle in the $(\mathbf{x}, \dot{\mathbf{x}})$ - plane. Define the state variables *as* $\mathbf{x}_1 = \mathbf{x}$ *and* $\mathbf{x}_2 = \dot{\mathbf{x}}$. *Therefore, consider the following parametric optimal control problem with fixed final time* $t_f = 4.5$ *and a perturbation* $p = 0.14$ *where the cost functional is given by*

$$F(\mathbf{x}, \mathbf{u}, p) = \int_0^{t_f} (\mathbf{u}^2(t) + \mathbf{x}_1^2(t))dt, \tag{3.193}$$

subject to

$$\dot{\mathbf{x}}_1(t) = \mathbf{x}_2(t), \qquad \dot{\mathbf{x}}_2(t) = -\mathbf{x}_1(t) + \mathbf{x}_2(t)(1.4 - p\mathbf{x}_2^2(t)) + 4\mathbf{u}(t), \tag{3.194}$$

$$\mathbf{x}_1(0) = \mathbf{x}_2(0) = -5, \qquad \mathbf{x}_1(t_f) = \mathbf{x}_2(t_f) = 0. \tag{3.195}$$

The Hamiltonian (3.4) becomes

$$H(\mathbf{x}_1, \mathbf{x}_2, \mathbf{u}, \bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2, p) = \mathbf{u}^2 + \mathbf{x}_1^2 + \bar{\mathbf{x}}_1\mathbf{x}_2 + \bar{\mathbf{x}}_2(-\mathbf{x}_1 + \mathbf{x}_2(1.4 - p\mathbf{x}_2^2) + 4\mathbf{u}), \tag{3.196}$$

where $\bar{\mathbf{x}}_1$, $\bar{\mathbf{x}}_2$ *are the adjoint variables associated with* $\mathbf{x}_1$, $\mathbf{x}_2$. *The optimal control is determined via minimum condition (3.21):*

$$H_{\mathbf{u}} = 2\mathbf{u} + 4\bar{\mathbf{x}}_2 = 0, \quad i.e. \quad \mathbf{u}(t) = -2\bar{\mathbf{x}}_2(t). \tag{3.197}$$

The strict Legendre-Clebsch condition (3.27) holds in view of $H_{\mathbf{uu}} \equiv 2 > 0$. *The adjoint equations (3.11) are:*

$$\dot{\bar{\mathbf{x}}}_1 = \bar{\mathbf{x}}_2 - 2\mathbf{x}_1, \qquad \dot{\bar{\mathbf{x}}}_2 = 3p\bar{\mathbf{x}}_2\mathbf{x}_2^2 - 1.4\bar{\mathbf{x}}_2 - \bar{\mathbf{x}}_1. \tag{3.198}$$

The solution is then obtained by solving the BVP consisting of equations (3.194), (3.195), and (3.198) where the control $\mathbf{u}$ *is substituted from (3.197).*

The BVP (3.194), (3.195), and (3.198) is solved using the Multiple Shooting Method Versus Reduced Compactification for Optimal Control Problems, according to Algorithm 3.4. Figures 3.11 - 3.12 show the obtained control $\mathbf{u}(t)$, *the state variables* $\mathbf{x}_1(t)$, $\mathbf{x}_2(t)$, *and the adjoint variables* $\bar{\mathbf{x}}_1(t)$, $\bar{\mathbf{x}}_2(t)$.

For the memory efficient implementation of Algorithm 3.4, the checkpointing techniques from Chapter 2 are applied. The time interval $[0, t_f] = [0, 4.25]$ *is uniformly discretised in* l *time steps with* $l = 200, 500, 1000$. *Therefore, binomial reversal schedules, which are optimal reversal schedules for uniform cases, are applied to the implementation of Algorithm 3.4. Figure 3.7 illustrates the run-time requirement, needed to implement Algorithm 3.4 using binomial reversal schedules with various numbers of checkpoints.*

The results are similar to Figure 3.3 in Example 3.1, namely, the memory requirement, i.e. the number of available checkpoints, can be pushed far below

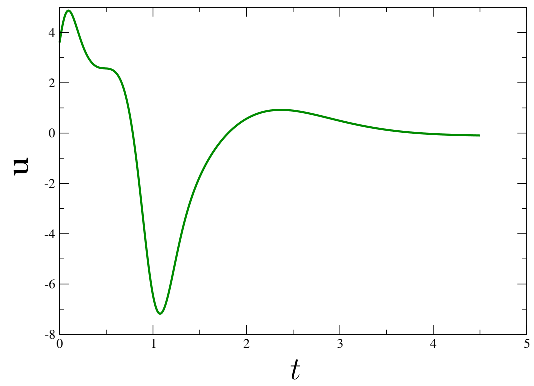

Figure 3.11: Optimal control $\mathbf{u}$

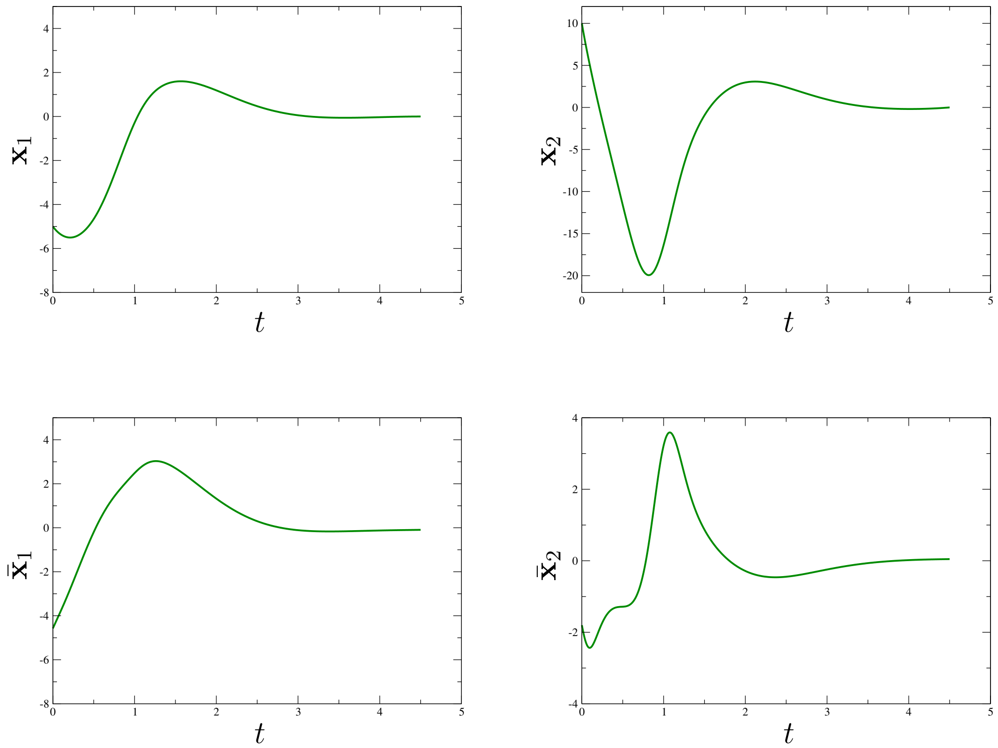

Figure 3.12: State variables $\mathbf{x}_1(t)$, $\mathbf{x}_2(t)$ and adjoint variables $\bar{\mathbf{x}}_1(t)$, $\bar{\mathbf{x}}_2(t)$

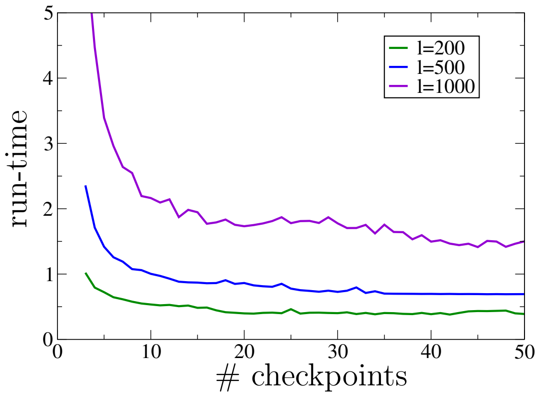

Figure 3.13: Run-time behaviour obtained from the implementation of Algorithm 3.4 for the solution of the BVP (3.194), (3.195), and (3.198), using binomial reversal schedules with various numbers of checkpoints

ten before any impact on the run-time becomes apparent. Only when the number of checkpoints becomes lower than three, we observe a significant increase in run-time. These threshold results are not constant and will decline as the total number of time steps, i.e. the discretisation refinement, grows. This implies that the utilisation of checkpointing techniques for this example results in a slight increase of run-time if the number of checkpoints is less than ten. At the same time, the memory requirement is reduced drastically, compared to the approach where all intermediate states are stored.

□

Suppose, we want to utilise another IVP method for solving the BVP (3.177) - (3.178), for example, the compactification for decoupled systems, according to Algorithm 3.1, or Riccati approach, according to Algorithm 3.3, for nonlinear problems. Then, we are facing some additional problems. Recall, using these methods, we evaluated the last $(n-k)$ components of the vectors w_i or y_i at first. Moreover, during this evaluation, the corresponding IVPs were integrated in the stable direction for this problem (in the forward direction) from the corresponding BC, which control these last $(n-k)$ components w_i or y_i.

Applying these considerations to solving the BVP (3.175) - (3.176), using compactification method or Riccati approach for decoupled systems, we should start to integrate from the end-BC, i.e. $t_0 = T$, backward in order to evaluate the last n of $2n$ components of corresponding vectors ξ_i. But, since the BC for $\bar{\mathbf{x}}(T)$ depends on $\mathbf{x}(T)$, i.e. $\bar{\mathbf{x}}(T) = \phi_{\mathbf{x}}^{\top}$, we have to make initially the forward sweep from 0 to T in order to determine the value $\bar{\mathbf{x}}(T)$ for corresponding value $\mathbf{x}(T)$, i.e. we have to integrate a trajectory

$$\dot{\mathbf{x}} = f(\mathbf{x}(t), \bar{\mathbf{x}}(t), t), \quad \mathbf{x}(0) = \mathbf{x}_0. \tag{3.199}$$

We illustrate details of this procedure in Algorithm 3.5 for solving optimal control problems, using Riccati approach for decoupled systems. Compactification approach for solving optimal control problems is to obtain in a similar way.

Algorithm 3.5 Riccati Approach for Solving Optimal Control Problems

Choose the mesh $\pi : 0 = t_0 < t_1 < ... < t_{l-1} < t_l = T$.
$j = -1$, $\xi_i^0 = 0$, *set* y_i^0, $i = 0, ..., l-1$.
Do:
  Initialisation:
  $j = j + 1$. $\mathbf{x}_0^j = \mathbf{x}_0$.
  For $i = 0, 1, ..., l-1$ **DO**
    Original Step:
      Integrate $\dot{\mathbf{x}}^j = f(\mathbf{x}^j(t), \bar{\mathbf{x}}^j(t), t), \quad t_i \le t \le t_{i+1}$.
  Adjoint Initialisation:
    Determine $\bar{\mathbf{x}}_l^j = \phi_{\mathbf{x}}^{\top}(\mathbf{x}_l^j)$.
    Determine K_l *and* w_l^2 *using (3.132) - (3.138) and (3.178).*
  For $i = l-1, l-2, ..., 0$ **DO**
    Adjoint Step:
      Evaluate T_{i+1} *and* U_i *using i.e. discrete Riccati equation.*
      Evaluate w_i^2 *using (3.113).*
  Final Initialisation:
    Evaluate w_0^j *using (3.132) - (3.138) and (3.178).*
    Evaluate $\xi_0^j = T_0 w_0$.
    $y_0^{j+1} = y_0^j + \xi_0^j$.
  For $i = 1, 2, ..., l$ **DO**
    Final Step:
      Evaluate w_i^1 *using (3.112).*
      Evaluate $\xi_i^j = T_i w_i$.
      $y_i^{j+1} = y_i^j + \xi_i^j$.
While: $\max_{0 \le i \le l} \|\xi_i^j\| \ge TOL$ *and* $j <$ *MAX_ITER.*

Algorithm 3.5 consists of three sweeps through the time window $[0, T]$, i.e.

the original, the adjoint, and the final sweep. Therefore, checkpointing techniques from Chapter 2 can not be applied for a memory efficient implementation of Algorithm 3.5. Checkpointing techniques for a memory efficient implementation of algorithms, consisting of more than two alternative sweeps, will be established later on (see Chapter 4).

3.4 Finite Difference Methods

In this section we consider methods which are conceptually different from the initial value techniques of previous sections. Here, an approximate solution is to find over the entire interval of interest. Therefore, these methods are called **global methods**. The description of these methods closely follows the description in [3]. There is a close theoretical relationship between these methods and initial value techniques. This theoretical relationship will be shown in this section.

The basic steps of a finite difference method are given in the following:

- Choose a mesh

  (3.200) $$\pi : a = t_0 < t_1 < ... < t_{l-1} < t_l = b.$$

  Approximate solution values will be found at these mesh points t_i.

- Replace derivatives of a BVP with differential equations. Using this and incorporating the boundary conditions, form a set of algebraic equations for the approximate solution values.

- Solve the resulting system of differential equations to obtain a set of discrete solution values $y_i \equiv y_\pi(t_i)$.

3.4.1 Simple Finite Difference Schemes for Linear Problems

Consider the linear first-order BVP

(3.201) $$\dot{y} = A(t)y + q(t), \qquad a < t < b,$$

(3.202) $$B_a y(a) + B_b y(b) = \beta.$$

Here, we present one example of one-step schemes, namely, the **trapezoidal scheme**, defined by

(3.203)
$$\frac{y_{i+1} - y_i}{h_i} = \frac{1}{2}[A(t_{i+1})y_{i+1} + A(t_i)y_i] + \frac{1}{2}[q(t_{i+1}) + q(t_i)], \qquad 0 \le i \le l-1.$$

In matrix form this method can be written as

$$\mathcal{A}y_\pi = \widehat{\beta}, \tag{3.204}$$

or,

$$\begin{pmatrix} S_0 & R_0 & & & \\ 0 & S_1 & R_1 & & \\ & & . & . & \\ & & 0 & S_{l-1} & R_{l-1} \\ B_a & & & 0 & B_b \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{l-1} \\ y_l \end{pmatrix} = \begin{pmatrix} q_0 \\ q_1 \\ \vdots \\ q_{l-1} \\ \beta \end{pmatrix}, \tag{3.205}$$

with S_i, R_i are $n \times n$ matrices, defined by

$$S_i = -h^{-1}I - \frac{1}{2}A(t_i), \qquad R_i = h^{-1}I - \frac{1}{2}A(t_{i+1}), \tag{3.206}$$

$$q_i = \frac{1}{2}[q(t_{i+1}) + q(t_i)], \qquad 0 \le i \le l-1. \tag{3.207}$$

$\mathcal{A}$ in (3.204) has the same block structure as the matrices $\mathcal{A}$ for multiple shooting, stabilised march, and Riccati method discussed in Section 3.2.

3.4.2 Simple Finite Difference Schemes for Non-Linear Problems

Consider the non-linear first-order system of n ODEs

$$\dot{y} = f(t, y(t)), \qquad a < t < b, \tag{3.208}$$

$$g(y(a), y(b)) = 0. \tag{3.209}$$

For numerical approximation consider a mesh π in (3.200). y_π denotes the vector of approximate solution values. The trapezoidal scheme (3.203) for linear problems can be straightforwardly extended for non-linear problems in the following way:

$$\frac{y_{i+1}-y_i}{h_i} = \frac{1}{2}[f(t_{i+1}, y_{i+1}) + f(t_i, y_i)], \qquad 0 \le i \le l-1, \tag{3.210}$$

$$g(y_0, y_l) = 0. \tag{3.211}$$

Thus, we obtain a system of $n(l+1)$ algebraic equations for the $n(l+1)$ unknowns y_π. Newton's method can be applied to solve the non-linear problems.

Recall, for the system of non-linear algebraic equations

$$F(s) = 0, \tag{3.212}$$

Newton's method is a fixed-point iteration

$$s^{j+1} = G(s^j), \qquad j = 0, 1, 2, ..., \tag{3.213}$$

with the iteration function $G(s)$ defined by

$$G(s) := s - [\dot{F}(s)]^{-1} F(s). \tag{3.214}$$

$\dot{F}(s) = \frac{\partial F(s)}{\partial s}$ is the Jacobian matrix. Therefore, we obtain

$$\dot{F}(s^j)\xi = -F(s^j), \tag{3.215}$$

and

$$s^{j+1} := s^j + \xi. \tag{3.216}$$

Let the non-linear algebraic equations (3.212) be given by the trapezoidal scheme (3.210) - (3.211), with $s^\top = y_\pi^\top = (y_0^\top, ..., y_l^\top)$, an $n(l+1)$ vector. Denoting

$$N_\pi(y_i) := \frac{y_{i+1}-y_i}{h_i} - \frac{1}{2}[f(t_{i+1}, y_{i+1}) + f(t_i, y_i)], \tag{3.217}$$

we obtain $F(s)^\top := (N_\pi^\top(y_0), ..., N_\pi^\top(y_{l-1}), g^\top(y_0, y_l))$. Newton's iteration (3.215) gives

$$\frac{w_{i+1}-w_i}{h_i} - \frac{1}{2}[A(t_{i+1})w_{i+1} + A(t_i)w_i] = -N_\pi(y_i^j), \qquad 0 \le i \le l-1, \tag{3.218}$$

$$B_a w_0 + B_b w_l = -g(y_0^j, y_l^j). \tag{3.219}$$

Here, $\xi^\top = w_\pi^\top = (w_0^\top, ..., w_l^\top)$, y_π^j are known values from a previous iteration (y_π^0 is an initial guess).

$$A(t_i) := \frac{\partial f}{\partial y}(t_i, y_i^j), \qquad 0 \le i \le l, \tag{3.220}$$

$$B_a = \frac{\partial g(u,v)}{\partial u}, \quad B_b = \frac{\partial g(u,v)}{\partial v} \quad at \quad u = y_0^j, \quad v = y_l^j. \tag{3.221}$$

The next iterate y_i^{j+1} is given, according to (3.216), by $y_i^{j+1} := y_i^j + w_i,\ i = 0, ..., l$.

The system (3.218) - (3.219) for the correction vector w_π is a linear system of equations. Note that in each iteration we perform two operations in succession, i.e. discretisation and linearisations.

Consider reversing the order of this operations. Letting $y^j(t)$ be an appropriately smooth function satisfying

$$y^j(t_i) = y_i^j, \qquad i = 0, ..., l, \tag{3.222}$$

and linearising the problem (3.208) - (3.209), we obtain

$$\dot{w} - A(t)w = -[\dot{y}^j(t) - f(t, y^j(t))] \tag{3.223}$$

$$B_a w(a) + B_b w(b) = -g(y^j(a), y^j(b)). \tag{3.224}$$

Here, $A(t)$, B_a, and B_b are given in (3.220) - (3.221) with an obvious extension. Use the trapezoidal scheme (3.203) for the **linear** problem (3.223) - (3.224). The result is precisely (3.218) - (3.219). Thus, the two operations of linearisations and discretisation commute for the trapezoidal scheme. The next iterate $y^{j+1}(t)$ for the quasilinearisation process is obtained by $y^{j+1}(t_i) := y^j(t_i) + w_i$. The process of first linearisation and second discretisation for non-linear problems is called **quasilinearisation** (for details see [3]). It is easy to verify that quasilinearisation is equivalent to Newton's method applied to the discretised difference equations for most methods.

3.5 Solving Optimal Control Problems Using Quasilinearisation Techniques

3.5.1 Continuous Problem

Consider anew the Euler-Lagrange equations for the optimal control problem (3.1) - (3.2)

$$\dot{\mathbf{x}} = f(\mathbf{x}(t), \mathbf{u}(t), t), \quad \mathbf{x}(0) = \mathbf{x}_0, \tag{3.225}$$

$$\dot{\bar{\mathbf{x}}} = -H_{\mathbf{x}}^{\top} = -f_{\mathbf{x}}^{\top}\,\bar{\mathbf{x}}, \qquad \bar{\mathbf{x}}(T) = \phi_{\mathbf{x}}^{\top}(\mathbf{x}(T)), \tag{3.226}$$

$$\left(\frac{\partial H}{\partial \mathbf{u}}\right)^{\top} = 0, \qquad 0 \le t \le T, \tag{3.227}$$

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

$$H(\mathbf{x}(t), \mathbf{u}(t), \bar{\mathbf{x}}(t), t) = \bar{\mathbf{x}}^{\top}(t)\, f(\mathbf{x}(t), \mathbf{u}(t), t). \tag{3.228}$$

We linearise (3.225), (3.226), and (3.227) about a reference solution $\mathbf{x}(t)$, $\mathbf{u}(t)$, and $\bar{\mathbf{x}}(t)$ and obtain the following equations for variations $\delta\mathbf{x}(t)$, $\delta\bar{\mathbf{x}}(t)$, and $\delta\mathbf{u}(t)$

$$\delta\dot{\mathbf{x}} - f_{\mathbf{x}}\delta\mathbf{x} - f_{\mathbf{u}}\delta\mathbf{u} = 0, \tag{3.229}$$

$$\delta\dot{\bar{\mathbf{x}}} + H_{\mathbf{xx}}^{\top}\delta\mathbf{x} + H_{\mathbf{xu}}^{\top}\delta\mathbf{u} + H_{\mathbf{x}\bar{\mathbf{x}}}^{\top}\delta\bar{\mathbf{x}} = 0, \tag{3.230}$$

$$H_{\mathbf{u}}^{\top} + H_{\mathbf{ux}}^{\top}\,\delta\mathbf{x} + H_{\mathbf{uu}}^{\top}\,\delta\mathbf{u} + H_{\mathbf{u}\bar{\mathbf{x}}}^{\top}\,\delta\bar{\mathbf{x}} = 0, \tag{3.231}$$

with the linearised initial and terminal conditions

$$\begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} \delta\mathbf{x}(0) \\ \delta\bar{\mathbf{x}}(0) \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ -\phi_{\mathbf{xx}}^{\top}(T) & I \end{pmatrix}\begin{pmatrix} \delta\mathbf{x}(T) \\ \delta\bar{\mathbf{x}}(T) \end{pmatrix} = -\begin{pmatrix} \mathbf{x}(0) - \mathbf{x}_0 \\ \bar{\mathbf{x}}(T) - \phi_{\mathbf{x}}^{\top}(T) \end{pmatrix}. \tag{3.232}$$

After expressing $\delta\mathbf{u}$ in terms of $\delta\mathbf{x}$ and $\delta\bar{\mathbf{x}}$ from the relation (3.231) we obtain

$$\delta\mathbf{u} = -H_{\mathbf{uu}}^{-1}\,\left(H_{\mathbf{u}}^{\top} + H_{\mathbf{ux}}^{\top}\,\delta\mathbf{x} + H_{\mathbf{u}\bar{\mathbf{x}}}^{\top}\,\delta\bar{\mathbf{x}}\right). \tag{3.233}$$

Substitution of this expression into (3.229) - (3.230) yields the following linear BVP

$$\begin{pmatrix} \delta\dot{\mathbf{x}} \\ \delta\dot{\bar{\mathbf{x}}} \end{pmatrix} = S(t)\begin{pmatrix} \delta\mathbf{x} \\ \delta\bar{\mathbf{x}} \end{pmatrix} + q(t), \tag{3.234}$$

where
(3.235)
$$S(t) = \begin{pmatrix} S^{11} & S^{12} \\ S^{21} & S^{22} \end{pmatrix} = \left(\begin{array}{c|c} f_{\mathbf{x}} - f_{\mathbf{u}}\, H_{\mathbf{uu}}^{-1}\, H_{\mathbf{xu}} & -f_{\mathbf{u}}\, H_{\mathbf{uu}}^{-1}\, f_{\mathbf{u}}^{\top} \\ -H_{\mathbf{xx}} + H_{\mathbf{ux}}\, H_{\mathbf{uu}}^{-1}\, H_{\mathbf{xu}} & H_{\mathbf{ux}}\, H_{\mathbf{uu}}^{-1}\, f_{\mathbf{u}}^{\top} - f_{\mathbf{x}}^{\top} \end{array} \right)$$

is the system matrix, and

$$q(t) = \begin{pmatrix} q^1 \\ q^2 \end{pmatrix} = \begin{pmatrix} -f_{\mathbf{u}}\, H_{\mathbf{uu}}^{-1}\, H_{\mathbf{u}}^{\top} \\ H_{\mathbf{ux}}\, H_{\mathbf{uu}}^{-1}\, H_{\mathbf{u}}^{\top} \end{pmatrix} \tag{3.236}$$

is the non-homogeneous part. The BC for this problem are given by the relation (3.232). Rather than solving this linear BVP using collocation or another 'global' discretisation scheme we prefer the Riccati approach which computes 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

$$\begin{pmatrix} \delta\mathbf{x}(t) \\ \delta\bar{\mathbf{x}}(t) \end{pmatrix} = \begin{pmatrix} I & 0 \\ K(t) & I \end{pmatrix} \begin{pmatrix} \delta\mathbf{x}(t) \\ a(t) \end{pmatrix}. \tag{3.237}$$

In order to determine a suitable $K(t)$ and the corresponding $a(t)$, we substitute the $\begin{pmatrix} \delta\mathbf{x} \\ \delta\bar{\mathbf{x}} \end{pmatrix}$ in terms of $\begin{pmatrix} \delta\mathbf{x} \\ a \end{pmatrix}$ in (3.234), which yields
(3.238)
$$\frac{d}{dt}\begin{pmatrix} I & 0 \\ K & I \end{pmatrix} \begin{pmatrix} \delta\mathbf{x} \\ a \end{pmatrix} = \begin{pmatrix} S^{11} & S^{12} \\ S^{21} & S^{22} \end{pmatrix} \begin{pmatrix} I & 0 \\ K & I \end{pmatrix} \begin{pmatrix} \delta\mathbf{x} \\ a \end{pmatrix} + \begin{pmatrix} q^1 \\ q^2 \end{pmatrix}.$$

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

$$\dot{K}(t) = S^{21} - K(t)S^{11} + S^{22}K(t) - K(t)S^{12}K(t), \tag{3.239}$$

then, the new variables $(\delta\mathbf{x}, a)^{\top}$ satisfy the block system

$$\begin{pmatrix} \delta\dot{\mathbf{x}} \\ \dot{a} \end{pmatrix} = \left(\begin{array}{c|c} S^{11} + S^{12}K & S^{12} \\ 0 & S^{22} - KS^{12} \end{array} \right) \begin{pmatrix} \delta\mathbf{x} \\ a \end{pmatrix} + \begin{pmatrix} q^1 \\ q^2 - Kq^1 \end{pmatrix}, \tag{3.240}$$

with the separated BC
(3.241)
$$\begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \delta\mathbf{x}(0) \\ a(0) \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ -\phi_{\mathbf{xx}}^{\top}(T) + K(T) & I \end{pmatrix} \begin{pmatrix} \delta\mathbf{x}(T) \\ a(T) \end{pmatrix} = -\begin{pmatrix} \mathbf{x}(0) - \mathbf{x}_0 \\ \bar{\mathbf{x}}(T) - \phi_{\mathbf{x}}^{\top}(T) \end{pmatrix}.$$

This approach leads to the following conceptual algorithm.

Algorithm 3.6 Quasilinearisation Scheme for Solving Optimal Control Problems Using Riccati Approach
Choose initial control trajectory $\mathbf{u}^0(t)$, $t \in [0, T]$, $j = 0$.
Do:
  Original initialisation:
    $\mathbf{x}^j(0) = \mathbf{x}_0$.
  Original sweep: $t : 0 \to T$
    Integrate forward $\dot{\mathbf{x}}^j = f(\mathbf{x}^j(t), \mathbf{u}^j(t), t)$.
  Adjoint initialisation:
    Set $\bar{\mathbf{x}}^j(T) = \phi_{\mathbf{x}}^\top(T)$.
    Set $K(T) = \phi_{\mathbf{xx}}^\top(T)$ *and* $a(T) = 0$.
  Adjoint sweep: $t : T \to 0$
    Integrate backward $\dot{\bar{\mathbf{x}}}^j = -f_{\mathbf{x}}^\top(\mathbf{x}^j(t), \mathbf{u}^j(t), t)\, \bar{\mathbf{x}}^j$.
    Integrate backward $\dot{K}(t) = S^{21} - K(t)S^{11} + S^{22}K(t) - K(t)S^{12}K(t)$.
    Integrate backward $\dot{a}(t) = (-K(t)S^{12} + S^{22})\, a(t) - K(t)q^1 + q^2$.
  Final initialisation:
    Set $\delta\mathbf{x}^j(0) = 0$.
  Final sweep: $t : 0 \to T$
    Integrate forward $\delta\dot{\mathbf{x}}^j = (S^{11} + S^{12}K)\delta\mathbf{x}^j + S^{12}a + q^1$.
    Evaluate $\delta\mathbf{u}^j = -H_{\mathbf{uu}}^{-1}\left(H_{\mathbf{u}}^\top + H_{\mathbf{ux}}^\top\, \delta\mathbf{x}^j + H_{\mathbf{u}\bar{\mathbf{x}}}^\top \delta\bar{\mathbf{x}}^j\right)$.
    $\mathbf{u}^{j+1}(t) = \mathbf{u}^j(t) + \delta\mathbf{u}^j(t)$.
  $j = j + 1$.
While: $\|\delta\mathbf{u}^j(t)\| \geq TOL$ *and* $j < MAX\_ITER$.

3.5.2 Discrete Problem

Consider a discrete case of the optimal control problem

$$\min_{\mathbf{x},\mathbf{u}} \phi(\mathbf{x}_l), \tag{3.242}$$

$$\mathbf{x}_{i+1} = f_i(\mathbf{x}_i, \mathbf{u}_i), \quad i = 0, ..., l-1, \quad \mathbf{x}_0 \text{ fixed}, \tag{3.243}$$

and the corresponding Euler-Lagrange equations

$$\mathbf{x}_{i+1} = f_i(\mathbf{x}_i, \mathbf{u}_i), \quad \mathbf{x}_0 \text{ fixed}, \tag{3.244}$$

$$\bar{\mathbf{x}}_i = H_{\mathbf{x},i}^\top = f_{\mathbf{x},i}^\top\, \bar{\mathbf{x}}_{i+1}, \quad \bar{\mathbf{x}}_l = \phi_{\mathbf{x}}^\top(\mathbf{x}_l), \tag{3.245}$$

$$H_{\mathbf{u},i}^\top = f_{\mathbf{u},i}^\top\, \bar{\mathbf{x}}_{i+1} = 0, \tag{3.246}$$

where the **Hamiltonian** H_i is evaluated by

$$H_i(\mathbf{x}_i, \mathbf{u}_i, \bar{\mathbf{x}}_{i+1}) = \bar{\mathbf{x}}_{i+1}^\top f_i(\mathbf{x}_i, \mathbf{u}_i). \tag{3.247}$$

We linearise the equations (3.244), (3.245), and (3.246) about a reference solution $\mathbf{x}_i(t)$, $\mathbf{u}_i(t)$, and $\bar{\mathbf{x}}_i(t)$ and obtain the following recursive equations for $\delta\mathbf{x}_i$, $\delta\bar{\mathbf{x}}_i$, and $\delta\mathbf{u}_i$

$$\delta\mathbf{x}_{i+1} - f_{\mathbf{x},i}\delta\mathbf{x}_i - f_{\mathbf{u},i}\delta\mathbf{u}_i = -(\mathbf{x}_{i+1} - f_i) = 0, \tag{3.248}$$

$$\delta\bar{\mathbf{x}}_i - H_{\mathbf{xx},i}^\top\delta\mathbf{x}_i - H_{\mathbf{xu},i}^\top\delta\mathbf{u}_i - H_{\mathbf{x\bar{x}},i}^\top\delta\bar{\mathbf{x}}_{i+1} = -(\bar{\mathbf{x}}_i - H_{\mathbf{x},i}^\top) = 0, \tag{3.249}$$

$$H_{\mathbf{u},i}^\top + H_{\mathbf{ux},i}^\top\,\delta\mathbf{x}_i + H_{\mathbf{uu},i}^\top\,\delta\mathbf{u}_i + H_{\mathbf{u\bar{x}},i}^\top\,\delta\bar{\mathbf{x}}_{i+1} = 0, \tag{3.250}$$

with the linearised initial and terminal conditions
(3.251)
$$\begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} \delta\mathbf{x}_0 \\ \delta\bar{\mathbf{x}}_0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ -\phi_{\mathbf{xx}}^\top(\mathbf{x}_l) & I \end{pmatrix}\begin{pmatrix} \delta\mathbf{x}_l \\ \delta\bar{\mathbf{x}}_l \end{pmatrix} = -\begin{pmatrix} \mathbf{x}_0 - \mathbf{x}_0 \\ \bar{\mathbf{x}}_l - \phi_{\mathbf{x}}^\top(\mathbf{x}_l) \end{pmatrix}.$$

After expressing $\delta\mathbf{u}_i$ in terms of $\delta\mathbf{x}_i$ and $\delta\bar{\mathbf{x}}_{i+1}$ from the relation (3.250) we obtain

$$\delta\mathbf{u}_i = -H_{\mathbf{uu},i}^{-\top}\left[H_{\mathbf{u},i}^\top + H_{\mathbf{ux},i}^\top\,\delta\mathbf{x}_i + H_{\mathbf{u\bar{x}},i}^\top\,\delta\bar{\mathbf{x}}_{i+1}\right]. \tag{3.252}$$

Substitution of this expression in the (3.248) - (3.249) yields the following linear BVP

$$\begin{pmatrix} \delta\mathbf{x}_{i+1} \\ \delta\bar{\mathbf{x}}_{i+1} \end{pmatrix} = S_i\begin{pmatrix} \delta\mathbf{x}_i \\ \delta\bar{\mathbf{x}}_i \end{pmatrix} + q_i, \tag{3.253}$$

where

$$S_i = \begin{pmatrix} S_i^{11} & S_i^{12} \\ S_i^{21} & S_i^{22} \end{pmatrix} = \begin{pmatrix} -H_4 - H_5H_1^{-1}H_2 & -H_5H_1^{-1} \\ -H_1^{-1}H_2 & -H_1^{-1} \end{pmatrix} \tag{3.254}$$

is the system matrix and

$$q_i = \begin{pmatrix} q_i^1 \\ q_i^2 \end{pmatrix} = \begin{pmatrix} -H_6 - H_5H_1^{-1}H_3 \\ -H_1^{-1}H_3 \end{pmatrix} \tag{3.255}$$

is the non-homogeneous part. Here, we use the following notations

$$H_1 = H_{\mathbf{xu},i}^\top\,H_{\mathbf{uu},i}^{-\top}\,f_{\mathbf{u},i}^\top - f_{\mathbf{x},i}^\top,$$

$$H_2 = H_{\mathbf{xu},i}^{\top}\, H_{\mathbf{uu},i}^{-\top}\, H_{\mathbf{ux},i}^{\top} - H_{\mathbf{xx},i}^{\top},$$

$$H_3 = H_{\mathbf{xu},i}^{\top}\, H_{\mathbf{uu},i}^{-\top}\, H_{\mathbf{u},i}^{\top},$$

$$H_4 = f_{\mathbf{u},i}\, H_{\mathbf{uu},i}^{-\top}\, H_{\mathbf{ux},i}^{\top} - f_{\mathbf{x},i},$$

$$H_5 = -f_{\mathbf{u},i}\, H_{\mathbf{uu},i}^{-\top}\, f_{\mathbf{u},i}^{\top},$$

$$H_6 = f_{\mathbf{u},i}\, H_{\mathbf{uu},i}^{-\top}\, H_{\mathbf{u},i}^{\top}.$$

The BC for this problem are given by the relation (3.251). In order to achieve a suitable decoupling for the solution components, we consider a linear transformation of the form

$$\begin{pmatrix} \delta\mathbf{x}_i \\ \delta\bar{\mathbf{x}}_i \end{pmatrix} = \begin{pmatrix} I & 0 \\ K_i & I \end{pmatrix} \begin{pmatrix} \delta\mathbf{x}_i \\ a_i \end{pmatrix}. \tag{3.256}$$

To determine a suitable K_i and the corresponding a_i we substitute the $\begin{pmatrix} \delta\mathbf{x}_i \\ \delta\bar{\mathbf{x}}_i \end{pmatrix}$ in terms of $\begin{pmatrix} \delta\mathbf{x}_i \\ a_i \end{pmatrix}$ in the equation (3.253), which yields

(3.257)
$$\begin{pmatrix} I & 0 \\ K_{i+1} & I \end{pmatrix} \begin{pmatrix} \delta\mathbf{x}_{i+1} \\ a_{i+1} \end{pmatrix} = \begin{pmatrix} S_i^{11} & S_i^{12} \\ S_i^{21} & S_i^{22} \end{pmatrix} \begin{pmatrix} I & 0 \\ K_i & I \end{pmatrix} \begin{pmatrix} \delta\mathbf{x}_i \\ a_i \end{pmatrix} + \begin{pmatrix} q_i^1 \\ q_i^2 \end{pmatrix}.$$

Finally, after some other transformations, we obtain the following linear BVP in new variables $\delta\mathbf{x}_i$ and a_i

(3.258)
$$\begin{pmatrix} \delta\mathbf{x}_{i+1} \\ a_{i+1} \end{pmatrix} = \begin{pmatrix} S_i^{11} + S_i^{12}K_i & | & S_i^{12} \\ -K_{i+1}(S_i^{11} + S_i^{12}K_i) + S_i^{21} + S_i^{22}K_i & | & S_i^{22} - K_{i+1}S_i^{12} \end{pmatrix} \begin{pmatrix} \delta\mathbf{x}_i \\ a_i \end{pmatrix} + \begin{pmatrix} q_i^1 \\ q_i^2 - K_{i+1}q_i^1 \end{pmatrix}.$$

If we choose K_i as the solution of the discrete Riccati equation

$$\left(S_i^{22} - K_{i+1}S_i^{12}\right) K_i = \left(-S_i^{21} + K_{i+1}S_i^{11}\right), \tag{3.259}$$

then, the new variables $(\delta\mathbf{x}, a)^{\top}$ satisfy the block system

(3.260)
$$\begin{pmatrix} \delta\mathbf{x}_{i+1} \\ a_{i+1} \end{pmatrix} = \begin{pmatrix} S_i^{11} + S_i^{12}K_i & S_i^{12} \\ 0 & S_i^{22} - K_{i+1}S_i^{12} \end{pmatrix} \begin{pmatrix} \delta\mathbf{x}_i \\ a_i \end{pmatrix} + \begin{pmatrix} q_i^1 \\ q_i^2 - K_{i+1}q_i^1 \end{pmatrix},$$

with the separated BC

(3.261)
$$\begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \delta\mathbf{x}_0 \\ a_0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ -\phi_{\mathbf{xx}}^{\top}(\mathbf{x}_l) + K_l & I \end{pmatrix} \begin{pmatrix} \delta\mathbf{x}_l \\ a_l \end{pmatrix} = -\begin{pmatrix} \mathbf{x}_0 - \mathbf{x}_0 \\ \bar{\mathbf{x}}_l - \phi_{\mathbf{x}}^{\top}(\mathbf{x}_l) \end{pmatrix}.$$

This approach leads to the following conceptual algorithm.

Algorithm 3.7 Quasilinearisation Scheme for Solving Optimal Control Problems Using Riccati Approach (Discrete Case)

Choose initial control trajectory $\mathbf{u}_0^0, \ldots, \mathbf{u}_{l-1}^0$, $j = 0$.

Do:

  Original Initialisation:

  $\mathbf{x}_0^j = \mathbf{x}_0$.

  For $i = 0, 1, \ldots, l-1$ **DO**

    Original Step:

      $\mathbf{x}_{i+1}^j = f_i(\mathbf{x}_i^j, \mathbf{u}_i^j)$.

  Adjoint Initialisation:

    Set $\bar{\mathbf{x}}_l^j = \phi_{\mathbf{x}}^\top(\mathbf{x}_l^j)$.

    Set $K_l = \phi_{\mathbf{xx}}^\top(\mathbf{x}_l^j)$ *and* $a_l = 0$.

  For $i = l-1, l-2, \ldots, 0$ **DO**

    Adjoint Step:

      Evaluate $\bar{\mathbf{x}}_i = f_{\mathbf{x},i}^\top \bar{\mathbf{x}}_{i+1}$.

      Evaluate $K_i = -H_1 \left(K_{i+1} H_5 - I\right)^{-1} K_{i+1} H_4 - H_2$.

      Evaluate $a_i = H_1 \left(K_{i+1} H_5 - I\right)^{-1} \left(a_{i+1} - K_{i+1} H_6\right) - H_3$.

  Final Initialisation:

    Set $\delta \mathbf{x}_0^j = 0$.

  For $i = 0, 1, \ldots, l-1$ **DO**

    Final Step:

      Evaluate $\delta \mathbf{u}_i^j = -H_{\mathbf{uu},i}^{-\top} \left[H_{\mathbf{u},i}^\top + H_{\mathbf{ux},i}^\top \delta \mathbf{x}_i^j + H_{\mathbf{u}\bar{\mathbf{x}},i}^\top \delta \bar{\mathbf{x}}_{i+1}^j\right]$.

      Evaluate $\delta \mathbf{x}_{i+1}^j = f_{\mathbf{x},i} \delta \mathbf{x}_i^j + f_{\mathbf{u},i} \delta \mathbf{u}_i^j$.

      $\mathbf{u}_i^{j+1} = \mathbf{u}_i^j + \delta \mathbf{u}_i^j$.

  $j = j + 1$.

While: $\max_{0 \le i < l} \|\delta \mathbf{u}_i^j\| \ge TOL$ *and* $j < MAX\_ITER$.

3.5.3 Quasilinearisation Scheme for Implicit State Equations

For stiff problems it is more convenient to discretise the state equations (3.225) using an implicit method, i.e. the implicit Euler method. Therefore, the discrete forward simulation (3.243) takes the following form:

$$F_i(\mathbf{x}_i, \mathbf{u}_i, \mathbf{x}_{i+1}) = 0, \quad i = 0, ..., l-1, \quad \mathbf{x}_0 \text{ fixed.} \tag{3.262}$$

To evaluate discrete variables $\mathbf{x}_{i+1},\ i = 0, ..., l-1$, from the equations (3.262) one can use one of available methods, i.e. the Newton method. Thus, the **Original Step** in Quasilinearisation scheme 3.7 is transformed to solving the implicit state equations $F(\mathbf{x}_i, \mathbf{u}_i, \mathbf{x}_{i+1}) = 0$ with respect to $\mathbf{x}_{i+1},\ i = 0, ..., l-1$.

The **Adjoint Step** and **Final Step** in Quasilinearisation scheme 3.7 use partial derivatives of the function $f_i(\mathbf{x}_i, \mathbf{u}_i)$ which is not explicit defined in the case of implicit state equations. Using the Implicit Function Theorem and the equality $\mathbf{x}_{i+1} = f_i(\mathbf{x}_i, \mathbf{u}_i)$, one can express partial derivatives of $f_i(\mathbf{x}_i, \mathbf{u}_i)$ as partial derivatives of the variable $\mathbf{x}_{i+1}$, which can be expressed in terms of the implicit function $F_i(\mathbf{x}_i, \mathbf{u}_i, \mathbf{x}_{i+1})$. Therefore, the explicit formulae for partial derivatives of $f_i(\mathbf{x}_i, \mathbf{u}_i)$ and for partial derivatives of the Hamiltonian function $H_i(\mathbf{x}_i, \mathbf{u}_i, \bar{\mathbf{x}}_{i+1})$ take the following form:

$$f_{\mathbf{x},i} = \frac{\partial \mathbf{x}_{i+1}}{\partial \mathbf{x}_i} = -\left(\frac{\partial F_i}{\partial \mathbf{x}_{i+1}}\right)^{-1} \frac{\partial F_i}{\partial \mathbf{x}_i} = H_{\bar{\mathbf{x}}\mathbf{x}}^\top, \tag{3.263}$$

$$f_{\mathbf{u},i} = \frac{\partial \mathbf{x}_{i+1}}{\partial \mathbf{u}_i} = -\left(\frac{\partial F_i}{\partial \mathbf{x}_{i+1}}\right)^{-1} \frac{\partial F_i}{\partial \mathbf{u}_i} = H_{\bar{\mathbf{x}}\mathbf{u}}^\top, \tag{3.264}$$

$$H_{\mathbf{u},i} = \bar{\mathbf{x}}_{i+1}^\top \frac{\partial \mathbf{x}_{i+1}}{\partial \mathbf{u}_i} = \bar{\mathbf{x}}_{i+1}^\top \left(-\frac{\partial F_i}{\partial \mathbf{x}_{i+1}}\right)^{-1} \frac{\partial F_i}{\partial \mathbf{u}_i}, \tag{3.265}$$

$$H_{\mathbf{x},i} = \bar{\mathbf{x}}_{i+1}^\top \frac{\partial \mathbf{x}_{i+1}}{\partial \mathbf{x}_i} = \bar{\mathbf{x}}_{i+1}^\top \left(-\frac{\partial F_i}{\partial \mathbf{x}_{i+1}}\right)^{-1} \frac{\partial F_i}{\partial \mathbf{x}_i}, \tag{3.266}$$

$$\begin{aligned} H_{\mathbf{uu},i} = {} & \bar{\mathbf{x}}_{i+1}^\top \left(\tfrac{\partial F_i}{\partial \mathbf{x}_{i+1}}\right)^{-2} \left(\tfrac{\partial F_i}{\partial \mathbf{x}_{i+1}\partial \mathbf{u}_i} + \tfrac{\partial F_i}{\partial \mathbf{x}_{i+1}\partial \mathbf{x}_{i+1}} \tfrac{\partial \mathbf{x}_{i+1}}{\partial \mathbf{u}_i}\right) \left(\tfrac{\partial F_i}{\partial \mathbf{u}_i}\right) + \\ & + \bar{\mathbf{x}}_{i+1}^\top \left(-\tfrac{\partial F_i}{\partial \mathbf{x}_{i+1}}\right)^{-1} \left(\tfrac{\partial F_i}{\partial \mathbf{u}_i \partial \mathbf{u}_i} + \tfrac{\partial F_i}{\partial \mathbf{u}_i \partial \mathbf{x}_{i+1}} \tfrac{\partial \mathbf{x}_{i+1}}{\partial \mathbf{u}_i}\right), \end{aligned} \tag{3.267}$$

(3.268)
$$H_{\mathbf{ux},i} = \bar{\mathbf{x}}_{i+1}^{\top}\left(\frac{\partial F_i}{\partial \mathbf{x}_{i+1}}\right)^{-2}\left(\frac{\partial F_i}{\partial \mathbf{x}_{i+1}\partial \mathbf{x}_i} + \frac{\partial F_i}{\partial \mathbf{x}_{i+1}\partial \mathbf{x}_{i+1}}\frac{\partial \mathbf{x}_{i+1}}{\partial \mathbf{x}_i}\right)\left(\frac{\partial F_i}{\partial \mathbf{u}_i}\right) + \\ + \bar{\mathbf{x}}_{i+1}^{\top}\left(-\frac{\partial F_i}{\partial \mathbf{x}_{i+1}}\right)^{-1}\left(\frac{\partial F_i}{\partial \mathbf{u}_i\partial \mathbf{x}_i} + \frac{\partial F_i}{\partial \mathbf{u}_i\partial \mathbf{x}_{i+1}}\frac{\partial \mathbf{x}_{i+1}}{\partial \mathbf{x}_i}\right),$$

(3.269)
$$H_{\mathbf{xx},i} = \bar{\mathbf{x}}_{i+1}^{\top}\left(\frac{\partial F_i}{\partial \mathbf{x}_{i+1}}\right)^{-2}\left(\frac{\partial F_i}{\partial \mathbf{x}_{i+1}\partial \mathbf{x}_i} + \frac{\partial F_i}{\partial \mathbf{x}_{i+1}\partial \mathbf{x}_{i+1}}\frac{\partial \mathbf{x}_{i+1}}{\partial \mathbf{x}_i}\right)\left(\frac{\partial F_i}{\partial \mathbf{x}_i}\right) + \\ + \bar{\mathbf{x}}_{i+1}^{\top}\left(-\frac{\partial F_i}{\partial \mathbf{x}_{i+1}}\right)^{-1}\left(\frac{\partial F_i}{\partial \mathbf{x}_i\partial \mathbf{x}_i} + \frac{\partial F_i}{\partial \mathbf{x}_i\partial \mathbf{x}_{i+1}}\frac{\partial \mathbf{x}_{i+1}}{\partial \mathbf{x}_i}\right).$$

The expressions for $\frac{\partial \mathbf{x}_{i+1}}{\partial \mathbf{x}_i}$ and $\frac{\partial \mathbf{x}_{i+1}}{\partial \mathbf{u}_i}$, involved in the equations (3.267) - (3.269), are given by the relations (3.263) and (3.264), respectively.

Summarising, the Quasilinearisation Scheme for implicit state equations differs from the Quasilinearisation Scheme for explicit state equations given in Algorithm 3.7 in solving the implicit equations $F(\mathbf{x}_i, \mathbf{u}_i, \mathbf{x}_{i+1}) = 0$ with respect to $\mathbf{x}_{i+1}$, $i = 0, ..., l-1$ in the **Original Step**. Moreover, in the following this uses the partial derivatives of the functions $f_i(\mathbf{x}_i, \mathbf{u}_i)$ and partial derivatives of the Hamiltonian function $H_i(\mathbf{x}_i, \mathbf{u}_i, \bar{\mathbf{x}}_{i+1})$, defined by the expressions (3.263) - (3.269).

3.5.4 Pantoja Approach

Recall from the [24, 30, 73] a discrete stage-wise second order approach for optimal control problems.

Consider a following discrete optimal control problem:

(3.270)
$$\min_{\mathbf{x},\mathbf{u}} \phi(\mathbf{x}_l),$$

with

(3.271)
$$\mathbf{x}_{i+1} = f_i(\mathbf{x}_i, \mathbf{u}_i), \qquad 0 \le i < l, \qquad \mathbf{x}_0 \text{ fixed.}$$

Define adjoint equations by

(3.272)
$$\bar{\mathbf{x}}_i = f_{\mathbf{x},i}^{\top}(\mathbf{x}_i, \mathbf{u}_i)\bar{\mathbf{x}}_{i+1}, \qquad 0 \le i < l, \qquad \bar{\mathbf{x}}_l = \phi_x^{\top}(\mathbf{x}_l).$$

Suppose, we have a starting solution $\mathbf{u}$. In order to find the Newton direction $\delta\mathbf{u}$, so that

(3.273)
$$\left[\frac{\partial^2 \phi}{\partial \mathbf{u}^2}\right]\delta\mathbf{u} = -\left[\frac{\partial \phi}{\partial \mathbf{u}}\right],$$

Algorithm 3.8 Pantoja Algorithm

Choose initial control trajectory $\mathbf{u}_0^0$, *...*, $\mathbf{u}_{l-1}^0$, $j = 0$.

Do:

  Original Initialisation:

  $\mathbf{x}_0^j = \mathbf{x}_0$.

  For $i = 0, 1, ..., l-1$ **DO**

    Original Step:

      $\mathbf{x}_{i+1}^j = f_i(\mathbf{x}_i^j, \mathbf{u}_i^j)$.

  Adjoint Initialisation:

    Set $\bar{\mathbf{x}}_l^j = \eta_l = \phi_{\mathbf{x}}^\top(\mathbf{x}_l^j) \in \mathbf{R}^n$.

    Set $K_l = \phi_{\mathbf{xx}}^\top(\mathbf{x}_l^j) \in \mathbf{R}^{n \times n}$.

  For $i = l-1, l-2, ..., 0$ **DO**

    Adjoint Step:

      Evaluate $\bar{\mathbf{x}}_i = f_{\mathbf{x},i}^\top \bar{\mathbf{x}}_{i+1} \in \mathbf{R}^n$.

      Evaluate $A_i = f_{\mathbf{x},i}^\top K_{i+1} f_{\mathbf{x},i} + f_{\mathbf{xx},i}^\top \bar{\mathbf{x}}_{i+1} \in \mathbf{R}^{n \times n}$.

      Evaluate $B_i = f_{\mathbf{u},i}^\top K_{i+1} f_{\mathbf{x},i} + f_{\mathbf{ux},i}^\top \bar{\mathbf{x}}_{i+1} \in \mathbf{R}^{m \times n}$.

      Evaluate $C_i = f_{\mathbf{u},i}^\top K_{i+1} f_{\mathbf{u},i} + f_{\mathbf{uu},i}^\top \bar{\mathbf{x}}_{i+1} \in \mathbf{R}^{m \times m}$.

      If C_i *is singular, the algorithm fails.*

      Else

        Evaluate $K_i = A_i - B_i^\top C_i^{-1} B_i \in \mathbf{R}^{n \times n}$.

        Evaluate $c_i = f_{\mathbf{u},i}^\top \eta_{i+1} \in \mathbf{R}^m$.

        Evaluate $\eta_i = f_{\mathbf{x},i}^\top \eta_{i+1} - B_i^\top C_i^{-1} c_i \in \mathbf{R}^n$.

  Final Initialisation:

    Set $\delta\mathbf{x}_0^j = 0$.

  For $i = 0, 1, ..., l-1$ **DO**

    Final Step:

      Evaluate $\delta\mathbf{u}_i^j = -C_i^{-1}(B_i \delta\mathbf{x}_i^j + c_i) \in \mathbf{R}^m$.

      Evaluate $\delta\mathbf{x}_{i+1}^j = f_{\mathbf{x},i}\delta\mathbf{x}_i^j + f_{\mathbf{u},i}\delta\mathbf{u}_i^j \in \mathbf{R}^n$.

      $\mathbf{u}_i^{j+1} = \mathbf{u}_i^j + \delta\mathbf{u}_i^j$.

  $j = j + 1$.

While: $\max\limits_{0 \le i < l} \|\delta\mathbf{u}_i^j\| \ge$ *TOL and* $j <$ *MAX_ITER.*

we first linearise both the original and the adjoint problems at the starting solution $\mathbf{u}$. The details for constructing the Newton direction $\delta\mathbf{u}$ are summarised in Algorithm 3.8.

It was described in [24] how Pantoja Algorithm 3.8 can be efficiently implemented using AD techniques. From Algorithm 3.8 we can see, that, similar to Algorithms 3.5, 3.6, and 3.7, each iteration of Pantoja approach consists of three sweeps through the time window $[0, T]$, which are referred to as **original**, **adjoint**, and **final** sweep. In Figure 3.14 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 represented by slanted lines. Two cameras pointing at the original and the adjoint sweeps represent the information which has to be stored, if the current composite state is saved as a checkpoint. Here, $B(t)$ is a $n \times m$ matrix path that must be communicated from the adjoint to the final sweep. In any case, the adjoint sweep causes much more computational effort as

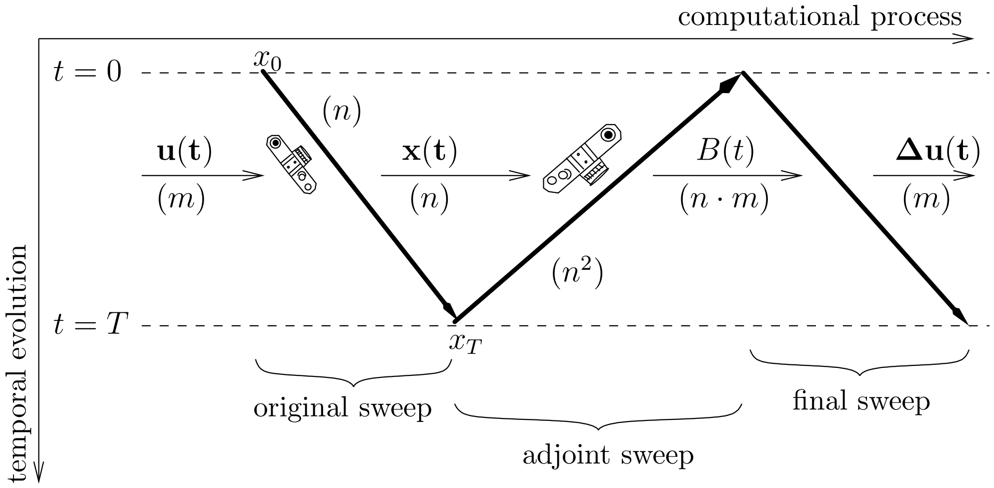

Figure 3.14: Information flow by Riccati/Pantoja computation of Newton step

the original and the final sweeps, because it involves matrix computations and factorisations. 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 original sweep of a subsequel Newton step.

The simplest strategy is to implement the Algorithms 3.5, 3.6, 3.7, and 3.8 straightforwardly storing all intermediate states of each sweep on a sequential data file and to restore them, if it is required. 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 realised 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 [34, 35, 83], 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 adjoint, a reverse sweep. But because of the triple sweep within each Newton iteration (see Figure 3.14), we are faced here with a new kind of checkpointing. Since now checkpoints from various sweeps must be kept simultaneously, we refer to this situation as **nested checkpointing**.

Christianson [24] has proposed a nested checkpoint strategy, which reduces the memory requirement from $n^2\,l$ for the basic algorithm to $n^2\sqrt{l}$ units. In this approach, data of each $\sqrt{l}$-th intermediate state on the original and adjoint sweeps is stored as a checkpoint. Subproblems of length $\sqrt{l}$ are handled using the basic algorithm. In Chapter 4 will be demonstrated that the dependence on l can be arranged polylogarithmically [34] by nested checkpoint strategies. Consequently, the operations count also grows as a small power of $\ln l$, which need however not result in an increase of the actual run-time due to memory effect. Moreover, we propose in Chapter 4 an optimal checkpoint strategy, which features these properties.

Similar to the Quasilinearisation Scheme for implicit state equations the original Pantoja Approach can be generalised to the Pantoja Approach for implicit state equations. In this case the implicit state equations $F(\mathbf{x}_i, \mathbf{u}_i, \mathbf{x}_{i+1}) = 0$ are to be solved with respect to $\mathbf{x}_{i+1},\ i = 0, ..., l-1$ in the **Original Step** of Algorithm 3.8. Furthermore, using the following formulae

$$\begin{aligned}
H^\top_{\mathbf{xx},i} &= f^\top_{\mathbf{xx},i}\bar{\mathbf{x}}_{i+1},\\
H^\top_{\mathbf{ux},i} &= f^\top_{\mathbf{ux},i}\bar{\mathbf{x}}_{i+1},\\
H^\top_{\mathbf{uu},i} &= f^\top_{\mathbf{uu},i}\bar{\mathbf{x}}_{i+1},
\end{aligned}$$

and the relations (3.263) - (3.269) for partial derivatives of the functions $f_i(\mathbf{x}_i, \mathbf{u}_i)$ and partial derivatives of the Hamiltonian function $H_i(\mathbf{x}_i, \mathbf{u}_i, \bar{\mathbf{x}}_{i+1})$, the **Adjoint Step** and the **Final Step** of Algorithm 3.8 can be performed.

3.5.5 Correspondence Between Quasilinearisation and Pantoja Approach

To show the correspondence between quasilinearisation and Pantoja approach, we rephrase slightly the first of them.

We write for convenience the linearised discrete Euler-Lagrange equations for the optimal control problem (3.242) - (3.243):

$$\delta\mathbf{x}_{i+1} - f_{\mathbf{x},i}\delta\mathbf{x}_i - f_{\mathbf{u},i}\delta\mathbf{u}_i = -(\mathbf{x}_{i+1} - f_i) = 0, \tag{3.274}$$

$$\delta\bar{\mathbf{x}}_i - H^\top_{\mathbf{xx},i}\delta\mathbf{x}_i - H^\top_{\mathbf{xu},i}\delta\mathbf{u}_i - H^\top_{\mathbf{x}\bar{\mathbf{x}},i}\delta\bar{\mathbf{x}}_{i+1} = -(\bar{\mathbf{x}}_i - H^\top_{\mathbf{x},i}) = 0, \tag{3.275}$$

$$H^\top_{\mathbf{u},i} + H^\top_{\mathbf{ux},i}\,\delta\mathbf{x}_i + H^\top_{\mathbf{uu},i}\,\delta\mathbf{u}_i + H^\top_{\mathbf{u}\bar{\mathbf{x}},i}\,\delta\bar{\mathbf{x}}_{i+1} = 0, \tag{3.276}$$

with the BC

$$\begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} \delta\mathbf{x}_0 \\ \delta\bar{\mathbf{x}}_0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ -\phi^\top_{\mathbf{xx}}(\mathbf{x}_l) & I \end{pmatrix}\begin{pmatrix} \delta\mathbf{x}_l \\ \delta\bar{\mathbf{x}}_l \end{pmatrix} = -\begin{pmatrix} \mathbf{x}_0 - \mathbf{x}_0 \\ \bar{\mathbf{x}}_l - \phi^\top_{\mathbf{x}}(\mathbf{x}_l) \end{pmatrix}. \tag{3.277}$$

If we introduce the linear transformation

$$\delta\bar{\mathbf{x}}_{i+1} = a_{i+1} + K_{i+1}\delta\mathbf{x}_{i+1}, \tag{3.278}$$

and express $\delta\mathbf{x}_{i+1}$ in terms of $\delta\mathbf{x}_i$ and $\delta\mathbf{u}_i$, using the formula (3.274), equation (3.276) takes the form

$$H^\top_{\mathbf{u},i} + f^\top_{\mathbf{u},i}\,a_{i+1} + \left(H^\top_{\mathbf{ux},i} + f^\top_{\mathbf{u},i}K_{i+1}f_{\mathbf{x},i}\right)\,\delta\mathbf{x}_i + \left(H^\top_{\mathbf{uu},i} + f^\top_{\mathbf{u},i}K_{i+1}f_{\mathbf{u},i}\right)\,\delta\mathbf{u}_i = 0. \tag{3.279}$$

Further, we express $\delta\mathbf{u}_i$ from the last equation (3.279) in terms of $\delta\mathbf{x}_i$, a_{i+1}, and K_{i+1} in the following way

$$\delta\mathbf{u}_i = -C_i^{-1}\left(H^\top_{\mathbf{u},i} + f^\top_{\mathbf{u},i}a_{i+1}\right) - C_i^{-1}\,B_i\,\delta\mathbf{x}_i, \tag{3.280}$$

and substitute this expression in (3.274) and (3.275), using the linear transformation (3.278) for $\delta\bar{\mathbf{x}}_{i+1}$. After that, we obtain

$$\begin{aligned} \delta\mathbf{x}_{i+1} &= \left(f_{\mathbf{x},i} - f_{\mathbf{u},i}\,C_i^{-1}\,B_i\right)\,\delta\mathbf{x}_i - f_{\mathbf{u},i}C_i^{-1}\,\left(H^\top_{\mathbf{u},i} + f^\top_{\mathbf{u},i}a_{i+1}\right), \\ \delta\mathbf{x}_0 &= 0, \end{aligned} \tag{3.281}$$

and

$$\begin{aligned} \delta\bar{\mathbf{x}}_i = a_i + K_i\,\delta\mathbf{x}_i &= f^\top_{\mathbf{x},i}\,a_{i+1} - B_i^\top C_i^{-1}\,\left(H^\top_{\mathbf{u},i} + f^\top_{\mathbf{u},i}\,a_{i+1}\right) + \\ &+ \left(A_i - B_i^\top\,C_i^{-1}\,B_i\right)\,\delta\mathbf{x}_i, \end{aligned} \tag{3.282}$$

where

$$A_i = H^{\top}_{\mathbf{xx},i} + f^{\top}_{\mathbf{x},i} K_{i+1} f_{\mathbf{x},i}, \tag{3.283}$$

$$B_i = H^{\top}_{\mathbf{ux},i} + f^{\top}_{\mathbf{u},i} K_{i+1} f_{\mathbf{x},i}, \tag{3.284}$$

$$C_i = H^{\top}_{\mathbf{uu},i} + f^{\top}_{\mathbf{u},i} K_{i+1} f_{\mathbf{u},i}. \tag{3.285}$$

Since (3.282) has to be satisfied for any arbitrary $\delta\mathbf{x}_i$, we obtain a pair of two-point boundary problems for a_i

$$a_i = f^{\top}_{\mathbf{x},i}\, a_{i+1} - B_i^{\top} C_i^{-1} \left(H^{\top}_{\mathbf{u},i} + f^{\top}_{\mathbf{u},i}\, a_{i+1}\right), \qquad a_l = 0, \tag{3.286}$$

and for K_i

$$K_i = A_i - B_i^{\top} C_i^{-1} B_i, \qquad K_l = \phi^{\top}_{\mathbf{xx}}(\mathbf{x}_l). \tag{3.287}$$

This approach leads to the following conceptual algorithm.

Algorithm 3.9 Transformed Pantoja Algorithm
Choose initial control trajectory $\mathbf{u}_0^0$, ..., $\mathbf{u}_{l-1}^0$, $j = 0$.
Do:
  Original Initialisation:
  $\mathbf{x}_0^j = \mathbf{x}_0$.
  For $i = 0, 1, ..., l-1$ **DO**
    Original Step:
      $\mathbf{x}_{i+1}^j = f_i(\mathbf{x}_i^j, \mathbf{u}_i^j)$.
  Adjoint Initialisation:
    Set $\bar{\mathbf{x}}_l^j = \phi_{\mathbf{x}}^\top(\mathbf{x}_l^j) \in \mathbf{R}^n$.
    Set $a_l = 0 \in \mathbf{R}^n$.
    Set $K_l = \phi_{\mathbf{xx}}^\top(\mathbf{x}_l^j) \in \mathbf{R}^{n\times n}$.
  For $i = l-1, l-2, ..., 0$ **DO**
    Adjoint Step:
      Evaluate $\bar{\mathbf{x}}_i = f_{\mathbf{x},i}^\top \bar{\mathbf{x}}_{i+1} \in \mathbf{R}^n$.
      Evaluate $A_i = H_{\mathbf{xx},i}^\top + f_{\mathbf{x},i}^\top\, K_{i+1}\, f_{\mathbf{x},i} \in \mathbf{R}^{n\times n}$.
      Evaluate $B_i = H_{\mathbf{ux},i}^\top + f_{\mathbf{u},i}^\top\, K_{i+1}\, f_{\mathbf{x},i} \in \mathbf{R}^{m\times n}$.
      Evaluate $C_i = H_{\mathbf{uu},i}^\top + f_{\mathbf{u},i}^\top\, K_{i+1}\, f_{\mathbf{u},i} \in \mathbf{R}^{m\times m}$.
      If C_i *is singular, the algorithm fails.*
      Else
        Evaluate $K_i = A_i - B_i^\top\, C_i^{-1}\, B_i \in \mathbf{R}^{n\times n}$.
        Evaluate $a_i = f_{\mathbf{x},i}^\top\, a_{i+1} - B_i^\top C_i^{-1} \left(H_{\mathbf{u},i}^\top + f_{\mathbf{u},i}^\top\, a_{i+1}\right) \in \mathbf{R}^n$.
  Final Initialisation:
    Set $\delta\mathbf{x}_0^j = 0$.
  For $i = 0, 1, ..., l-1$ **DO**
    Final Step:
      Evaluate $\delta\mathbf{u}_i^j = -C_i^{-1}\left(H_{\mathbf{u},i}^\top + f_{\mathbf{u},i}^\top a_{i+1}\right) - C_i^{-1}\, B_i\, \delta\mathbf{x}_i \in \mathbf{R}^m$.
      Evaluate $\delta\mathbf{x}_{i+1}^j = f_{\mathbf{x},i}\delta\mathbf{x}_i^j + f_{\mathbf{u},i}\delta\mathbf{u}_i^j \in \mathbf{R}^n$.
      $\mathbf{u}_i^{j+1} = \mathbf{u}_i^j + \delta\mathbf{u}_i^j$.
  $j = j + 1$.
While: $\max\limits_{0\le i<l} \|\delta\mathbf{u}_i^j\| \ge$ *TOL and* $j <$ *MAX_ITER*.

To prove the equivalence of Quasilinearisation scheme 3.7 and Pantoja Algorithm 3.8, we prove at first the correspondence between Transformed Pantoja Algorithm 3.9 and Pantoja Algorithm 3.8. This can be shown very easy. After that, we prove the equivalence between Transformed Pantoja Algorithm 3.9 and Algorithm 3.7. Therefore, this implies the equivalence between Quasilinearisation scheme 3.7 and Pantoja Algorithm 3.8.

Theorem 3.2 *Transformed Pantoja Algorithm 3.9 is equivalent to Pantoja Algorithm 3.8.*

Proof: To prove the assertion of Theorem 3.2, we have to show that corrections $\delta \mathbf{u}_i$, $0 \le i < l$, evaluated in Algorithm 3.8 and Algorithm 3.9, coincide. In order to make this, we note firstly that the variables $\mathbf{x}_i$, $\bar{\mathbf{x}}_i$, A_i, B_i, C_i, and K_i, evaluated in Algorithm 3.8, are equal to the variables $\mathbf{x}_i$, $\bar{\mathbf{x}}_i$, A_i, B_i, C_i, and K_i, respectively, evaluated in Algorithm 3.9 for all i. This is because

$$\begin{aligned} H^\top_{\mathbf{xx},i} &= f^\top_{\mathbf{xx},i}\bar{\mathbf{x}}_{i+1}, \\ H^\top_{\mathbf{ux},i} &= f^\top_{\mathbf{ux},i}\bar{\mathbf{x}}_{i+1}, \\ H^\top_{\mathbf{uu},i} &= f^\top_{\mathbf{uu},i}\bar{\mathbf{x}}_{i+1}. \end{aligned}$$

Furthermore, we prove the following equation

$$\bar{\mathbf{x}}_i + a_i = \eta_i, \qquad 0 \le i \le l, \tag{3.288}$$

with η_i being evaluated in Algorithm 3.8; and $\bar{\mathbf{x}}_i$ and a_i being evaluated in Algorithm 3.9. Identity (3.288) is proved by induction:

For $i = l$, we obtain

$$\bar{\mathbf{x}}_l + a_l = \phi^\top_{\mathbf{x}}(\mathbf{x}_l) + 0 = \phi^\top_{\mathbf{x}}(\mathbf{x}_l) = \eta_l.$$

Assume that equation (3.288) is true for all $\tilde{i}$ with $\tilde{i} \ge i+1$. Now, we have to show that equality (3.288) satisfies for i. Thus,

$$\begin{aligned} \bar{\mathbf{x}}_i + a_i &= f^\top_{\mathbf{x},i}\bar{\mathbf{x}}_{i+1} + f^\top_{\mathbf{x},i}\, a_{i+1} - B_i^\top C_i^{-1}\, \left(H^\top_{\mathbf{u},i} + f^\top_{\mathbf{u},i}\, a_{i+1}\right) = \\ &= f^\top_{\mathbf{x},i}\bar{\mathbf{x}}_{i+1} + f^\top_{\mathbf{x},i}\, a_{i+1} - B_i^\top C_i^{-1}\, f^\top_{\mathbf{u},i}\bar{\mathbf{x}}_{i+1} - B_i^\top C_i^{-1}\, f^\top_{\mathbf{u},i}\, a_{i+1} = \\ &= f^\top_{\mathbf{x},i}\eta_{i+1} - B_i C_i^{-1} f^\top_{\mathbf{u},i}\eta_{i+1} = f^\top_{\mathbf{x},i}\eta_{i+1} - B_i C_i^{-1} c_i = \eta_i. \end{aligned}$$

Therefore, the identity (3.288) is proved for all $i, 0 \le i \le l$. Now, identity between corrections $\delta \mathbf{u}_i$, $0 \le i < l$, evaluated in Algorithm 3.8 and Algorithm 3.9, is proved by induction as well:

For $i = 0$, we obtain

$$\begin{aligned}
\delta\mathbf{u}_0(Alg3.9) &= -C_0^{-1}\left(H_{\mathbf{u},0}^\top + f_{\mathbf{u},0}^\top a_1\right) - C_0^{-1}\, B_0\, \delta\mathbf{x}_0 = \\
&= -C_0^{-1}\left(f_{\mathbf{u},0}^\top \bar{\mathbf{x}}_1 + f_{\mathbf{u},0}^\top a_1\right) = -C_0^{-1} f_{\mathbf{u},0}^\top\left(\bar{\mathbf{x}}_1 + a_1\right) = -C_0^{-1} f_{\mathbf{u},0}^\top \eta_1,
\end{aligned}$$

since $\delta\mathbf{x}_0 = 0$, $H_{\mathbf{u},0}^\top = f_{\mathbf{u},0}^\top \bar{\mathbf{x}}_1$, and $\bar{\mathbf{x}}_1 + a_1 = \eta_1$.

$$\delta\mathbf{u}_0(Alg3.8) = -C_0^{-1}(B_0 s_0 + c_0) = -C_0^{-1} f_{\mathbf{u},0}^\top \eta_1,$$

since $s_0 = 0$ and $c_0 = f_{\mathbf{u},0}^\top \eta_1$. Therefore, $\delta\mathbf{u}_0(Alg3.9) = \delta\mathbf{u}_0(Alg3.8)$. Moreover, $s_0 = \delta\mathbf{x}_0 = 0$ and $s_1 = \delta\mathbf{x}_1$, due to the construction of s_1 and $\delta\mathbf{x}_1$.

Assume that $\delta\mathbf{u}_{\tilde{i}}(Alg3.9) = \delta\mathbf{u}_{\tilde{i}}(Alg3.8)$ is true for all $\tilde{i}$ with $\tilde{i} \leq i-1$. This implies obviously that $s_{\tilde{i}} = \delta\mathbf{x}_{\tilde{i}}$ is true for all $\tilde{i}$ with $\tilde{i} \leq i$. Now, we have to show that $\delta\mathbf{u}_i(Alg3.9) = \delta\mathbf{u}_i(Alg3.8)$ satisfies. Thus,

$$\begin{aligned}
\delta\mathbf{u}_i(Alg3.9) &= -C_i^{-1}\left(H_{\mathbf{u},i}^\top + f_{\mathbf{u},i}^\top a_{i+1}\right) - C_i^{-1}\, B_i\, \delta\mathbf{x}_i = \\
&= -C_i^{-1}\left(f_{\mathbf{u},i}^\top \bar{\mathbf{x}}_{i+1} + f_{\mathbf{u},i}^\top a_{i+1}\right) - C_i^{-1}\, B_i\, \delta\mathbf{x}_i = \\
&= -C_i^{-1} f_{\mathbf{u},i}^\top\left(\bar{\mathbf{x}}_{i+1} + a_{i+1}\right) - C_i^{-1}\, B_i\, \delta\mathbf{x}_i = \\
&= -C_i^{-1} f_{\mathbf{u},i}^\top \eta_{i+1} - C_i^{-1}\, B_i\, \delta\mathbf{x}_i,
\end{aligned}$$

since $H_{\mathbf{u},i}^\top = f_{\mathbf{u},i}^\top \bar{\mathbf{x}}_{i+1}$ and $\bar{\mathbf{x}}_{i+1} + a_{i+1} = \eta_{i+1}$.

$$\delta\mathbf{u}_i(Alg3.8) = -C_i^{-1}(B_i s_i + c_i) = -C_i^{-1} f_{\mathbf{u},i}^\top \eta_{i+1} - C_i^{-1} B_i s_i.$$

$\Rightarrow \delta\mathbf{u}_i(Alg3.8) = \delta\mathbf{u}_i(Alg3.9)$.

□

Hence, we have proved that Algorithm 3.9 is equivalent to Pantoja Algorithm 3.8. Now, we prove that Algorithm 3.9 is equivalent to Quasilinearisation scheme 3.7 as well. From the construction of Quasilinearisation scheme 3.7 and Algorithm 3.9 it is apparent how they differ from each over. In Quasilinearisation scheme 3.7 we firstly express the correction $\delta\mathbf{u}$ in terms of $\delta\mathbf{x}$ and $\delta\bar{\mathbf{x}}$, using (3.233) in the continuous and (3.252) in the discrete case, respectively. After that, we construct linear equations for $\delta\mathbf{x}$ and $\delta\bar{\mathbf{x}}$ (see (3.234) and (3.253)). Finally, introducing a new variable a instead of $\delta\bar{\mathbf{x}}$ in (3.237) and (3.256), we obtain a linear equations for $\delta\mathbf{x}$ and a (see (3.240) and (3.258)). In Algorithm 3.9 we firstly introduce a new variable a instead of $\delta\bar{\mathbf{x}}$ in (3.278). After that, expressing the correction $\delta\mathbf{u}$ in terms of $\delta\mathbf{x}$ and the new variable a in (3.280), we obtain linear equations for $\delta\mathbf{x}$ and a (see (3.281) and (3.286)). To prove the equivalence of Algorithm 3.7 and Algorithm 3.9 more formally, see the following theorem.

Theorem 3.3 *Transformed Pantoja Algorithm 3.9 is equivalent to Quasilinearisation scheme 3.7.*

Proof:

To prove the assertion of Theorem 3.3, we have firstly to show that values for a_i and K_i, evaluated in Algorithm 3.7 and Algorithm 3.9 for i, $0 \le i \le l$, coincide, i.e.

$$a_i(Alg3.7) = a_i(Alg3.9) \text{ and } K_i(Alg3.7) = K_i(Alg3.9),\ 0 \le i \le l. \tag{3.289}$$

Identity (3.289) can be proved by induction:

For $i = l$, we obtain

$$a_l(Alg3.7) = a_l(Alg3.9) = 0.$$

$$K_l(Alg3.7) = K_l(Alg3.9) = \phi^{\top}_{\mathbf{xx}}(\mathbf{x}_l).$$

Assume that equation (3.289) is true for all $\tilde{i}$ with $\tilde{i} \ge i+1$. Now, we have to show that equality (3.289) satisfies for i. Firstly, we show this for a_i. For this purpose we write the both equations in terms of original functions. Therefore, from Algorithm 3.9 we have

(3.290)
$$\begin{aligned} a_i(Alg3.9) &= \Big(f^{\top}_{\mathbf{x},i} - \Big(H^{\top}_{\mathbf{xu},i} + f^{\top}_{\mathbf{x},i}\, K_{i+1}\, f_{\mathbf{u},i}\Big) \times \\ &\times \Big(H^{\top}_{\mathbf{uu},i} + f^{\top}_{\mathbf{u},i}\, K_{i+1}\, f_{\mathbf{u},i}\Big)^{-1} f^{\top}_{\mathbf{u},i}\Big)\, a_{i+1} - \\ &- \Big(H^{\top}_{\mathbf{xu},i} + f^{\top}_{\mathbf{x},i}\, K_{i+1}\, f_{\mathbf{u},i}\Big)\Big(H^{\top}_{\mathbf{uu},i} + f^{\top}_{\mathbf{u},i}\, K_{i+1}\, f_{\mathbf{u},i}\Big)^{-1} H^{\top}_{\mathbf{u},i}. \end{aligned}$$

From Algorithm 3.7 we obtain, respectively,

(3.291)
$$\begin{aligned} a_i(Alg3.7) &= \Big(H^{\top}_{\mathbf{xu},i} H^{-\top}_{\mathbf{uu},i} f^{\top}_{\mathbf{u},i} - f^{\top}_{\mathbf{x},i}\Big)\Big(-K_{i+1} f_{\mathbf{u},i} H^{-\top}_{\mathbf{uu},i} f^{\top}_{\mathbf{u},i} - I\Big)^{-1} a_{i+1} - \\ &- H^{\top}_{\mathbf{xu},i} H^{-\top}_{\mathbf{uu},i} H^{\top}_{\mathbf{u},i} - \Big(H^{\top}_{\mathbf{xu},i} H^{-\top}_{\mathbf{uu},i} f^{\top}_{\mathbf{u},i} - f^{\top}_{\mathbf{x},i}\Big) \times \\ &\times \Big(-K_{i+1} f_{\mathbf{u},i} H^{-\top}_{\mathbf{uu},i} f^{\top}_{\mathbf{u},i} - I\Big)^{-1} K_{i+1} f_{\mathbf{u},i} H^{-\top}_{\mathbf{uu},i} H^{\top}_{\mathbf{u},i}. \end{aligned}$$

We compare the appropriate components of the equations (3.291) and (3.290). First of all, we compare the components of a_{i+1} in these equations. Therefore, we have to prove the following:

$$\begin{aligned} &\Big(f^{\top}_{\mathbf{x},i} - \big(H^{\top}_{\mathbf{xu},i} + f^{\top}_{\mathbf{x},i}\, K_{i+1}\, f_{\mathbf{u},i}\big)\,\big(H^{\top}_{\mathbf{uu},i} + f^{\top}_{\mathbf{u},i}\, K_{i+1}\, f_{\mathbf{u},i}\big)^{-1} f^{\top}_{\mathbf{u},i}\Big) = \\ &= \big(H^{\top}_{\mathbf{xu},i} H^{-\top}_{\mathbf{uu},i} f^{\top}_{\mathbf{u},i} - f^{\top}_{\mathbf{x},i}\big)\big(-K_{i+1} f_{\mathbf{u},i} H^{-\top}_{\mathbf{uu},i} f^{\top}_{\mathbf{u},i} - I\big)^{-1}. \end{aligned} \tag{3.292}$$

To show it, we apply the Woodbury Formula

$$(A + UV^\top)^{-1} = A^{-1} - \left(A^{-1}U(I + V^\top A^{-1}U)^{-1}V^\top A^{-1}\right). \tag{3.293}$$

Then, we have

$$\begin{aligned} & f^\top_{\mathbf{x},i}\left(I - K_{i+1}f_{\mathbf{u},i}\left(H^\top_{\mathbf{uu},i} + f^\top_{\mathbf{u},i}\,K_{i+1}\,f_{\mathbf{u},i}\right)^{-1}\ f^\top_{\mathbf{u},i}\right) - \\ & -H^\top_{\mathbf{xu},i}\left(H^\top_{\mathbf{uu},i} + f^\top_{\mathbf{u},i}\,K_{i+1}\,f_{\mathbf{u},i}\right)^{-1}\ f^\top_{\mathbf{u},i} = \\ & = \left(f^\top_{\mathbf{x},i} - H^\top_{\mathbf{xu},i}H^{-\top}_{\mathbf{uu},i}f^\top_{\mathbf{u},i}\right)\left(I + K_{i+1}f_{\mathbf{u},i}H^{-\top}_{\mathbf{uu},i}f^\top_{\mathbf{u},i}\right)^{-1}. \end{aligned} \tag{3.294}$$

Using the Woodbury formula, we obtain

$$\begin{aligned} & \left(I - K_{i+1}f_{\mathbf{u},i}\left(H^\top_{\mathbf{uu},i} + f^\top_{\mathbf{u},i}\,K_{i+1}\,f_{\mathbf{u},i}\right)^{-1}\ f^\top_{\mathbf{u},i}\right) = \\ & \left(I - K_{i+1}f_{\mathbf{u},i}\left(H^\top_{\mathbf{uu},i}(I + H^{-\top}_{\mathbf{uu},i}f^\top_{\mathbf{u},i}\,K_{i+1}\,f_{\mathbf{u},i})\right)^{-1}\ f^\top_{\mathbf{u},i}\right) = \\ & \left(I - K_{i+1}f_{\mathbf{u},i}\left(I + H^{-\top}_{\mathbf{uu},i}f^\top_{\mathbf{u},i}\,K_{i+1}\,f_{\mathbf{u},i}\right)^{-1}\ H^{-\top}_{\mathbf{uu},i}f^\top_{\mathbf{u},i}\right) = \\ & \left(I + K_{i+1}f_{\mathbf{u},i}H^{-\top}_{\mathbf{uu},i}f^\top_{\mathbf{u},i}\right)^{-1}. \end{aligned} \tag{3.295}$$

After the substitution of (3.295) in the left hand side of (3.294), we obtain

$$\begin{aligned} & f^\top_{\mathbf{x},i}\left(I + K_{i+1}f_{\mathbf{u},i}H^{-\top}_{\mathbf{uu},i}f^\top_{\mathbf{u},i}\right)^{-1} - \\ & -H^\top_{\mathbf{xu},i}\left(I + H^{-\top}_{\mathbf{uu},i}f^\top_{\mathbf{u},i}\,K_{i+1}\,f_{\mathbf{u},i}\right)^{-1}H^{-\top}_{\mathbf{uu},i}\,f^\top_{\mathbf{u},i} = \\ & = f^\top_{\mathbf{x},i}\left(I + K_{i+1}f_{\mathbf{u},i}H^{-\top}_{\mathbf{uu},i}f^\top_{\mathbf{u},i}\right)^{-1} - \\ & -H^\top_{\mathbf{xu},i}H^{-\top}_{\mathbf{uu},i}f^\top_{\mathbf{u},i}\left(I + K_{i+1}f_{\mathbf{u},i}H^{-\top}_{\mathbf{uu},i}f^\top_{\mathbf{u},i}\right)^{-1}. \end{aligned} \tag{3.296}$$

Therefore, to prove (3.296) we have to show

$$\left(I + H^{-\top}_{\mathbf{uu},i}f^\top_{\mathbf{u},i}\,K_{i+1}\,f_{\mathbf{u},i}\right)^{-1}H^{-\top}_{\mathbf{uu},i}\,f^\top_{\mathbf{u},i} - H^{-\top}_{\mathbf{uu},i}f^\top_{\mathbf{u},i}\left(I + K_{i+1}f_{\mathbf{u},i}H^{-\top}_{\mathbf{uu},i}f^\top_{\mathbf{u},i}\right)^{-1} = 0. \tag{3.297}$$

Applying the Woodbury formula to the left hand side of (3.297), we obtain

$$\begin{aligned} & \left(I + H^{-\top}_{\mathbf{uu},i}f^\top_{\mathbf{u},i}\,K_{i+1}\,f_{\mathbf{u},i}\right)^{-1}H^{-\top}_{\mathbf{uu},i}\,f^\top_{\mathbf{u},i} = \\ & = \left(I - \left(H^{-\top}_{\mathbf{uu},i}f^\top_{\mathbf{u},i}(I + K_{i+1}\,f_{\mathbf{u},i}H^{-\top}_{\mathbf{uu},i}f^\top_{\mathbf{u},i})^{-1}K_{i+1}\,f_{\mathbf{u},i}\right)\right)H^{-\top}_{\mathbf{uu},i}\,f^\top_{\mathbf{u},i} \end{aligned} \tag{3.298}$$

Then, (3.297) is equivalent to

$$\begin{aligned} H^{-\top}_{\mathbf{uu},i}\,f^\top_{\mathbf{u},i} &- \left(H^{-\top}_{\mathbf{uu},i}f^\top_{\mathbf{u},i}(I + K_{i+1}\,f_{\mathbf{u},i}H^{-\top}_{\mathbf{uu},i}f^\top_{\mathbf{u},i})^{-1}K_{i+1}\,f_{\mathbf{u},i}\right)H^{-\top}_{\mathbf{uu},i}\,f^\top_{\mathbf{u},i} - \\ &- H^{-\top}_{\mathbf{uu},i}f^\top_{\mathbf{u},i}\left(I + K_{i+1}f_{\mathbf{u},i}H^{-\top}_{\mathbf{uu},i}f^\top_{\mathbf{u},i}\right)^{-1} = \\ &= H^{-\top}_{\mathbf{uu},i}\,f^\top_{\mathbf{u},i} - H^{-\top}_{\mathbf{uu},i}f^\top_{\mathbf{u},i}(I + K_{i+1}\,f_{\mathbf{u},i}H^{-\top}_{\mathbf{uu},i}f^\top_{\mathbf{u},i})^{-1}\times \\ &\times \left(I + K_{i+1}\,f_{\mathbf{u},i}H^{-\top}_{\mathbf{uu},i}\,f^\top_{\mathbf{u},i}\right) = \\ &= H^{-\top}_{\mathbf{uu},i}\,f^\top_{\mathbf{u},i} - H^{-\top}_{\mathbf{uu},i}\,f^\top_{\mathbf{u},i} = 0. \end{aligned} \tag{3.299}$$

Therefore, (3.296) and, consequently, the (3.292) is proven. Furthermore, to prove that (3.290) is equivalent to (3.291), we have to show

$$\begin{aligned} & - \left(H^{\top}_{\mathbf{xu},i} + f^{\top}_{\mathbf{x},i}\, K_{i+1}\, f_{\mathbf{u},i}\right)\left(H^{\top}_{\mathbf{uu},i} + f^{\top}_{\mathbf{u},i}\, K_{i+1}\, f_{\mathbf{u},i}\right)^{-1} H^{\top}_{\mathbf{u},i} = \\ & = -\left(H^{\top}_{\mathbf{xu},i} H^{-\top}_{\mathbf{uu},i} f^{\top}_{\mathbf{u},i} - f^{\top}_{\mathbf{x},i}\right)\left(-K_{i+1} f_{\mathbf{u},i} H^{-\top}_{\mathbf{uu},i} f^{\top}_{\mathbf{u},i} - I\right)^{-1} \times \\ & \times\ K_{i+1} f_{\mathbf{u},i} H^{-\top}_{\mathbf{uu},i} H^{\top}_{\mathbf{u},i} - H^{\top}_{\mathbf{xu},i} H^{-\top}_{\mathbf{uu},i} H^{\top}_{\mathbf{u},i}. \end{aligned} \tag{3.300}$$

The right hand side of (3.300) can be transformed, using the Woodbury formula. Therefore, we obtain

$$\begin{aligned} & -\left(f^{\top}_{\mathbf{x},i} - H^{\top}_{\mathbf{xu},i} H^{-\top}_{\mathbf{uu},i} f^{\top}_{\mathbf{u},i}\right)\left(I + K_{i+1} f_{\mathbf{u},i} H^{-\top}_{\mathbf{uu},i} f^{\top}_{\mathbf{u},i}\right)^{-1} \times \\ & \times K_{i+1} f_{\mathbf{u},i} H^{-\top}_{\mathbf{uu},i} H^{\top}_{\mathbf{u},i} - H^{\top}_{\mathbf{xu},i} H^{-\top}_{\mathbf{uu},i} H^{\top}_{\mathbf{u},i} = \\ & = -H^{\top}_{\mathbf{xu},i}\left(I - H^{-\top}_{\mathbf{uu},i} f^{\top}_{\mathbf{u},i}\left(I + K_{i+1} f_{\mathbf{u},i} H^{-\top}_{\mathbf{uu},i} f^{\top}_{\mathbf{u},i}\right)^{-1} K_{i+1} f_{\mathbf{u},i}\right) \times \\ & \times H^{-\top}_{\mathbf{uu},i} H^{\top}_{\mathbf{u},i} - f^{\top}_{\mathbf{x},i}\left(I + K_{i+1} f_{\mathbf{u},i} H^{-\top}_{\mathbf{uu},i} f^{\top}_{\mathbf{u},i}\right)^{-1} K_{i+1} f_{\mathbf{u},i} H^{-\top}_{\mathbf{uu},i} H^{\top}_{\mathbf{u},i} = \\ & = -H^{\top}_{\mathbf{xu},i}\left(I + H^{-\top}_{\mathbf{uu},i} f^{\top}_{\mathbf{u},i}\, K_{i+1}\, f_{\mathbf{u},i}\right)^{-1} H^{-\top}_{\mathbf{uu},i} H^{\top}_{\mathbf{u},i} - \\ & - f^{\top}_{\mathbf{x},i}\left(I + K_{i+1} f_{\mathbf{u},i} H^{-\top}_{\mathbf{uu},i} f^{\top}_{\mathbf{u},i}\right)^{-1} K_{i+1} f_{\mathbf{u},i} H^{-\top}_{\mathbf{uu},i} H^{\top}_{\mathbf{u},i}. \end{aligned} \tag{3.301}$$

Finally, to prove (3.300), we have to show

$$\left(I + K_{i+1} f_{\mathbf{u},i} H^{-\top}_{\mathbf{uu},i} f^{\top}_{\mathbf{u},i}\right)^{-1} K_{i+1} f_{\mathbf{u},i} - K_{i+1} f_{\mathbf{u},i}\left(I + H^{-\top}_{\mathbf{uu},i} f^{\top}_{\mathbf{u},i} K_{i+1} f_{\mathbf{u},i}\right)^{-1} = 0. \tag{3.302}$$

Applying the Woodbury formula to the left part of (3.302), we obtain

$$\begin{aligned} & \left(I + K_{i+1} f_{\mathbf{u},i} H^{-\top}_{\mathbf{uu},i} f^{\top}_{\mathbf{u},i}\right)^{-1} K_{i+1} f_{\mathbf{u},i} = \\ & = \left(I - K_{i+1} f_{\mathbf{u},i}\left(I + H^{-\top}_{\mathbf{uu},i} f^{\top}_{\mathbf{u},i} K_{i+1} f_{\mathbf{u},i}\right)^{-1} H^{-\top}_{\mathbf{uu},i} f^{\top}_{\mathbf{u},i}\right) K_{i+1} f_{\mathbf{u},i}. \end{aligned} \tag{3.303}$$

Therefore, (3.302) is equivalent to

$$\begin{aligned} & K_{i+1} f_{\mathbf{u},i} - K_{i+1} f_{\mathbf{u},i}\left(I + H^{-\top}_{\mathbf{uu},i} f^{\top}_{\mathbf{u},i} K_{i+1} f_{\mathbf{u},i}\right)^{-1} \times \\ & \times \left(I + H^{-\top}_{\mathbf{uu},i} f^{\top}_{\mathbf{u},i} K_{i+1} f_{\mathbf{u},i}\right) = K_{i+1} f_{\mathbf{u},i} - K_{i+1} f_{\mathbf{u},i} = 0. \end{aligned} \tag{3.304}$$

Thus, we have proved identity (3.289) for a_i.

Furthermore, we prove identity (3.289) for K_i. For this purpose we write the both equations in terms of original functions. From Algorithm 3.7 we have

$$\begin{aligned} K_i(Alg3.7) &= -H^{\top}_{\mathbf{xu},i}\, H^{-\top}_{\mathbf{uu},i}\, H^{\top}_{\mathbf{ux},i} + H^{\top}_{\mathbf{xx},i} - \\ & - \left(H^{\top}_{\mathbf{xu},i}\, H^{-\top}_{\mathbf{uu},i}\, f^{\top}_{\mathbf{u},i} - f^{\top}_{\mathbf{x},i}\right)\left(-I - K_{i+1} f_{\mathbf{u},i}\, H^{-\top}_{\mathbf{uu},i}\, f^{\top}_{\mathbf{u},i}\right)^{-1} \times \\ & \times\ K_{i+1}\left(f_{\mathbf{u},i}\, H^{-\top}_{\mathbf{uu},i}\, H^{\top}_{\mathbf{ux},i} - f_{\mathbf{x},i}\right). \end{aligned} \tag{3.305}$$

From Algorithm 3.9 we obtain, respectively,

$$
\begin{aligned}
K_i(Alg3.9) &= H^\top_{\mathbf{xx},i} + f^\top_{\mathbf{x},i}\, K_{i+1}\, f_{\mathbf{x},i} - \\
&- \left(H^\top_{\mathbf{xu},i} + f^\top_{\mathbf{x},i}\, K_{i+1}\, f_{\mathbf{u},i}\right) \left(H^\top_{\mathbf{uu},i} + f^\top_{\mathbf{u},i}\, K_{i+1}\, f_{\mathbf{u},i}\right)^{-1} \times \\
&\times \left(H^\top_{\mathbf{ux},i} + f^\top_{\mathbf{u},i}\, K_{i+1}\, f_{\mathbf{x},i}\right).
\end{aligned} \tag{3.306}
$$

Denote $V = \left(I + K_{i+1} f_{\mathbf{u},i}\, H^{-\top}_{\mathbf{uu},i}\, f^\top_{\mathbf{u},i}\right)$ and $W = \left(I + H^{-\top}_{\mathbf{uu},i} f^\top_{\mathbf{u},i}\, K_{i+1}\, f_{\mathbf{u},i}\right)$. Then, (3.305) can be written as

$$
\begin{aligned}
K_i &= -H^\top_{\mathbf{xu},i}\, H^{-\top}_{\mathbf{uu},i}\, H^\top_{\mathbf{ux},i} + H^\top_{\mathbf{xx},i} - \\
&- \left(f^\top_{\mathbf{x},i} - H^\top_{\mathbf{xu},i}\, H^{-\top}_{\mathbf{uu},i}\, f^\top_{\mathbf{u},i}\right) \left(I + K_{i+1} f_{\mathbf{u},i}\, H^{-\top}_{\mathbf{uu},i}\, f^\top_{\mathbf{u},i}\right)^{-1} \times \\
&\times K_{i+1} \left(f_{\mathbf{u},i}\, H^{-\top}_{\mathbf{uu},i}\, H^\top_{\mathbf{ux},i} - f_{\mathbf{x},i}\right) = \\
&= -H^\top_{\mathbf{xu},i}\, H^{-\top}_{\mathbf{uu},i}\, H^\top_{\mathbf{ux},i} + H^\top_{\mathbf{xx},i} - \left(f^\top_{\mathbf{x},i} - H^\top_{\mathbf{xu},i}\, H^{-\top}_{\mathbf{uu},i}\, f^\top_{\mathbf{u},i}\right) V^{-1} \times \\
&\times K_{i+1} \left(f_{\mathbf{u},i}\, H^{-\top}_{\mathbf{uu},i}\, H^\top_{\mathbf{ux},i} - f_{\mathbf{x},i}\right) = \\
&= -H^\top_{\mathbf{xu},i}\, H^{-\top}_{\mathbf{uu},i}\, H^\top_{\mathbf{ux},i} + H^\top_{\mathbf{xx},i} - \\
&- \left(f^\top_{\mathbf{x},i} V^{-1} K_{i+1} f_{\mathbf{u},i}\, H^{-\top}_{\mathbf{uu},i}\, H^\top_{\mathbf{ux},i} - H^\top_{\mathbf{xu},i}\, H^{-\top}_{\mathbf{uu},i}\, f^\top_{\mathbf{u},i} V^{-1} K_{i+1} \times \right. \\
&\times \left. f_{\mathbf{u},i}\, H^{-\top}_{\mathbf{uu},i}\, H^\top_{\mathbf{ux},i} - f^\top_{\mathbf{x},i} V^{-1} K_{i+1} f_{\mathbf{x},i} + H^\top_{\mathbf{xu},i}\, H^{-\top}_{\mathbf{uu},i}\, f^\top_{\mathbf{u},i} V^{-1} K_{i+1} f_{\mathbf{x},i}\right) = \\
&= H^\top_{\mathbf{xx},i} - H^\top_{\mathbf{xu},i} \left(I - H^{-\top}_{\mathbf{uu},i}\, f^\top_{\mathbf{u},i} V^{-1} K_{i+1} f_{\mathbf{u},i}\right) H^{-\top}_{\mathbf{uu},i}\, H^\top_{\mathbf{ux},i} - \\
&- f^\top_{\mathbf{x},i} V^{-1} K_{i+1} f_{\mathbf{u},i}\, H^{-\top}_{\mathbf{uu},i}\, H^\top_{\mathbf{ux},i} + \\
&+ f^\top_{\mathbf{x},i} V^{-1} K_{i+1} f_{\mathbf{x},i} - H^\top_{\mathbf{xu},i}\, H^{-\top}_{\mathbf{uu},i}\, f^\top_{\mathbf{u},i} V^{-1} K_{i+1} f_{\mathbf{x},i} = \\
&= H^\top_{\mathbf{xx},i} - H^\top_{\mathbf{xu},i} W^{-1} H^{-\top}_{\mathbf{uu},i}\, H^\top_{\mathbf{ux},i} - \\
&- f^\top_{\mathbf{x},i} V^{-1} K_{i+1} f_{\mathbf{u},i}\, H^{-\top}_{\mathbf{uu},i}\, H^\top_{\mathbf{ux},i} + \\
&+ f^\top_{\mathbf{x},i} V^{-1} K_{i+1} f_{\mathbf{x},i} - H^\top_{\mathbf{xu},i}\, H^{-\top}_{\mathbf{uu},i}\, f^\top_{\mathbf{u},i} V^{-1} K_{i+1} f_{\mathbf{x},i}.
\end{aligned} \tag{3.307}
$$

Analog, (3.306) can be written in the following way

$$
\begin{aligned}
K_i &= H^\top_{\mathbf{xx},i} + f^\top_{\mathbf{x},i}\, K_{i+1}\, f_{\mathbf{x},i} - \left(H^\top_{\mathbf{xu},i} W^{-1} + f^\top_{\mathbf{x},i}\, K_{i+1}\, f_{\mathbf{u},i} W^{-1}\right) \times \\
&\times \left(H^{-\top}_{\mathbf{uu},i} H^\top_{\mathbf{ux},i} + H^{-\top}_{\mathbf{uu},i} f^\top_{\mathbf{u},i}\, K_{i+1}\, f_{\mathbf{x},i}\right) = \\
&= H^\top_{\mathbf{xx},i} + f^\top_{\mathbf{x},i}\, K_{i+1}\, f_{\mathbf{x},i} - \left(H^\top_{\mathbf{xu},i} W^{-1} H^{-\top}_{\mathbf{uu},i} H^\top_{\mathbf{ux},i} + \right. \\
&+ f^\top_{\mathbf{x},i}\, K_{i+1}\, f_{\mathbf{u},i} W^{-1} H^{-\top}_{\mathbf{uu},i} H^\top_{\mathbf{ux},i} + H^\top_{\mathbf{xu},i} W^{-1} H^{-\top}_{\mathbf{uu},i} f^\top_{\mathbf{u},i}\, K_{i+1}\, f_{\mathbf{x},i} + \\
&+ \left. f^\top_{\mathbf{x},i}\, K_{i+1}\, f_{\mathbf{u},i} W^{-1} H^{-\top}_{\mathbf{uu},i} f^\top_{\mathbf{u},i}\, K_{i+1}\, f_{\mathbf{x},i}\right) = \\
&= H^\top_{\mathbf{xx},i} + f^\top_{\mathbf{x},i} \left(I + K_{i+1}\, f_{\mathbf{u},i} W^{-1} H^{-\top}_{\mathbf{uu},i} f^\top_{\mathbf{u},i}\right) K_{i+1}\, f_{\mathbf{x},i} - \\
&- H^\top_{\mathbf{xu},i} W^{-1} H^{-\top}_{\mathbf{uu},i} H^\top_{\mathbf{ux},i} -
\end{aligned} \tag{3.308}
$$

$$
\begin{aligned}
&- \; f_{\mathbf{x},i}^{\top}\, K_{i+1}\, f_{\mathbf{u},i} W^{-1} H_{\mathbf{uu},i}^{-\top} H_{\mathbf{ux},i}^{\top} - H_{\mathbf{xu},i}^{\top} W^{-1} H_{\mathbf{uu},i}^{-\top} f_{\mathbf{u},i}^{\top}\, K_{i+1}\, f_{\mathbf{x},i} = \\
&= \; H_{\mathbf{xx},i}^{\top} + f_{\mathbf{x},i}^{\top} V^{-1} K_{i+1}\, f_{\mathbf{x},i} - H_{\mathbf{xu},i}^{\top} W^{-1} H_{\mathbf{uu},i}^{-\top} H_{\mathbf{ux},i}^{\top} - \\
&- \; f_{\mathbf{x},i}^{\top}\, K_{i+1}\, f_{\mathbf{u},i} W^{-1} H_{\mathbf{uu},i}^{-\top} H_{\mathbf{ux},i}^{\top} - H_{\mathbf{xu},i}^{\top} W^{-1} H_{\mathbf{uu},i}^{-\top} f_{\mathbf{u},i}^{\top}\, K_{i+1}\, f_{\mathbf{x},i}.
\end{aligned}
$$

Thus, to show that (3.307) is equivalent to (3.308), and, consequently, (3.305) is equivalent to (3.306), it is leaved to show that

$$\left(I + K_{i+1} f_{\mathbf{u},i}\, H_{\mathbf{uu},i}^{-\top}\, f_{\mathbf{u},i}^{\top}\right)^{-1} K_{i+1} f_{\mathbf{u},i} = K_{i+1} f_{\mathbf{u},i} \left(I + H_{\mathbf{uu},i}^{-\top} f_{\mathbf{u},i}^{\top}\, K_{i+1}\, f_{\mathbf{u},i}\right)^{-1},$$

and

$$H_{\mathbf{uu},i}^{-\top} f_{\mathbf{u},i}^{\top} \left(I + K_{i+1} f_{\mathbf{u},i}\, H_{\mathbf{uu},i}^{-\top}\, f_{\mathbf{u},i}^{\top}\right)^{-1} = \left(I + H_{\mathbf{uu},i}^{-\top} f_{\mathbf{u},i}^{\top}\, K_{i+1}\, f_{\mathbf{u},i}\right)^{-1} H_{\mathbf{uu},i}^{-\top} f_{\mathbf{u},i}^{\top}.$$

It was already shown in (3.302) and (3.297), respectively. Therefore, we have proved identity (3.289) for K_i.

The next step to prove the assertion of Theorem 3.3, we have to show that corrections $\delta\mathbf{u}_i$, $0 \le i < l$, evaluated in Algorithm 3.7 and Algorithm 3.9, coincide. For this purpose, rewrite the expression for $\delta\mathbf{u}$ in Algorithm 3.7 as follows:

(3.309)

$$
\begin{aligned}
\delta\mathbf{u}_i \;&=\; -H_{\mathbf{uu},i}^{-\top}\left(H_{\mathbf{u},i}^{\top} + H_{\mathbf{ux},i}^{\top}\,\delta\mathbf{x}_i + H_{\mathbf{u\bar{x}},i}^{\top}\delta\bar{\mathbf{x}}_{i+1}\right) = \\
&=\; -H_{\mathbf{uu},i}^{-\top}\left(H_{\mathbf{u},i}^{\top} + H_{\mathbf{ux},i}^{\top}\,\delta\mathbf{x}_i + H_{\mathbf{u\bar{x}},i}^{\top}(a_{i+1} + K_{i+1}(f_{\mathbf{x},i}\delta\mathbf{x}_i + f_{\mathbf{u},i}\delta\mathbf{u}_i))\right).
\end{aligned}
$$

If we express $\delta\mathbf{u}_i$ from (3.309), we obtain

$$
\begin{aligned}
\delta\mathbf{u}_i \;&=\; -(H_{\mathbf{uu},i}^{\top} + f_{\mathbf{u},i}^{\top}\, K_{i+1}\, f_{\mathbf{u},i})^{-1}\left(H_{\mathbf{u},i}^{\top} + f_{\mathbf{u},i}^{\top} a_{i+1}\right) - \\
&-\; (H_{\mathbf{uu},i}^{\top} + f_{\mathbf{u},i}^{\top}\, K_{i+1}\, f_{\mathbf{u},i})^{-1}\left(H_{\mathbf{ux},i}^{\top} + f_{\mathbf{u},i}^{\top}\, K_{i+1}\, f_{\mathbf{x},i}\right)\delta\mathbf{x}_i = \\
&=\; -C_i^{-1}\left(H_{\mathbf{u},i}^{\top} + f_{\mathbf{u},i}^{\top} a_{i+1}\right) - C_i^{-1}\, B_i\, \delta\mathbf{x}_i,
\end{aligned}
\tag{3.310}
$$

which coincides with the expression for $\delta\mathbf{u}$ in Algorithm 3.9. Since $\delta\mathbf{x}_0(Alg3.7) = \delta\mathbf{x}_0(Alg3.9) = 0$, we obtain

(3.311)

$$\delta\mathbf{x}_{i+1}(Alg3.7) = \delta\mathbf{x}_{i+1}(Alg3.9) \text{ and } \delta\mathbf{u}_i(Alg3.7) = \delta\mathbf{u}_i(Alg3.9),\; 0 \le i \le l.$$

Hence, the assertion of Theorem 3.3 is proved.

□

Summarising the statements of Theorem 3.2 and Theorem 3.3, we obtain the following conclusion: Pantoja Algorithm 3.8 is equivalent to Quasilinearisation scheme 3.7.

In numerical applications it is more convenient to use Pantoja Algorithm 3.8 instead of Quasilinearisation scheme 3.7. The reason for this is that the first one requires inverse for matrices of size $m \times m$, while the last one requires inverse for matrices of size $n \times n$. Here, $\mathbf{u} \in \mathbf{R}^m$ and $\mathbf{x} \in \mathbf{R}^n$. In many applications we have the following situation

$$n >> m. \tag{3.312}$$

Obviously, it is more efficient to apply Pantoja Algorithm 3.8 to find solutions of this kind of problems.

3.6 Higher-Order One-Step Schemes

The only actual schemes we have seen so far are the trapezoidal schemes, introduced in Section 3.4. They are simple schemes, but their order is only 2, i.e. the error is $\mathcal{O}(h^2)$. If high accuracy is desired, then, higher-order schemes can be much more effective. This leads to Runge-Kutta schemes. We will concentrate on these for the remainder of this section. The following description of higher-order one-step schemes closely follows the description in [3].

3.6.1 Implicit Runge-Kutta Schemes

An **s-stage Runge-Kutta scheme** for $\dot{y} = f(t, y)$ is defined by

$$y_{i+1} = y_i + h \sum_{j=1}^{s} \beta_j f_{ij}, \qquad 0 \le i \le l-1, \tag{3.313}$$

where

$$f_{ij} = f(t_{ij}, y_i + h \sum_{k=1}^{s} \alpha_{jk} f_{ik}), \qquad 1 \le j \le s. \tag{3.314}$$

The points t_{ij} are given by

$$t_{ij} = t_i + h\rho_j, \qquad 1 \le j \le s, \qquad 0 \le i \le l-1, \tag{3.315}$$

with

$$0 \le \rho_1 \le \rho_2 \le \ldots \le \rho_s \le 1, \tag{3.316}$$

being the 'canonical points'. Thus, the points t_{ij}, which are sometimes called collocation points, are l scaled translation of the canonical set of s points $\rho_1, ..., \rho_s$ into each subinterval of the mesh π.

The subscript i is used as a mesh interval index, while j and k are stage counters within a particular subinterval. Thus, $0 \leq i \leq l-1$ and $1 \leq j, k \leq s$. We will continue to use the indices in this way throughout this section. We will use the standard Tableau 3.1 to represent Runge-Kutta schemes.

$$\begin{array}{c|ccc} \rho_1 & \alpha_{11} & \dots & \alpha_{1s} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_s & \alpha_{s1} & \dots & \alpha_{ss} \\ \hline & \beta_1 & \dots & \beta_s \end{array}$$

Table 3.1: s-stage Runge-Kutta scheme

The scheme is called explicit if $\rho_1 = 0$ and $\alpha_{jk} = 0, j \leq k$, and implicit otherwise. So, for explicit schemes the matrix (α_{jk}) in Tableau 3.1 is strictly lower triangular. Here, we consider a general implicit case and are interesting only in schemes which satisfy

$$\sum_{k=1}^{s} \beta_k = 1, \qquad \sum_{k=1}^{s} \alpha_{jk} = \rho_j, \qquad 1 \leq j \leq s. \tag{3.317}$$

To interpret (3.313) and (3.314), note that we can write

$$f_{ij} = f(t_{ij}, y_{ij}), \tag{3.318}$$

where

$$y_{ij} = y_i + h \sum_{k=1}^{s} \alpha_{jk} f_{ik} \tag{3.319}$$

is an approximation of $y(t_{ij})$.

3.6.2 Linear Case

Let us now treat the linear case, i.e. consider the BVP

$$\dot{y} = A(t)y + q(t), \qquad a < t < b, \tag{3.320}$$

$$B_a y(a) + B_b y(b) = \beta. \tag{3.321}$$

In the Runge-Kutta formulation (3.313) there are two types of unknowns. One is the vector $y_\pi^\top = (y_0^\top, ..., y_l^\top)$ of mesh values. These variables are global in that their determination is in some sense done simultaneously, over the entire interval $[a, b]$. The over variables are local to $[t_i, t_{i+1}]$, for each $i, 0 \leq i \leq l-1$. These are

$$f_i^\top = (f_{i1}^\top, ..., f_{is}^\top), \qquad 0 \leq i \leq l-1. \tag{3.322}$$

Alternatively, we can use (3.318) - (3.319) to express the scheme in terms of the local unknowns $y_{i1}, ..., y_{is}$. We then obtain

$$h^{-1}(y_{i+1} - y_i) = \sum_{k=1}^{s} \beta_k \left\{A(t_{ik})y_{ik} + q(t_{ik})\right\}, \qquad 0 \leq i \leq l-1, \tag{3.323}$$

$$h^{-1}(y_{ij} - y_i) = \sum_{k=1}^{s} \alpha_{jk} \left\{A(t_{ik})y_{ik} + q(t_{ik})\right\}, \qquad 1 \leq j \leq s. \tag{3.324}$$

For both, practical and theoretical reasons, we now proceed to eliminate the local unknowns. This is sometimes referred to in the literature of finite element methods as parameter condensation. To see how it can be done, we can write (3.314) for a linear BVP in vector form as

$$WF_i = Vy_i + q_i, \tag{3.325}$$

where $W \in \mathbb{R}^{ns \times ns}$, $V \in \mathbb{R}^{ns \times n}$ and $q_i \in \mathbb{R}^{ns}$ are defined by

(3.326)

$$W = I - h \begin{bmatrix} \alpha_{11}A(t_{i1}) & \dots & \alpha_{1s}A(t_{i1}) \\ \vdots & & \\ \alpha_{s1}A(t_{is}) & \dots & \alpha_{ss}A(t_{is}) \end{bmatrix}, V = \begin{bmatrix} A(t_{i1}) \\ \vdots \\ A(t_{is}) \end{bmatrix}, q_i = \begin{bmatrix} q(t_{i1}) \\ \vdots \\ q(t_{is}) \end{bmatrix}.$$

The obvious dependence of the matrices W and V on the index i is suppressed for notational simplicity. Obviously, for h small enough, W is invertible, with

$$W^{-1} = I + \mathcal{O}(h). \tag{3.327}$$

Again, using linearity of the ODE, we find that substitution in (3.313) yields

$$y_{i+1} = G_i y_i + r_i, \qquad 0 \leq i \leq l-1, \tag{3.328}$$

with

$$G_i = I + hDW^{-1}V, \qquad r_i = hDW^{-1}q_i, \tag{3.329}$$

and $D \in \mathbb{R}^{n \times ns}$ being defined by

$$D = (\beta_1 I, ..., \beta_s I)\,. \tag{3.330}$$

Combining (3.328) with the BC (3.321) yields, once again, a linear system of algebraic equations with a multiple shooting type matrix. Indeed, the process here resembles multiple shooting even more than before (cf. Section 3.2), because in multiple shooting, the IVP integrations from one shooting point to the next can also be considered as a local elimination process. Alternatively, the process here can be considered as multiple shooting where IVP integrations are done by one step of a Runge-Kutta scheme. Now, with (3.328) - (3.330) schemes fall within the theory of Section 3.2.

3.6.3 Non-linear Case

For the non-linear BVP

$$\dot{y} = f(t, y(t)), \qquad a < t < b, \tag{3.331}$$

$$g(y(a), y(b)) = 0, \tag{3.332}$$

since $f(t, y)$ is non-linear, the Runge-Kutta scheme (3.313) yields a set of non-linear algebraic equations, for which we consider Newton's method. We perform Newton iterations on the entire set of unknowns

$$y_0, f_0, y_1, f_1, ..., y_{l-1}, f_{l-1}, y_l,$$

and carry out the local parameter elimination of the f_i only after the linearisation. This process is also one of which the quasilinearisation method yields. More formally, see Algorithm 4.1.

If BC (3.332) are decoupled, then, the Step 2 and Step 3 of Algorithm 4.1 can be combined into two sweeps, namely, the forward and the adjoint sweeps. They are to be performed in the forward and backward direction, respectively. The simplest strategy is to implement Algorithm 4.1 by straightforwardly storing all intermediate states of each sweep on a sequential data file and to restore them when they are needed. However, this approach can be realised only when there is a sufficiently large amount of memory available. If this is not the case, then, we can apply checkpointing techniques from Chapter 2.

Algorithm 3.10 Runge-Kutta with Quasilinearisation
Input:
a. A procedure which return values of $f = f(t, y)$, *and* $A = \frac{\partial f}{\partial y}(t, y)$ *for given values of* t *and* y.
b. A procedure which return values of $g(u, v)$, $B_1 = \frac{\partial g}{\partial u}(u, v)$, *and* $B_2 = \frac{\partial g}{\partial v}(u, v)$ *for given values of* u *and* v.
c. An initial solution profile $y_\pi(t)$ *or, equivalently, the values* y_0, f_0, y_1, f_1, ..., y_{l-1}, f_{l-1}, y_l, *where* $f_{ij} = \dot{y}_\pi(t_{ij})$.
d. A mesh π *and a tolerance* TOL.
Output:
Solution which satisfies the equations (3.331) - (3.332) to the extent that the difference between two consecutive Newton iterates is at most TOL.
DO
1. BC equations
Set B_1 *and* B_2 *at* $u = y_0$, $v = y_l$; $\beta := -g(y_0, y_l)$.
2. **FOR** $i = 0, 1, ..., l-1$ **DO**
For each subinterval, assemble equations
FOR $j = 1, 2, ..., s$ **DO**
Get relevant quantities at a point

$$t_{ij} = t_i + h\rho_j,$$

$$y_{ij} = y_i + h\sum_{k=1}^{s} \alpha_{jk} f_{ik},$$

$$A(t_{ij}) = \frac{\partial f}{\partial y}(y_{ij}, t_{ij}),$$

$$q(t_{ij}) = f(t_{ij}, y_{ij}) - f_{ij},$$

END FOR j
Local elimination by (3.322) - (3.327)
$U_i := W_i^{-1} V_i$; $p_i := W_i^{-1} q_i$.
END FOR i
3. Set up and solve the linear system with G_i *and* r_i *given by (3.329)*

$$\begin{pmatrix} -G_1 & I & & & \\ 0 & -G_2 & I & & \\ & & \cdot & \cdot & \\ & & 0 & -G_l & I \\ B_a & & & 0 & B_b \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_l \\ w_{l+1} \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_l \\ \beta \end{pmatrix}.$$

4. Update approximate solution for each subinterval
FOR $i = 0, 1, ..., l-1$ **DO**

$$y_i := y_i + w_i,$$

$$f_i := f_i + U_i w_i + p_i,$$

END FOR i
5. Finish update: $y_l := y_l + w_l$.
While: $\|w_\pi\| \geq TOL$ *(or iteration limit exceeded)*

3.6.4 Runge-Kutta Schemes for Optimal Control Problems

The Runge-Kutta scheme, presented in this section, is initially introduced in [42]. Therefore, the description of this schemes closely follows the description in [42].

Consider the following unconstrained optimal control problem

$$\min_{\mathbf{x},\mathbf{u}} \phi(\mathbf{x}(T)), \tag{3.333}$$

where the system is described by

$$\dot{\mathbf{x}} = f(\mathbf{x}(t), \mathbf{u}(t), t), \quad \mathbf{x}(0) = \mathbf{x}_0. \tag{3.334}$$

The first-order optimality conditions or the Euler-Lagrange equations for the optimal control problem (3.333) - (3.334) are given by

$$\dot{\mathbf{x}} = f(\mathbf{x}(t), \mathbf{u}(t), t), \quad \mathbf{x}(0) = \mathbf{x}_0, \tag{3.335}$$

$$\dot{\bar{\mathbf{x}}} = -H_{\mathbf{x}}^{\top} = -f_{\mathbf{x}}^{\top}\,\bar{\mathbf{x}}, \qquad \bar{\mathbf{x}}(T) = \phi_{\mathbf{x}}^{\top}(\mathbf{x}(T)), \tag{3.336}$$

$$\left(\frac{\partial H}{\partial \mathbf{u}}\right)^{\top} = 0, \qquad 0 \le t \le T, \tag{3.337}$$

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

$$H(\mathbf{x}(t), \mathbf{u}(t), \bar{\mathbf{x}}(t), t) = \bar{\mathbf{x}}^{\top}(t)\, f(\mathbf{x}(t), \mathbf{u}(t), t). \tag{3.338}$$

We consider the discrete approximation to this continuous problem that is obtained by solving the differential equation using a Runge-Kutta integration scheme. For convenience, we consider a uniform mesh of width $h = T/l$, where l is the number of time steps, as before. We let $\mathbf{x}_i$ denote the approximation to $\mathbf{x}(t_i)$ where $t_i = ih$. Then, an s-stage Runge-Kutta scheme with coefficients α_{kj} and β_k, $1 \le k, j \le s$, is given by

$$\dot{\mathbf{x}}_i = \sum_{k=1}^{s} \beta_k f(\mathbf{y}_k, \mathbf{u}_{ik}), \qquad 0 \le i \le l-1, \tag{3.339}$$

where

$$\mathbf{y}_k = \mathbf{x}_i + h \sum_{j=1}^{s} \alpha_{kj} f(\mathbf{y}_j, \mathbf{u}_{ij}), \qquad 1 \le k \le s, \tag{3.340}$$

and dot denotes, in this discrete context, the forward divide difference:

$$\dot{\mathbf{x}}_i = \frac{\mathbf{x}_{i+1} - \mathbf{x}_i}{h}. \tag{3.341}$$

In (3.339) and (3.340), $\mathbf{y}_j$ and $\mathbf{u}_{ij}$ are the intermediate state and control variables on the interval $[t_i, t_{i+1}]$. The dependence of the intermediate state variables on i is not explicit in this notation even though these variables have different values on different intervals.

In this notation, the discrete control problem is the following:

$$\min_{\mathbf{x},\mathbf{u}} \phi(\mathbf{x}_l), \tag{3.342}$$

where the system is described by

$$\begin{aligned} \dot{\mathbf{x}}_i &= \sum_{k=1}^{s} \beta_k f(\mathbf{y}_k, \mathbf{u}_{ik}), \quad \mathbf{x}_0 = \mathbf{x}(0), \qquad 0 \le i \le l-1, \\ \mathbf{y}_k &= \mathbf{x}_i + h \sum_{j=1}^{s} \alpha_{kj} f(\mathbf{y}_j, \mathbf{u}_{ij}), \qquad 1 \le k \le s. \end{aligned} \tag{3.343}$$

For $\mathbf{x}_i$ near an optimal state variable $\mathbf{x}^\star(t_i)$ and $\mathbf{u}_{ij}$, $1 \le j \le s$, near an optimal control variable $\mathbf{u}^\star(t_i)$, in the case when h is small enough, the intermediate variables $\mathbf{y}_k$ in (3.340) are uniquely determined.

Let $f^h : \mathbf{R}^n \times \mathbf{R}^{sm} \to \mathbf{R}^n$ be defined by

$$f^h(\mathbf{x}, \mathbf{u}) = \sum_{k=1}^{s} \beta_k f(\mathbf{y}_k(\mathbf{x}, \mathbf{u}), \mathbf{u}_k). \tag{3.344}$$

In other words,

$$f^h(\mathbf{x}, \mathbf{u}) = \sum_{k=1}^{s} \beta_k f(\mathbf{y}_k, \mathbf{u}_k), \tag{3.345}$$

where $\mathbf{y}$ is the solution of

$$\mathbf{y}_k = \mathbf{x}_i + h \sum_{j=1}^{s} \alpha_{kj} f(\mathbf{y}_j, \mathbf{u}_j), \qquad 1 \le k \le s, \tag{3.346}$$

and $\mathbf{u} = (\mathbf{u}_1^\top, \mathbf{u}_2^\top, ..., \mathbf{u}_s^\top)^\top \in \mathbf{R}^{sm}$. The corresponding discrete Hamiltonian $H^h : \mathbf{R}^n \times \mathbf{R}^n \times \mathbf{R}^{sm} \to \mathbf{R}$ is defined by

$$H^h(\mathbf{x}, \bar{\mathbf{x}}, \mathbf{u}) = \bar{\mathbf{x}}^\top f^h(\mathbf{x}, \mathbf{u}). \tag{3.347}$$

Consider the following version of the first-order necessary optimality conditions or the Euler-Lagrange equations associated with (3.342) - (3.343):

$$\dot{\mathbf{x}}_i = f^h(\mathbf{x}_i, \mathbf{u}_i), \quad \mathbf{x}_0 = \mathbf{x}(0), \tag{3.348}$$

$$\dot{\bar{\mathbf{x}}}_i = -H_{\mathbf{x}}^{h^\top}(\mathbf{x}_i, \bar{\mathbf{x}}_{i+1}, \mathbf{u}_i), \qquad \bar{\mathbf{x}}_l = \phi_{\mathbf{x}}^\top(\mathbf{x}_l), \tag{3.349}$$

$$H_{\mathbf{u}_k}^h(\mathbf{x}_i, \bar{\mathbf{x}}_{i+1}, \mathbf{u}_i) = 0, \qquad 1 \le k \le s, \tag{3.350}$$

where $\bar{\mathbf{x}}_i \in \mathbf{R}^n,\ 0 \le i \le l-1$. Here, $\mathbf{u}_i \in \mathbf{R}^{ms}$ is the entire discrete control vector at time level i:

$$\mathbf{u}_i = (\mathbf{u}_{i1}^\top, \mathbf{u}_{i2}^\top, ..., \mathbf{u}_{is}^\top)^\top \in \mathbf{R}^{ms}. \tag{3.351}$$

Throughout this thesis, the index i refers to the time level in the discrete problem, while $\mathbf{u}_k$ and $\mathbf{u}_j \in \mathbf{R}^m$ denote components of the vector $\mathbf{u} \in \mathbf{R}^{sm}$.

We now rewrite the first-order conditions (3.348) - (3.350) in a way that is better suited for analysis and computations. Suppose that a multiplier λ_k is introduced for the k-th intermediate equation (3.340) in addition to the multiplier $\bar{\mathbf{x}}_{i+1}$ for the equation (3.339). Taking into account these additional multipliers, the first-order necessary conditions are the following:

$$\bar{\mathbf{x}}_i - \bar{\mathbf{x}}_{i+1} = \sum_{k=1}^{s} \lambda_k, \quad \bar{\mathbf{x}}_l = \phi_{\mathbf{x}}^\top(\mathbf{x}_l), \tag{3.352}$$

$$h f_{\mathbf{x}}^\top(\mathbf{y}_j, \mathbf{u}_{ij})(\beta_j \bar{\mathbf{x}}_{i+1} + \sum_{k=1}^{s} \alpha_{kj}\lambda_k) = \lambda_j, \tag{3.353}$$

$$(\beta_j \bar{\mathbf{x}}_{i+1} + \sum_{k=1}^{s} \alpha_{kj}\lambda_k)^\top f_{\mathbf{u}}(\mathbf{y}_j, \mathbf{u}_{ij}) = 0, \tag{3.354}$$

$1 \le j \le s$ and $0 \le i \le l-1$.

In the case that $\beta_j > 0$ for each j, we now reformulate the first-order conditions in terms of the variables χ_j defined by

$$\chi_j = \bar{\mathbf{x}}_{i+1} + \sum_{k=1}^{s} \frac{\alpha_{kj}}{\beta_j}\lambda_k, \qquad 1 \le j \le s. \tag{3.355}$$

With this definition, (3.353) reduces to

$$h\beta_j f_{\mathbf{x}}^\top(\mathbf{y}_j, \mathbf{u}_{ij})\chi_j = \lambda_j. \tag{3.356}$$

Multiplying (3.356) by $\frac{\alpha_{jk}}{\beta_k}$, summing over j, and substituting from (3.355), we have

$$h\sum_{j=1}^{s} \frac{\beta_j \alpha_{jk}}{\beta_k} f_{\mathbf{x}}^\top(\mathbf{y}_j, \mathbf{u}_{ij})\chi_j = \sum_{j=1}^{s} \frac{\alpha_{jk}}{\beta_k}\lambda_j = \chi_k - \bar{\mathbf{x}}_{i+1}. \tag{3.357}$$

Summing (3.356) over j and utilising (3.352) gives

$$h\sum_{j=1}^{s} \beta_j f_{\mathbf{x}}^\top(\mathbf{y}_j, \mathbf{u}_{ij})\chi_j = \sum_{j=1}^{s} \lambda_j = \bar{\mathbf{x}}_i - \bar{\mathbf{x}}_{i+1}. \tag{3.358}$$

Finally, substituting (3.355) in (3.354) yields

$$\beta_j \chi_j^\top f_{\mathbf{u}}(\mathbf{y}_j, \mathbf{u}_{ij}) = 0, \qquad 1 \le j \le s. \tag{3.359}$$

The positive factor β_j can be removed in (3.359) and equations (3.357) - (3.359) yield the transformed first-order system:

$$\bar{\mathbf{x}}_i = \bar{\mathbf{x}}_{i+1} + h\sum_{j=1}^{s} \beta_j f_{\mathbf{x}}^\top(\mathbf{y}_j, \mathbf{u}_{ij})\chi_j, \quad \bar{\mathbf{x}}_l = \phi_{\mathbf{x}}^\top(\mathbf{x}_l), \tag{3.360}$$

$$\chi_k = \bar{\mathbf{x}}_{i+1} + h\sum_{j=1}^{s} \frac{\beta_j \alpha_{jk}}{\beta_k} f_{\mathbf{x}}^\top(\mathbf{y}_j, \mathbf{u}_{ij})\chi_j, \tag{3.361}$$

$$\chi_j^\top f_{\mathbf{u}}(\mathbf{y}_j, \mathbf{u}_{ij}) = 0, \tag{3.362}$$

$1 \le k \le s$ and $0 \le i \le l-1$.

Observe that conditions (3.360) and (3.361) are in essence a Runge-Kutta scheme applied to the continuous adjoint equation (3.336). Although the adjoint Runge-Kutta scheme is generally not the same as the scheme (3.339) and (3.340) for the state equation, it is observed in [41] that some common Runge-Kutta schemes are symmetric in the sense that the state and adjoint schemes are the same.

Proposition 3.1 *If $\beta_j > 0$ for each j, then, the first-order system (3.352) - (3.354) and the transformed first-order system (3.360) - (3.362) are equivalent. That is, if $\lambda_1, ..., \lambda_s$ satisfy (3.352) - (3.354), then, (3.360) - (3.362) hold for χ_j defined in (3.355). Conversely, if $\chi_1, ..., \chi_s$ satisfy (3.360) - (3.362), then, (3.352) - (3.354) hold for λ_j defined in (3.356).*

Proof: See [42].

Let $\mathbf{u} \in \mathbf{R}^{sml}$ denote the vector of intermediate control values for the entire interval $[0, T]$, and let $\phi(\mathbf{u})$ denote the value $\phi(\mathbf{x}_l)$ for the discrete cost function associated with these controls. The transformed first-order system (3.360) - (3.362) provides a convenient way to compute the gradient of the discrete cost function $\phi(\mathbf{x}_l)$. In particular, as seen in [43],

$$\phi_{\mathbf{u}_{ij}}(\mathbf{u}) = h\beta_j \chi_j^\top f_{\mathbf{u}}(\mathbf{y}_j, \mathbf{u}_{ij}), \tag{3.363}$$

where the intermediate values for the discrete state and costate variables are gotten by first solving the discrete state equations (3.339) and (3.340), for $i = 0, 1, ..., l-1$, using the given values for the controls, and then, using these computed values for both the state and intermediate variables in (3.360) and (3.361) when computing the values of the discrete costate for $i = l-1, l-2, ..., 0$. Thus, the discrete state equation is solved by marching forward from $i = 0$ while the discrete costate equation is solved by marching backward from $i = l-1$.

We now observe that the multiplier $\bar{\mathbf{x}}_i$ gotten by solving (3.361) for χ and substituting in (3.360) is identical to the multiplier gotten from (3.349). Moreover, the condition (3.362) involving χ_j satisfying (3.361) is equivalent to the condition (3.350).

Proposition 3.2 *Suppose that*

$$(\mathbf{x}_i, \mathbf{u}_{ij}) \in B_\beta(\mathbf{x}^\star(t_i, \mathbf{u}^\star(t_i))), \qquad 1 \le j \le s, \tag{3.364}$$

and for $\mathbf{y} = \mathbf{y}(\mathbf{x}_i, \mathbf{u}_i)$, let M be the $s \times s$ block matrix whose (i, j) block is the $n \times n$ matrix $\alpha_{ij} f_{\mathbf{x}}(\mathbf{y}_j, \mathbf{u}_{ij})$. If h is small enough, so that $I - hM$ is invertible, then, there exists a solution $\chi_1, ..., \chi_s$ to (3.361), and we have

$$H_{\mathbf{x}}^h(\mathbf{x}_i, \chi_{i+1}, \mathbf{u}_i) = \sum_{k=1}^{s} \beta_k \chi_k^\top f_{\mathbf{x}}(\mathbf{y}_k, \mathbf{u}_{ik}) = \sum_{k=1}^{s} \beta_k H_{\mathbf{x}}(\mathbf{y}_k, \chi_k, \mathbf{u}_{ik}), \tag{3.365}$$

and

$$H_{\mathbf{u}_j}^h(\mathbf{x}_i, \chi_{i+1}, \mathbf{u}_i) = \beta_j \chi_j^\top f_{\mathbf{u}}(\mathbf{y}_j, \mathbf{u}_{ij}) = \beta_j H_{\mathbf{u}}(\mathbf{y}_j, \chi_j, \mathbf{u}_{ij}). \tag{3.366}$$

Proof: See [42].

Since the boundary conditions for $\bar{\mathbf{x}}_l$ are the same in both (3.349) and (3.352), it follows from (3.365) that when h is sufficiently small and $(\mathbf{x}_i, \mathbf{u}_{ij}) \in B_\beta(\mathbf{x}^\star(t_i, \mathbf{u}^\star(t_i)))$ for each j and i, then, the $\bar{\mathbf{x}}_i$ given by (3.349) and by (3.360)

- (3.361) are the same. Moreover, the control gradient satisfies (3.350) if and only if the following relation holds in the transformed variables:

$$H_{\mathbf{u}}(\mathbf{y}_k, \chi_k, \mathbf{u}_{ik}) = 0, \qquad 1 \leq k \leq s. \tag{3.367}$$

The transformed discrete costate equations (3.360) - (3.361) march backwards in time while the discrete state equations (3.339) - (3.340) march forward in time. To facilitate the following analysis, reverse the order of time in the costate equation. That is, we solve for $\bar{\mathbf{x}}_{i+1}$ in (3.360) and substitute in (3.361) to obtain the following forward marching scheme:

$$\bar{\mathbf{x}}_{i+1} = \bar{\mathbf{x}}_i - h \sum_{k=1}^{s} \beta_k f_{\mathbf{x}}^{\top}(\mathbf{y}_k, \mathbf{u}_{ik})\chi_k, \tag{3.368}$$

$$\chi_k = \bar{\mathbf{x}}_i - h \sum_{j=1}^{s} \bar{\alpha}_{kj} f_{\mathbf{x}}^{\top}(\mathbf{y}_j, \mathbf{u}_{ij})\chi_j, \qquad \bar{\alpha}_{kj} = \frac{\beta_k\beta_j - \beta_j\alpha_{jk}}{\beta_k}. \tag{3.369}$$

Remove the control from the state equation and the transformed adjoint equation by use of the minimum principle. Focusing on the situation where the control is uniquely determined by $(\mathbf{x}, \bar{\mathbf{x}})$ through minimisation of the Hamiltonian, let σ denote the function defined by

$$\sigma(\mathbf{x}, \bar{\mathbf{x}}) = -H_{\mathbf{x}}^{\top}(\mathbf{x}, \mathbf{u}, \bar{\mathbf{x}}) \mid_{\mathbf{u}\equiv\mathbf{u}(\mathbf{x},\bar{\mathbf{x}})} . \tag{3.370}$$

Let $f(\mathbf{x}, \bar{\mathbf{x}})$ denote the function $f(\mathbf{x}, \mathbf{u}(\mathbf{x}, \bar{\mathbf{x}}))$. In the case where the control has the special form $\mathbf{u}_{ij} = \mathbf{u}(\mathbf{y}_j, \chi_j)$, the Runge-Kutta discretization (3.339) - (3.340), coupled with the transformed, time reversed costate equation (3.368) - (3.369), can be expressed:

$$\mathbf{x}_{i+1} = \mathbf{x}_i + h \sum_{k=1}^{s} \beta_k f(\mathbf{y}_k, \chi_k), \quad \mathbf{x}(0) = \mathbf{x}_0, \tag{3.371}$$

$$\bar{\mathbf{x}}_{i+1} = \bar{\mathbf{x}}_i + h \sum_{k=1}^{s} \beta_k \sigma(\mathbf{y}_k, \chi_k), \quad \bar{\mathbf{x}}_l = \phi_{\mathbf{x}}^{\top}(\mathbf{x}_l) \tag{3.372}$$

$$\mathbf{y}_k = \mathbf{x}_i + h \sum_{j=1}^{s} \alpha_{kj} f(\mathbf{y}_j, \chi_j), \tag{3.373}$$

(3.374) $$\chi_k = \bar{\mathbf{x}}_i + h \sum_{j=1}^{s} \bar{\alpha}_{kj} \sigma(\mathbf{y}_j, \chi_j), \qquad \bar{\alpha}_{kj} = \frac{\beta_k \beta_j - \beta_j \alpha_{jk}}{\beta_k}.$$

The scheme (3.371) - (3.374) can be viewed as a discretisation of the following two-point boundary-value problem:

(3.375) $$\dot{\mathbf{x}} = f(\mathbf{x}(t), \bar{\mathbf{x}}(t)), \quad \mathbf{x}(0) = \mathbf{x}_0,$$

(3.376) $$\dot{\bar{\mathbf{x}}} = \sigma(\mathbf{x}(t), \bar{\mathbf{x}}(t)), \quad \bar{\mathbf{x}}(T) = \phi_{\mathbf{x}}(\mathbf{x}(T)).$$

This two-point boundary-value problem is gotten by substituting in (3.335) - (3.336) the control obtained by solving (3.337) for $\mathbf{u}(t)$ in terms of $(\mathbf{x}(t), \bar{\mathbf{x}}(t))$.

3.7 Hamiltonian Systems

In this section Hamiltonian systems and their properties are introduced (see e.g. [82, 10]). Using this, one can characterise systems of differential equations handled before and introduce other stable numerical methods to integrate them.

Definition 3.1 *Hamiltonian System*

A system of differential equations is called a **Hamiltonian System** *if there exists a real-valued function $H(p, q, t)$, so that*

(3.377) $$\begin{aligned} \tfrac{dp}{dt} &= \frac{\partial H^\top}{\partial q} = \mathcal{F}(p, q, t), \\ \tfrac{dq}{dt} &= -\frac{\partial H^\top}{\partial p} = \mathcal{G}(p, q, t), \end{aligned}$$

for all p and q. The function H is called the **Hamiltonian function** *for the system.*

Example 3.6 *An example of a Hamiltonian system is the coupled system of state and adjoint equations (3.335) - (3.336), where the Hamiltonian function H is given by (3.338). Here, $p = \mathbf{x}$ and $q = \bar{\mathbf{x}}$ in the notations of Definition 3.1. Particularly, consider the following equations*

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

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

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

$$H(\mathbf{x}(t), \mathbf{u}(t), \bar{\mathbf{x}}(t), t) = \bar{\mathbf{x}}^\top(t)\, f(\mathbf{x}(t), \mathbf{u}(t), t). \tag{3.380}$$

The control function $\mathbf{u}(t)$ *can be eliminated from the Hamiltonian function* H *and its partial derivatives, if we apply the Implicit Function Theorem for the first order optimality condition*

$$\left(\frac{\partial H}{\partial \mathbf{u}}(\mathbf{x}(t), \mathbf{u}(t), \bar{\mathbf{x}}(t), t)\right)^\top = 0, \qquad 0 \le t \le T. \tag{3.381}$$

Since near an optimal point the second order optimality condition, namely, $H_{\mathbf{uu}}(t) > 0$ *is satisfied, due to the Implicit Function Theorem, there is a function* $\mathbf{u} = \mathbf{u}(\mathbf{x}, \bar{\mathbf{x}}, t)$*, so that*

$$\left(\frac{\partial H}{\partial \mathbf{u}}(\mathbf{x}(t), \mathbf{u}(\mathbf{x}, \bar{\mathbf{x}}, t), \bar{\mathbf{x}}(t), t)\right)^\top = 0, \qquad 0 \le t \le T. \tag{3.382}$$

Moreover, the partial derivative $\frac{\partial \mathbf{u}}{\partial \bar{\mathbf{x}}}(\mathbf{x}, \bar{\mathbf{x}}, t)$ *exists and has the following form*

$$\frac{\partial \mathbf{u}}{\partial \bar{\mathbf{x}}}(\mathbf{x}, \bar{\mathbf{x}}, t) = -H_{\mathbf{uu}}^{-1}(\mathbf{x}, \bar{\mathbf{x}}, t) H_{\mathbf{u}\bar{\mathbf{x}}}(\mathbf{x}, \bar{\mathbf{x}}, t). \tag{3.383}$$

Thus, to prove that the system (3.378) - (3.379) is a Hamiltonian system, we have to show that $\dot{\mathbf{x}} = H_{\bar{\mathbf{x}}}^\top$*. Using (3.378) and (3.380), we obtain*

$$\begin{aligned}
\left(\frac{\partial H}{\partial \bar{\mathbf{x}}}(\mathbf{x}(t), \mathbf{u}(\mathbf{x}, \bar{\mathbf{x}}, t), \bar{\mathbf{x}}(t), t)\right)^\top &= \left(\frac{\partial H}{\partial \bar{\mathbf{x}}}(\cdot) + \frac{\partial H}{\partial \mathbf{u}}(\cdot)\, \frac{\partial \mathbf{u}}{\partial \bar{\mathbf{x}}}(\mathbf{x}, \bar{\mathbf{x}}, t)\right)^\top = \\
&= \left(\frac{\partial H}{\partial \bar{\mathbf{x}}}(\cdot) - \underbrace{\frac{\partial H}{\partial \mathbf{u}}(\cdot)}_{=0}\, H_{\mathbf{uu}}^{-1}(\mathbf{x}, \bar{\mathbf{x}}, t) H_{\mathbf{u}\bar{\mathbf{x}}}(\mathbf{x}, \bar{\mathbf{x}}, t)\right)^\top = \\
&= \left(\frac{\partial\left(\sum_{i=1}^{n} \bar{\mathbf{x}}_i f_i(\cdot)\right)}{\partial \bar{\mathbf{x}}_1}, \ldots, \frac{\partial\left(\sum_{i=1}^{n} \bar{\mathbf{x}}_i f_i(\cdot)\right)}{\partial \bar{\mathbf{x}}_n}\right)^\top = \\
&= (f_1(\cdot), \ldots, f_n(\cdot))^\top = \begin{pmatrix} f_1(\cdot) \\ \cdots \\ f_n(\cdot) \end{pmatrix} = \dot{\mathbf{x}}.
\end{aligned} \tag{3.384}$$

where $(\cdot) = (\mathbf{x}(t), \mathbf{u}(t), \bar{\mathbf{x}}(t), t)$*. Therefore, the system (3.378) - (3.379) is a Hamiltonian system.*

□

In many Hamiltonian systems of interest, the Hamiltonian H does not explicitly depend on t, then, (3.377) is an autonomous system of differential equations.

It is sometimes useful to combine all the dependent variables in (3.377) in a $2n$-dimensional vector $y^\top = (p^\top, q^\top)$. Then (3.377) takes the simple form

$$\frac{dy}{dt} = J\nabla H, \tag{3.385}$$

where ∇ is the gradient operator

$$(\partial/\partial p_1, ..., \partial/\partial p_n, \partial/\partial q_1, ..., \partial/\partial q_n)^\top, \tag{3.386}$$

and J is the $2n \times 2n$ skew-symmetric matrix

$$J = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix}, \tag{3.387}$$

where I and 0 respectively represent the unit and zero $n \times n$ matrices.

Upon differentiation of H with respect to t along a solution of (3.377), we find

$$\frac{d}{dt}H(y(t),t) = (\nabla H)^\top \frac{dy}{dt} + \frac{\partial H}{\partial t}, \tag{3.388}$$

so that in view of (3.385) and of the skew-symmetry of J,

$$\frac{d}{dt}H(y(t),t) = (\nabla H)^\top J \nabla H + \frac{\partial H}{\partial t} = \frac{\partial H}{\partial t}. \tag{3.389}$$

In particular, if H is autonomous, $\frac{d}{dt}H(y(t),t) = 0$. Then, H is a conserved quantity that remains constant along solutions of the system. In the applications, this usually corresponds to conservation of energy. More formally, use the following definition.

Definition 3.2 *Conserved Quantity*

A real-valued function $H(p,q)$ of the two variables p and q is a **conserved quantity** *for a system of differential equations if H is constant along all solution curves of the system. That is, if $(p(t), q(t))$ is a solution of the system, then, $H(p(t), q(t))$ is constant. In other words,*

$$\frac{d}{dt}H(p(t), q(t)) = 0. \tag{3.390}$$

Note, H is always a conserved quantity for autonomous Hamiltonian systems, defined by Definition 3.1. Then,

$$\begin{aligned} \frac{d}{dt}H(p(t), q(t)) &= \frac{\partial H}{\partial p}\frac{dp}{dt} + \frac{\partial H}{\partial q}\frac{dq}{dt} = \\ &= \frac{\partial H}{\partial p}\left(\frac{\partial H^\top}{\partial q}\right) + \frac{\partial H}{\partial q}\left(-\frac{\partial H^\top}{\partial p}\right) = 0. \end{aligned} \tag{3.391}$$

The first equality is the Chain Rule, and the second equality uses the fact that the system is Hamiltonian and that $(p(t), q(t))$ is a solution to replace $\frac{dp}{dt}$ with $\frac{\partial H^\top}{\partial q}$ and $\frac{dq}{dt}$ with $-\frac{\partial H^\top}{\partial p}$.

3.7.1 Symplecticness

Symplectiness is a characterisation of Hamiltonian systems in terms of their solutions, rather than in terms of the actual form of the differential equations. The description of this property closely follows the description in [82]. We first need the concept of **solution operator**.

If t, t_0 are real numbers in the interval I, then, we denote by $\Phi_H(t, t_0)$ the solution operator of the system (3.377). By definition, $\Phi_H(t, t_0)$ is a transformation mapping the phase space Ω into itself, so that for (p^0, q^0) in Ω,

$$(p, q) = \Phi_H(t, t_0)(p^0, q^0), \tag{3.392}$$

is the value at time t of the solution of (3.377) that at time t_0 has the initial condition (p^0, q^0).

The idea of symplectic integration revolves around the use of symplectic transformations. In the one-degree-of-freedom case, the symplecticness is nothing but preservation of area.

For each real t, t_0 the solution mapping $\Phi_H(t, t_0)$ is an area-preserving transformation in Ω, in the sense that, for each bounded subdomain $D \subset \Omega$ for which $\Phi_H(t, t_0)(D)$ is defined, it holds true that D and $\Phi_H(t, t_0)(D)$ have the same area (and orientation). To see this, it is enough to observe that, for each t, the vector field in phase space $[\frac{\partial H}{\partial q}, -\frac{\partial H}{\partial p}]^\top$ that features in (3.377) is divergence-free because

$$\frac{\partial}{\partial p}\left(-\frac{\partial H}{\partial q}\right) + \frac{\partial}{\partial q}\left(\frac{\partial H}{\partial p}\right) = 0. \tag{3.393}$$

The preservation of area can be checked using various methods. One of them uses differential form. Particularly, let $(p^\star, q^\star) = \psi(p, q)$ be a transformation defined in a domain Ω. According to the standard rule for changing variables in an integral, ψ preserves area and orientation if and only if the Jacobian determinant is identically 1:

$$\forall (p, q) \in \Omega, \qquad \frac{\partial p^\star}{\partial p}\frac{\partial q^\star}{\partial q} - \frac{\partial p^\star}{\partial q}\frac{\partial q^\star}{\partial p} = 1. \tag{3.394}$$

We see a differential 1-form in Ω as a formal combination $P(p,q)dp+Q(p,q)dq$ where P and Q are smooth real-valued functions defined in Ω. The differentials $dp^\star$ and $dq^\star$ of the components of the transformation ψ are differential 1-forms

$$dp^\star = \frac{\partial p^\star}{\partial p}dp + \frac{\partial p^\star}{\partial q}dq, \qquad dq^\star = \frac{\partial q^\star}{\partial p}dp + \frac{\partial q^\star}{\partial q}dq. \tag{3.395}$$

Two differential 1-forms w and w' give rise, via the exterior wedge product $\wedge$, to a new entity $w \wedge w'$ called a differential 2-form. The exterior product is bilinear, so that

$$\begin{aligned} dp^\star \wedge dq^\star &= \tfrac{\partial p^\star}{\partial p}\tfrac{\partial q^\star}{\partial p}dp \wedge dp + \tfrac{\partial p^\star}{\partial p}\tfrac{\partial q^\star}{\partial q}dp \wedge dq + \\ &+ \tfrac{\partial p^\star}{\partial q}\tfrac{\partial q^\star}{\partial p}dq \wedge dp + \tfrac{\partial p^\star}{\partial q}\tfrac{\partial q^\star}{\partial q}dq \wedge dq. \end{aligned} \tag{3.396}$$

The exterior product is skew-symmetric. In particular, it holds that

$$dp \wedge dp = dq \wedge dq = 0, \qquad dp \wedge dq = -dq \wedge dp. \tag{3.397}$$

Thus,

$$dp^\star \wedge dq^\star = \left(\frac{\partial p^\star}{\partial p}\frac{\partial q^\star}{\partial q} - \frac{\partial p^\star}{\partial q}\frac{\partial q^\star}{\partial p} \right) dp \wedge dq, \tag{3.398}$$

and from (3.394) we see that the conservation of area is equivalent to

$$dp^\star \wedge dq^\star = dp \wedge dq. \tag{3.399}$$

This usually provides a convenient way of checking preservation of area. Generalising the considerations above to the n dimensional space with $n \geq 1$ we obtain the following theorem.

Theorem 3.4 *For each t, t_0 the solution operator $\Phi_H(t,t_0)$ of a Hamiltonian system (3.377) is a symplectic transformation if the following is satisfied*

$$dp_1^\star \wedge dq_1^\star + ... + dp_n^\star \wedge dq_n^\star = dp_1 \wedge dq_1 + ... + dp_n \wedge dq_n. \tag{3.400}$$

This relation can be rewritten more compactly as

$$dp^\star \wedge dq^\star = dp \wedge dq. \tag{3.401}$$

Proof: See [82].

3.7.2 Symplectic Numerical Methods

In the previous section we saw that the distinctive feature of the solution operator Φ of Hamiltonian problems is symplecticness (or in the one-degree-of-freedom case conservation of oriented area). Moreover, there are different numerical methods which replace Φ by an approximation Ψ. If we wish the approximation Ψ to retain the 'Hamiltonian' features of Φ, we should insist on Ψ also being a symplectic transformation. However, most standard numerical integrators replace Φ by a non symplectic mapping Ψ. These are for instance, the explicit and implicit Euler's methods. A simple example of a symplectic integrator is given by the midpoint rule.

3.7.3 Symplectic Runge-Kutta Methods

The application of the s-stage Runge-Kutta method to the integration of the Hamiltonian system (3.377) results in the relations (see [82])

$$P_k = p_i + h \sum_{j=1}^{s} \alpha_{kj} \mathcal{F}(P_j, Q_j), \tag{3.402}$$

$$Q_k = q_i + h \sum_{j=1}^{s} \alpha_{kj} \mathcal{G}(P_j, Q_j), \tag{3.403}$$

$$p_{i+1} = p_i + h \sum_{k=1}^{s} \beta_k \mathcal{F}(P_k, Q_k), \tag{3.404}$$

$$q_{i+1} = q_i + h \sum_{k=1}^{s} \beta_k \mathcal{G}(P_k, Q_k), \tag{3.405}$$

where $\mathcal{F}$ and $\mathcal{G}$ respectively denote the n-vectors with components $\frac{\partial H}{\partial q_k}$ and $-\frac{\partial H}{\partial p_k}$ and P_k and Q_k are the internal stages corresponding to the p and q variables. Then, the following result is valid.

Theorem 3.5 *Assume that the coefficients of the method (3.402) - (3.405) satisfy the relations*

$$\beta_k \alpha_{kj} + \beta_j \alpha_{jk} - \beta_k \beta_j = 0, \qquad k, j = 1, ..., s. \tag{3.406}$$

Then, the method is symplectic.

Proof: See [82].

3.7.4 Symplectic Runge-Kutta Schemes for Optimal Control Problems

Recall the scheme, presented in Section 3.6.4, namely the resulting Runge-Kutta scheme

$$\begin{aligned}
(3.407)\qquad \mathbf{y}_k &= \mathbf{x}_i + h\sum_{j=1}^{s}\alpha_{kj} f(\mathbf{y}_j,\chi_j),\\
(3.408)\qquad \chi_k &= \bar{\mathbf{x}}_i + h\sum_{j=1}^{s}\bar{\alpha}_{kj}\sigma(\mathbf{y}_j,\chi_j), \qquad \bar{\alpha}_{kj} = \frac{\beta_k\beta_j-\beta_j\alpha_{jk}}{\beta_k},\\
(3.409)\qquad \mathbf{x}_{i+1} &= \mathbf{x}_i + h\sum_{k=1}^{s}\beta_k f(\mathbf{y}_k,\chi_k),\\
(3.410)\qquad \bar{\mathbf{x}}_{i+1} &= \bar{\mathbf{x}}_i + h\sum_{k=1}^{s}\beta_k \sigma(\mathbf{y}_k,\chi_k),
\end{aligned}$$

which can be viewed as a discretisation of the following two-point boundary-value problem:

$$\begin{aligned}
(3.411)\qquad \dot{\mathbf{x}} &= f(\mathbf{x}(t),\bar{\mathbf{x}}(t)), \qquad \mathbf{x}(0)=\mathbf{x}_0,\\
(3.412)\qquad \dot{\bar{\mathbf{x}}} &= \sigma(\mathbf{x}(t),\bar{\mathbf{x}}(t)), \qquad \bar{\mathbf{x}}(T)=\phi_{\mathbf{x}}(\mathbf{x}(T)).
\end{aligned}$$

The boundary-value problem (3.411) - (3.412) is a Hamiltonian system due to its construction in Section 3.6.4 and Example 3.6. The next theorem proves that the "method (3.407) - (3.410) is a symplectic method for the Hamiltonian system (3.411) - (3.412)" ([14]).

Theorem 3.6 *The Runge-Kutta method (3.407) - (3.410) is a symplectic one.*

Proof: We follow the techniques used by Sans-Sena and Calvo [82] in the proof of Theorem 3.5. We employ the notation

$$(3.413)\qquad f_k = f(\mathbf{y}_k,\chi_k), \qquad \sigma_k = \sigma(\mathbf{y}_k,\chi_k)$$

for the slopes at the stages. Differentiate (3.409) - (3.410) and form the exterior product to arrive at

$$
\begin{aligned}
d\mathbf{x}_{i+1} \wedge d\bar{\mathbf{x}}_{i+1} &= d\mathbf{x}_i \wedge d\bar{\mathbf{x}}_i + h\sum_{k=1}^{s} \beta_k df_k \wedge d\bar{\mathbf{x}}_i + \\
&+ h\sum_{j=1}^{s} \beta_j d\mathbf{x}_i \wedge d\sigma_j + \\
&+ h^2 \sum_{k,j=1}^{s} \beta_k\beta_j df_k \wedge d\sigma_j.
\end{aligned} \tag{3.414}
$$

The next step is to eliminate $df_k \wedge d\bar{\mathbf{x}}_i$ and $d\mathbf{x}_i \wedge d\sigma_j$ from this expression. This is easily achieved by differentiating (3.407) and (3.408) and taking the exterior product of the result with df_k, $d\sigma_j$, respectively. The outcome of the elimination is

$$
\begin{aligned}
d\mathbf{x}_{i+1} \wedge\ & d\bar{\mathbf{x}}_{i+1} - d\mathbf{x}_i \wedge d\bar{\mathbf{x}}_i = \\
&= h\sum_{k=1}^{s} \beta_k \left[df_k \wedge d\chi_k + d\mathbf{y}_k \wedge d\sigma_k\right] - \\
&- h^2 \sum_{k,j=1}^{s} (\beta_k\bar{\alpha}_{kj} + \beta_j\alpha_{jk} - \beta_k\beta_j) df_k \wedge d\sigma_j.
\end{aligned} \tag{3.415}
$$

The second term in the right-hand side vanishes in view of (3.408), i.e. because $\bar{\alpha}_{kj} = \frac{\beta_k\beta_j - \beta_j a_{jk}}{\beta_k}$. To finish the proof it is then sufficient to show that, for each k,

$$
df_k \wedge d\chi_k + d\mathbf{y}_k \wedge d\sigma_k = 0. \tag{3.416}
$$

Dropping the subscript k that numbers the stages, one can write

$$
\begin{aligned}
df \wedge d\chi + d\mathbf{y} \wedge d\sigma &= \sum_{\mu=1}^{n} \left[df_\mu \wedge d\chi_\mu + d\mathbf{y}_\mu \wedge d\sigma_\mu\right] = \\
&= \sum_{\mu,\nu=1}^{n} \left[\frac{\partial f_\mu}{\partial \mathbf{x}_\nu} d\mathbf{y}_\nu \wedge d\chi_\mu + \frac{\partial f_\mu}{\partial \bar{\mathbf{x}}_\nu} d\chi_\nu \wedge d\chi_\mu + \right. \\
&+ \left. \tfrac{\partial \sigma_\mu}{\partial \mathbf{x}_\nu} d\mathbf{y}_\mu \wedge d\mathbf{y}_\nu + \tfrac{\partial \sigma_\mu}{\partial \bar{\mathbf{x}}_\nu} d\mathbf{y}_\mu \wedge d\chi_\nu \right].
\end{aligned} \tag{3.417}
$$

To see that this expression vanishes, write the derivatives of f_μ and σ_μ as derivatives of H

$$
\begin{aligned}
\tfrac{\partial f_\mu}{\partial \mathbf{x}_\nu} &= \tfrac{\partial^2 H}{\partial \mathbf{x}_\nu \partial \bar{\mathbf{x}}_\mu}, \\
\tfrac{\partial f_\mu}{\partial \bar{\mathbf{x}}_\nu} &= \tfrac{\partial^2 H}{\partial \bar{\mathbf{x}}_\nu \partial \bar{\mathbf{x}}_\mu}, \\
\tfrac{\partial \sigma_\mu}{\partial \mathbf{x}_\nu} &= -\tfrac{\partial^2 H}{\partial \mathbf{x}_\nu \partial \mathbf{x}_\mu}, \\
\tfrac{\partial \sigma_\mu}{\partial \bar{\mathbf{x}}_\nu} &= -\tfrac{\partial^2 H}{\partial \bar{\mathbf{x}}_\nu \partial \mathbf{x}_\mu}.
\end{aligned} \tag{3.418}
$$

This and the skew-symmetry of the wedge product imply

$$
\begin{aligned}
\sum_{\mu,\nu=1}^{n}\left[\frac{\partial f_\mu}{\partial \mathbf{x}_\nu} d\mathbf{y}_\nu \wedge d\chi_\mu + \frac{\partial \sigma_\mu}{\partial \bar{\mathbf{x}}_\nu} d\mathbf{y}_\mu \wedge d\chi_\nu\right] &= 0, \\
\sum_{\mu,\nu=1}^{n} \frac{\partial f_\mu}{\partial \bar{\mathbf{x}}_\nu} d\chi_\nu \wedge d\chi_\mu &= 0, \\
\sum_{\mu,\nu=1}^{n} \frac{\partial \sigma_\mu}{\partial \mathbf{x}_\nu} d\mathbf{y}_\mu \wedge d\mathbf{y}_\nu &= 0.
\end{aligned} \tag{3.419}
$$

This concludes the proof.

□

3.8 Conclusions

The iterative solution of BVPs by various methods leads to a succession of double sweep through the discretised time interval. The second (adjoint) sweep relies on information from the first (original) sweep. In order to avoid storing full traces of the original sweep, reversal schedules can be applied, which require only the storage of selected original intermediate states. Consequently, memory requirement needed for the implementation of these algorithms reduces drastically. Some of these methods can be applied to the iterative solution of optimal control problems in ODEs.

Another type of methods for iterative solution of optimal control problems in ODEs leads to a succession of triple sweeps through the discretized time interval. The second (adjoint) sweep relies on information from the first (original) sweep, and the third (final) sweep depends on both of them. This flow of information is depicted in Fig. 3.14. 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 original and adjoint sweeps, we consider nested reversal schedules that require only the storage of selected original and adjoint intermediate states called thin and fat checkpoints (see Chapter 4).

One of multiple shooting methods, namely, Riccati approach for iterative solution of optimal control problems in ODEs was considered in a more detailed way. Riccati method was encountered as a continuous counterpart for the so-called Pantoja approach, which describes a stagewise construction of the Newton direction for discrete optimal control problems. It was established a correspondence between discrete Riccati approach and Pantoja method.

Moreover, Pantoja method is extended to the case where the state equations are discretised using one of implicit numerical methods.

Because a BVP obtained from the first order conditions for optimal control problems in ODEs builds a Hamiltonian system, the solution of which is a symplectic operator, it is required to use symplectic numerical methods for the solution of these optimal control problems. One appropriate method, based on an s-stage Runge-Kutta method was introduced. Moreover, the symplecticness of this method was proved. For the memory efficient implementation of this method nested reversal schedules can be used.

Chapter 4

Nested Reversal Schedules

In this chapter previous techniques of checkpointing are extended to a so-called "nested checkpointing" for multiple transversals. We introduce a formal concept of nested checkpointing. Nested checkpointing can be applied to obtain a space efficient implementation of Pantoja/Riccati/Newton method. Moreover, nested reversal schedules can be used for an implementation of adaptive multi-grid methods. It will be shown that the dependence on number of time steps can be arranged polylogarithmically by nested checkpoint strategies. Consequently, the operations count also grows as a second power of $\ln l$, with l being a number of time steps.

4.1 Basic Assumptions and Definitions

Consider a multiple sweep evolution $\mathcal{E}_{N\times l}$ which contains N sweeps ($N \geq 2$). Each sweep consists of l consecutive time steps. An example of such an evolution is shown in Figure 4.1. k and j are sweep counters with the following properties $k = 0, \ldots, N-1$ and $j = 0, \ldots, \lfloor (N-1)/2 \rfloor$, respectively. A subset of an evolution $\mathcal{E}_{N\times l}$ for a specified j is shown in Figure 4.1 more concisely. This subset contains three sweeps. Basic properties and elements of a multiple sweep evolution $\mathcal{E}_{N\times l}$ can be identified by them.

Time steps are shown as horizontal arrows. Their directions denote the information flow. Nodes denote different intermediate states. Each sweep is characterised by a specified direction, i.e. the direction of horizontal arrows within a single sweep. This shows the corresponding information flow between neighbouring time states of a single sweep. Note that the information flow within a single sweep has a constant direction. An additional

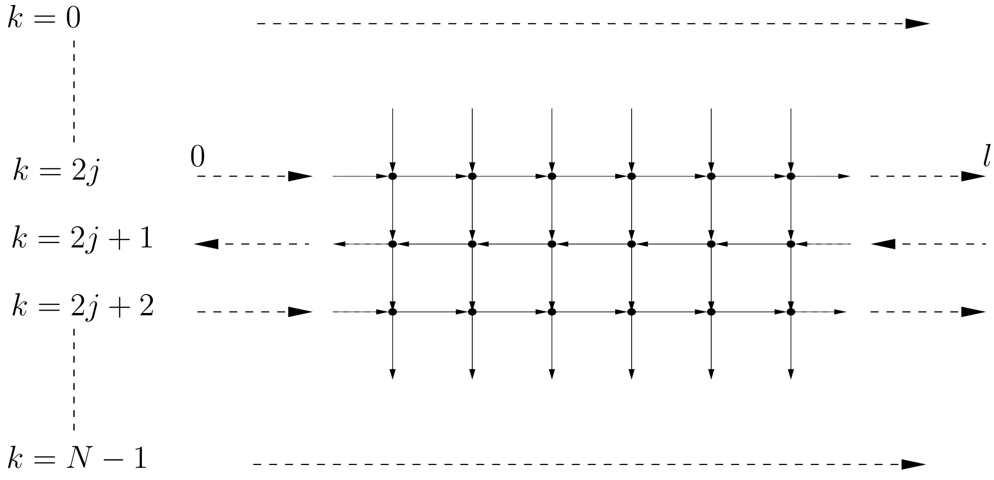

Figure 4.1: Multiple sweep evolution $\mathcal{E}_{N \times l}$

information flow exists between nodes being intermediate states of different successive sweeps. This is shown by vertical lines in Figure 4.1.

4.1.1 Characterisation of Time Steps and Intermediate States

Firstly, we denote the ith intermediate state of the kth sweep x_i^k, $i = 0, \ldots, l$; $k = 0, \ldots, N-1$. In the same manner we identify time steps of various sweeps with F_i^k, $i = 1, \ldots, l$; $k = 0, \ldots, N-1$. In accordance with the information flow in a specified sweep, the corresponding intermediate state is evaluated by

$$x_i^k = F_i^k(x_{i-1}^k), \qquad \text{if } k = 0, \tag{4.1}$$

$$x_i^k = F_i^k(x_{i-1}^k, x_i^{k-1}), \qquad \text{if } k > 0 \text{ and } k \text{ is even}, \tag{4.2}$$

and

$$x_{i-1}^k = F_i^k(x_i^k, x_{i-1}^{k-1}), \qquad \text{if } k \text{ is odd}. \tag{4.3}$$

We assume that $x_i^{k-1} \subset x_i^{k+1}$, $1 \leq k \leq N-2$ and k is odd. Furthermore, we introduce evaluation costs, i.e. the computational effort for time steps of different sweeps by

$$t_i^k \equiv TIME(F_i^k), \quad \text{for} \quad 1 \leq i \leq l,\ 0 \leq k \leq N-1. \tag{4.4}$$

For a multiple sweep evolution $\mathcal{E}_{N\times l}$ a corresponding step cost distribution is defined by the matrix

$$\mathbf{t_{N\times l}} = \begin{pmatrix} t_1^0 & \dots & t_l^0 \\ & \dots & \\ t_1^{N-1} & \dots & t_l^{N-1} \end{pmatrix}. \tag{4.5}$$

$\mathbf{t_{N\times l}}$ contains $N \times l$ elements. Each ith element of the kth row corresponds to the evaluation cost t_i^k, $1 \leq i \leq l$, $0 \leq k \leq N-1$.

If there are at least two indexes i, j, $i \neq j$, for a specified sweep k, so that $t_i^k \neq t_j^k$, $1 \leq i, j \leq l$, we call costs within a single sweep non-uniform. If all N sweeps of a step cost distribution $\mathbf{t_{N\times l}}$ are uniform, it is also uniform. In this case there are N constants $t^0, \dots, t^{N-1}$, so that $t_i^k = t^k$ for all i. Thus, uniform step cost distribution is defined by

$$\mathbf{t_{N\times l}} = \begin{pmatrix} t^0 & \dots & t^0 \\ & \dots & \\ t^{N-1} & \dots & t^{N-1} \end{pmatrix}. \tag{4.6}$$

For the sake of simplicity, we define a uniform step cost distribution by the vector $\vec{\mathbf{t}}_{\mathbf{N\times l}} = (t^0, \dots, t^{N-1})^\top$ of corresponding costs t^k, $0 \leq k \leq N-1$. Note that, according to this definition, evaluation costs of time steps of various sweeps are allowed to take different values. The uniformness of costs is merely provided within particular sweeps.

The size of the ith intermediate state in the kth sweep is denoted $d_i^k = \dim(x_i^k)$, $0 \leq i \leq l$, $0 \leq k \leq N-1$. For a given multiple sweep evolution $\mathcal{E}_{N\times l}$, a corresponding size distribution is defined by the matrix

$$\mathbf{d_{N\times l}} = \begin{pmatrix} d_0^0 & \dots & d_l^0 \\ & \dots & \\ d_0^{N-1} & \dots & d_l^{N-1} \end{pmatrix}, \tag{4.7}$$

so that $\mathbf{d_{N\times l}}$ contains $N \times (l+1)$ elements d_i^k, $0 \leq i \leq l$, $0 \leq k \leq N-1$. In the case of non-uniform sizes there are at least two indexes i, j, $i \neq j$, for the corresponding sweep k, so that $d_i^k \neq d_j^k$, $1 \leq i, j \leq l$. If sizes of intermediate states within single sweeps are constant, then, N constants d^k exist there, so that $d_i^k = d^k$ for all i, $0 \leq i \leq l$, $0 \leq k \leq N-1$. Thus, uniform size distribution is defined by

$$\mathbf{d_{N\times l}} = \begin{pmatrix} d^0 & \dots & d^0 \\ & \dots & \\ d^{N-1} & \dots & d^{N-1} \end{pmatrix}, \tag{4.8}$$

with $d_i^k = d^k,\ 0 \leq i \leq l,\ 0 \leq k \leq N-1$. Similarly to the case of uniform step cost distribution, we define a uniform size distribution by the vector $\vec{\mathbf{d}}_{\mathbf{N}\times\mathbf{l}} = (d^0, \ldots, d^{N-1})^\top$ of corresponding sizes $d^k,\ 0 \leq k \leq N-1$.

Furthermore, we denote

$$d^{even} = \max_{j=0,1,2,\ldots,\lfloor (N-1)/2 \rfloor, i=0,\ldots,l} d_i^{2j}, \tag{4.9}$$

and

$$d^{odd} = \max_{j=0,1,2,\ldots,\lfloor (N-1)/2 \rfloor, i=0,\ldots,l} d_i^{2j+1}, \tag{4.10}$$

maximal sizes of intermediate states for even and odd sweeps, respectively. In the uniform case the notations (4.9) - (4.10) can be simplified to

$$d^{even} = \max_{j=0,1,2,\ldots,\lfloor (N-1)/2 \rfloor} d^{2j}, \tag{4.11}$$

and

$$d^{odd} = \max_{j=0,1,2,\ldots,\lfloor (N-1)/2 \rfloor} d^{2j+1}. \tag{4.12}$$

These notations play a crucial role in the construction of nested reversal schedules.

4.1.2 Definition of Nested Reversal Schedule

Our aim is to implement an evolution $\mathcal{E}_{N\times l}$ using nested checkpointing. Assume, a certain amount of memory is available for this implementation.

The simplest strategy is to implement an evolution $\mathcal{E}_{N\times l}$ with 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 $\mathcal{O}(l\, n^2)$, with l giving the number of time steps between 0 and T. However, this approach can be realised 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 [34, 35, 83], checkpointing means that not all intermediate states are saved but only a small subset of them is stored as checkpoints. In Section 2 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 multiple sweep we are faced here with a new kind of checkpointing task. Since now checkpoints in various sweeps must be kept simultaneously, we refer to this situation as **nested checkpointing**.

Since the amount of information to be stored on various sweeps differs from that needed on other sweeps, we have different classes of checkpoints. It is another difference to the standard checkpointing in Section 2.

The question is how to place various checkpoints to implement the evolution $\mathcal{E}_{N\times l}$ most efficiently. We call each possible strategy a **nested reversal schedule** because checkpoints are set and released at several different levels.

Thus, it is obviously 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 intermediate states of the size d^{odd} defined by (4.10) that it can accommodate.

The implementation of an evolution $\mathcal{E}_{N\times l}$ can be interpreted as a sequence of assignments. They are to combine into appropriate actions. Each action defines which time step is to evaluate, which intermediate state is to store or restore from a corresponding checkpoint. A certain combination of actions represents a **nested reversal schedule**. More formally, use the following definition.

Definition 4.1 (Nested Reversal Schedule S) *Consider an evolution $\mathcal{E}_{N\times l}$ traversing l time steps in N alternative sweeps. Let $C \in \mathbb{N}$ checkpoints be available each of which can accommodate one intermediate state vector of the size d^{odd}. Moreover, assume that checkpoints can be stored during all sweeps except for the final one, provided sufficient memory is available. Then, a* **nested reversal schedule S** *initialises $j^k = 0$ for k being an even number between $0, ..., N-1$, $j^k = l$ for k being an odd number between $0, ..., N-1$, $I^0 = 0$, $I^k = -1$ for $k = 1, 2, \ldots, N-1$, and subsequently performs a sequence of the following basic actions*

$$A^k \quad \equiv \quad \begin{cases} \textit{If } k \textit{ is even and } j^{k-1} - j^k = 1, \textit{carry out the step } F^k_{j^k+1} \\ \textit{and increment } j^k \textit{by } 1, \\ \textit{If } k \textit{ is odd and } j^k - j^{k-1} = 1, \textit{carry out the step } F^k_{j^k} \\ \textit{and decrement } j^k \textit{by } 1, \end{cases}$$

$$W^k_i \quad \equiv \quad \textit{Store data of the state } x^k_{j^k} \textit{ into the checkpoint } i^k = j^k, \; I^k = I^k + 1,$$

$R_i^k \equiv$ *Reset state* $x_{j^k}^k$ *from the checkpoint* $i^k = j^k$, $i \in \{0, 1, 2, \ldots, I^k\}$,

$$D \equiv \begin{cases} \textit{If } N \textit{ is even and } j^{N-2} = l-1, \textit{ carry out the step } F^{N-1}_{j^{N-2}+1}. \\ \textit{Set } l = l-1, \\ \textit{If } N \textit{ is odd and } j^{N-2} = 0^{N-3}+1, \textit{ carry out the step } F^{N-1}_{j^{N-2}} \\ \textit{and store data of the state } x^{N-3}_{j^{N-2}} \textit{ into the checkpoint } 0^{N-3}. \\ \textit{Set } 0^{N-3} = 0^{N-3}+1, \end{cases}$$

until l *has been reduced to* 0 *(if* N *is even) or until* $l - 0^{N-3} = 0$ *(if* N *is odd).*

We normally assume that all nested reversal schedules terminate successfully in that $l = 0$ (if N is even) or $l - 0^{N-3} = 0$ (if N is odd) at the end. We then say that the schedule is admissible for the given combination $(\mathbf{d_{N\times l}}, C)$. Without loss of generality, we assume that each admissible nested reversal schedule $\mathbf{S}$ begins with the action R_0^0, so that the original state x_0^0 is read from the additional checkpoint $0^0 = 0$. Moreover, each other sweep k, $k = 1, \ldots, N-2$, starts with the action W_0^k storing corresponding intermediate states x_0^k for $k = 2, 4, \ldots, N-3$ or x_l^k for $k = 1, 3, 5, \ldots, N-2$ into the additional checkpoint $0^k = 0$ and $0^k = l$, respectively. In Definition 4.1, j^k denotes a currently available intermediate state in the kth sweep, so that $j = 0, 1, \ldots, l$ and $k = 0, 1, \ldots, N-1$. I^k denotes number of checkpoints currently stored in the kth sweep, $k = 0, 1, \ldots, N-2$.

One example of a nested reversal schedule $\mathbf{S}$ for an evolution $\mathcal{E}_{3\times 9}$ with the corresponding size distribution $\vec{\mathbf{d}}_{\mathbf{3\times 9}} = (1, 3, 1)^\top$ is shown in Figure 4.2. $\mathcal{E}_{3\times 9}$ contains 3 alternative sweeps each of which consists of 9 time steps. 1 checkpoint is available, i.e. one intermediate state of the 1st sweep can be stored except for the state which data is stored into the additional checkpoint. Each of these states is of the size $d^{odd} = 3$. Moreover, 3 checkpoints of the size $d^{even} = 1$ can be stored instead of a single one of the size d^{odd}. Furthermore, the original state x_0^0 is stored as an additional checkpoint of the size $d^{even} = 1$.

In Figure 4.2 time steps are plotted along the vertical axis and 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 checkpoint of the size $d^{even} = 1$. Each solid thick horizontal line represents a checkpoint of the size $d^{odd} = 3$. Solid slanted thin lines represent steps F_i^0 of the 0th sweep, whereas steps F_i^1 of

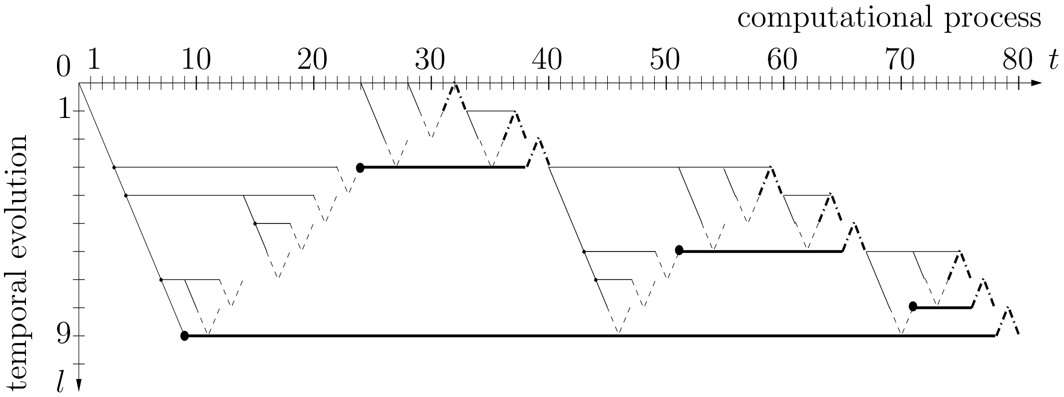

Figure 4.2: Example of a nested reversal schedule **S**

the 1st sweep are visualised by dotted slanted lines. Steps F_i^2 of the 2nd sweep are drawn by slanted dashed-dotted thick lines.

One starts with the action R_0^0 restoring the original state x_0^0 from the $0^0 = 0$ checkpoint. Three actions A^0 are executed by performing three steps F_1^0, F_2^0, and F_3^0, consecutively. The state x_3^0 is stored into the first checkpoint $1^0 = 3$ by the action W_1^0. Now again, the action A^0 is applied to perform one step F_4^0, and the state x_4^0 is stored into the second checkpoint $2^0 = 4$ by the action W_2^0. Furthermore, another three steps are executed by the three actions A^0 and the state x_7^0 is stored into the third checkpoint $3^0 = 7$. Then, two further steps are evaluated by the two actions A^0 and finally, the state x_9^1 is initialised. This state x_9^1 is stored into the additional checkpoint $0^1 = 9$ by the action W_0^1. The 1st sweep is started by this action.

Furthermore, the state x_7^0 is restored from the third checkpoint $3^0 = 7$ by the action R_3^0, the state x_8^0 is reevaluated by the action A^0, and the states x_8^1 and x_7^1 are evaluated by the application of the action A^1 two times. In this manner we come to the state x_3^1, which is to store into the first checkpoint $1^1 = 3$. On the way backward, all checkpoints $1^0 = 3$, $2^0 = 4$, $3^0 = 7$ are removed, all checkpoints $0^1 = 9$ (additional checkpoint), $1^1 = 3$ (one available checkpoint) are occupied consecutively, and we have no more memory available to store checkpoints. Then, one goes back to the state x_1^1 by reevaluating required intermediate states. Consequently, the first final step F_1^2 is performed. Furthermore, one stores the current original state x_1^0 into the checkpoint $0^0 = 1$ and continues in the same manner in order to execute all other final steps $F_2^2, \ldots, F_9^2$.

Using the nested reversal schedule **S** in Figure 4.2, one needs to perform 28 steps F_i^0, 17 steps F_i^1, and 9 steps F_i^2.

The nested reversal schedule **S** in Figure 4.2 performs the following sequence of actions

$$\begin{aligned}
\mathbf{S} &= R_0^0 \triangleright A^0 \triangleright A^0 \triangleright A^0 \triangleright W_1^0 \triangleright A^0 \triangleright W_2^0 \triangleright A^0 \triangleright A^0 \triangleright A^0 \triangleright \\
&\triangleright W_3^0 \triangleright A^0 \triangleright A^0 \triangleright W_0^1 \triangleright R_3^0 \triangleright A^0 \triangleright A^1 \triangleright R_3^0 \triangleright A^1 \triangleright R_2^0 \triangleright \\
&\triangleright A^0 \triangleright W_3^0 \triangleright A^0 \triangleright A^1 \triangleright R_3^0 \triangleright A^1 \triangleright R_2^0 \triangleright A^1 \triangleright R_1^0 \triangleright A^1 \triangleright \\
&\triangleright W_1^1 \triangleright R_0^0 \triangleright A^0 \triangleright A^0 \triangleright A^1 \triangleright R_0^0 \triangleright A^0 \triangleright A^1 \triangleright A^2 \triangleright A^0 \triangleright \\
&\triangleright A^1 \triangleright A^2 \triangleright A^2 \triangleright A^0 \triangleright A^0 \triangleright A^0 \triangleright W_1^0 \triangleright A^0 \triangleright W_2^0 \triangleright A^0 \triangleright \\
&\triangleright A^1 \triangleright R_2^0 \triangleright A^1 \triangleright R_1^0 \triangleright A^1 \triangleright W_1^1 \triangleright R_0^0 \triangleright A^0 \triangleright A^0 \triangleright A^1 \triangleright \\
&\triangleright R_0^0 \triangleright A^0 \triangleright A^1 \triangleright A^2 \triangleright A^0 \triangleright A^1 \triangleright A^2 \triangleright A^2 \triangleright A^0 \triangleright A^0 \triangleright \\
&\triangleright A^1 \triangleright W_1^1 \triangleright R_0^0 \triangleright A^0 \triangleright A^1 \triangleright A^2 \triangleright A^2 \triangleright A^2.
\end{aligned}$$

The operator '$\triangleright$' between two actions can be interpreted in the following way: the left action is evaluated first, the action on the right hand side is evaluated second.

In the following, we introduce some quantities for the characterisation of nested reversal schedules.

4.1.3 Repetition Numbers and Nested Repetition Profile

For a nested reversal schedule **S**, it is convenient to determine the number of evaluations of each time step.

Definition 4.2 (Repetition Number for S) *Consider a nested reversal schedule* **S***, so that* **S** *is admissible for an evolution* $\mathcal{E}_{N\times l}$*. The* **repetition numbers** $r_i^k \equiv r^k(i)$*, defined by the functions*

$$r^k : [1, l] \to \mathbb{N},\ 0 \le k \le N-1, \tag{4.13}$$

count how often the i*th time step* F_i^k *of the* k*th sweep is evaluated during the execution of the nested reversal schedule* **S**.

Definition 4.2 can be interpreted regarding computations of intermediate states. Thus, r_i^k denotes how often the intermediate state x_i^k (if k is even) or the intermediate state x_{i-1}^k (if k is odd) $1 \le i \le l,\ 0 \le k \le N-1$, is computed during the execution of the nested reversal schedule **S**.

Definition 4.3 (Nested Repetition Profile for S) *Consider a nested reversal schedule* $\mathbf{S}$*, so that* $\mathbf{S}$ *is admissible for an evolution* $\mathcal{E}_{N\times l}$*. The matrix*

$$\mathbf{r}(\mathbf{S}) = \langle \vec{r}^{0}, \ldots, \vec{r}^{N-1} \rangle^{\top} = \begin{pmatrix} r_1^0 & \ldots & r_l^0 \\ & \ldots & \\ r_1^{N-1} & \ldots & r_l^{N-1} \end{pmatrix}, \tag{4.14}$$

defines the **Nested Repetition Profile** *for the reversal schedule* $\mathbf{S}$*. Each row of the matrix* $\mathbf{r}(\mathbf{S})$ *is represented by the vector* $\vec{r}^{k\top} = (r_1^k, \ldots, r_l^k)$ *of repetition numbers* r_i^k *for a corresponding sweep* k*,* $0 \leq k \leq N-1$*.*

Definition 4.4 (Maximal Repetition Numbers for S) *For a nested reversal schedule* $\mathbf{S}$*, so that* $\mathbf{S}$ *is admissible for an evolution* $\mathcal{E}_{N\times l}$*, the vector* $\vec{\mathbf{r}}_{max}(\mathbf{S})$ *of maximal repetition numbers is defined by*

$$\vec{\mathbf{r}}_{max}(\mathbf{S}) = \left(\|\vec{r}^{0}\|_\infty, \ldots, \|\vec{r}^{N-1}\|_\infty \right)^{\top} = \left(\max_{1\leq i\leq l} r_i^0, \ldots, \max_{1\leq i\leq l} r_i^{N-1} \right)^{\top}. \tag{4.15}$$

For the example of the reversal schedule $\mathbf{S}$ in Figure 4.2 the corresponding nested repetition profile can be defined by

$$\mathbf{r}(\mathbf{S}) = \begin{pmatrix} 3 & 3 & 1 & 4 & 5 & 3 & 4 & 4 & 1 \\ 0 & 1 & 2 & 1 & 2 & 3 & 2 & 3 & 3 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}. \tag{4.16}$$

This profile is illustrated graphically in Figure 4.3. Each graph in Figure 4.3 corresponds to a single row of the matrix $\mathbf{r}(\mathbf{S})$ in (4.16), 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^k, $1 \leq i \leq 9$. y-axis gives a corresponding repetition number r_i^0, r_i^1, or r_i^2 for a specified time step F_i^0, F_i^1, or F_i^2, respectively. The vector $\vec{\mathbf{r}}_{max}(\mathbf{S})$ of maximal repetition numbers reaches $\vec{\mathbf{r}}_{max}(\mathbf{S}) = (5, 3, 1)^{\top}$.

4.1.4 Temporal Complexity of Nested Reversal Schedules

Provided a schedule $\mathbf{S}$ is admissible in the sense that for given $\mathbf{d}_{\mathbf{N}\times\mathbf{l}}$, C, and the initial l, it successfully reduces l to 0 (if N is even) or it successfully increases 0^{N-1} from 0 to l (if N is odd), its total runtime complexity can be computed from the additional problem parameters $\mathbf{t}_{\mathbf{N}\times\mathbf{l}}$. The temporal

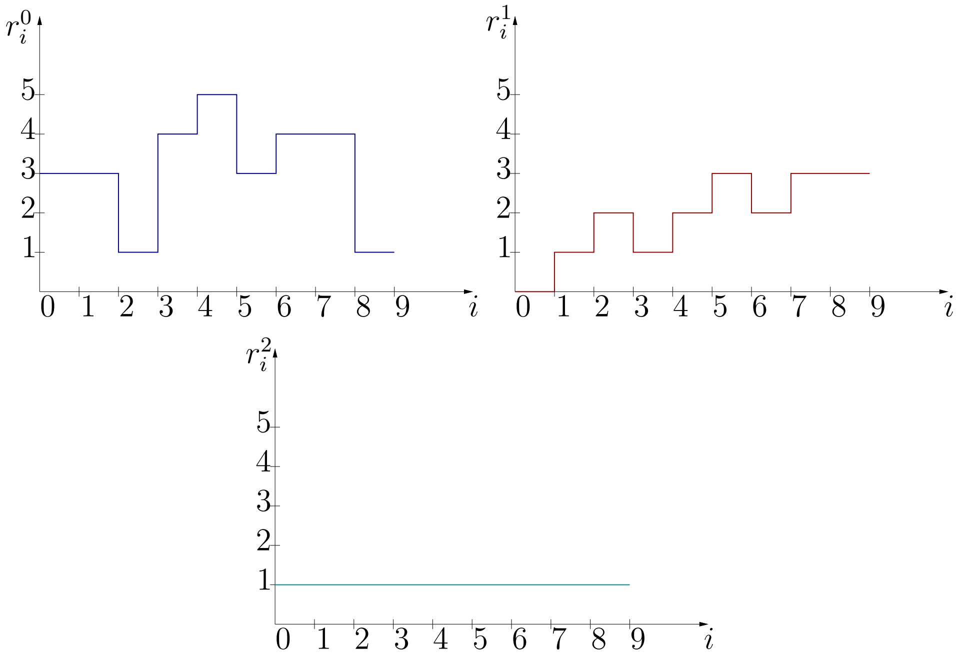

Figure 4.3: Repetition profile for **S** in Figure 4.2

complexity of a nested reversal schedule **S**, i.e. the run-time effort required to execute this nested reversal schedule can be computed by

$$\mathbf{T}(\mathbf{S}) = \sum_{k=0}^{N-1} \left\{ \mathbf{r}(\mathbf{S})^\top \mathbf{t}_{\mathbf{N}\times\mathbf{1}} \right\}_{kk} = \sum_{k=0}^{N-1} \sum_{i=1}^{l} r_i^k t_i^k. \tag{4.17}$$

In the case of uniform step costs $\vec{\mathbf{t}}_{\mathbf{N}\times\mathbf{1}}$ the temporal complexity $\mathbf{T}(\mathbf{S})$ of a nested reversal schedule **S** can be defined by

$$\mathbf{T}(\mathbf{S}) = \sum_{k=0}^{N-1} t^k \sum_{i=1}^{l} r_i^k. \tag{4.18}$$

The inner sum $\sum_{i=1}^{l} r_i^k$ in (4.18) corresponds to the overall number of time steps F_i^k, $1 \leq i \leq l$, of the kth sweep evaluated during the execution of **S**.

The temporal complexity is one of the most important properties of nested reversal schedules. In general, it determines the quality of a corresponding nested reversal schedule. It will be shown in the following how the temporal complexity can be minimised. A definition of an optimal nested reversal schedule is established based on these investigations.

4.1.5 Optimal Nested Reversal Schedules

Our aim is to establish an optimal implementation strategy which allows us to implement the given multiple sweep evolution $\mathcal{E}_{N\times l}$ in minimal time using the available memory.

Obviously, there are several nested reversal schedules for a given multiple sweep evolution $\mathcal{E}_{N\times l}$. We are only interested in the small subset of them, that produce minimal temporal complexity.

We denote

$$Adm(l, N, C) = \{\mathbf{S} : \mathbf{S} \text{ is admissible}\} \tag{4.19}$$

set of all admissible nested reversal schedules $\mathbf{S}$ that can be applied to the implementation of a multiple sweep evolution $\mathcal{E}_{N\times l}$ with a corresponding size distribution $\mathbf{d_{N\times l}}$ and a given number C of checkpoints of the size d^{odd}.

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

$$\mathbf{T}_{min}(l, N, C) \equiv \min \{\mathbf{T}(\mathbf{S}),\ \mathbf{S} \in Adm(l, N, C)\}\,. \tag{4.20}$$

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

$$\mathbf{T}(\mathbf{S}_{min}(l, N, C)) \equiv \mathbf{T}_{min}(l, N, C). \tag{4.21}$$

Now, we face the task of constructing an appropriate optimal nested reversal schedule $\mathbf{S}_{min}(l, N, C)$. There are several ways to construct optimal nested reversal schedules. In the following, we investigate one of them which is in a natural way suitable to the solution of control problems, using the iterative methods in Chapter 3.

Let us look closely on Algorithm 3.7, 3.8 or on Algorithm 3.9 in Chapter 3. Note that the information flow within a single sweep and between different sweeps agrees with the information flow in Figure 4.1. Furthermore, note that, before the execution of each forward sweep, some information has to be provided, namely, control values $\mathbf{u}_i^j$ from the previous Newton's step. Memory has to be available to store this information during the execution of next three sweeps. Then, old values $\mathbf{u}_i^j$ can be overwritten by new evaluated values $\mathbf{u}_i^{j+1}$ required for consecutive three sweeps.

Thus, we can split the overall implementation of the algorithms in Chapter 3 into some groups, each of which contains exactly three sweeps, namely, the original, the adjoint, and the final sweep. Therefore, this group of three sweeps corresponds to a single Newton's step. These single groups are in some sense independent from each over. However, they are to be executed only in a certain order. The $(j+1)$th triple immediately after the jth, since each next group uses the control information $\mathbf{u}_i^j$ computed during the execution of the previous group. For each group, or, for each Newton step will be proved that

$$\frac{\text{Reversal Time}}{\text{Forward Time}} = \mathcal{O}(\ln^2 l), \qquad \frac{\text{Reversal Space}}{\text{Forward Space}} = \mathcal{O}(\ln^2 l), \tag{4.22}$$

with Reversal Time and Reversal Space being time and space, respectively, needed for the implementation of one triple sweep, i.e. one Newton step; and Forward Time and Forward Space being time and space, respectively, needed to implement a forward simulation.

Therefore, in the sequel of this thesis we investigate a special case of multiple sweep evolutions, namely, evolutions containing three sweeps. Then, we can combine them to construct an appropriate nested reversal schedule for an evolution containing any arbitrary number N of sweeps (suitable for the implementation of corresponding algorithms in Chapter 3).

4.2 Special Case: $N = 3$

4.2.1 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 4.4. Time steps are shown as horizontal arrows. Their directions denote the information flow. Nodes denote different intermediate states. Each sweep is characterised by one specified direction,

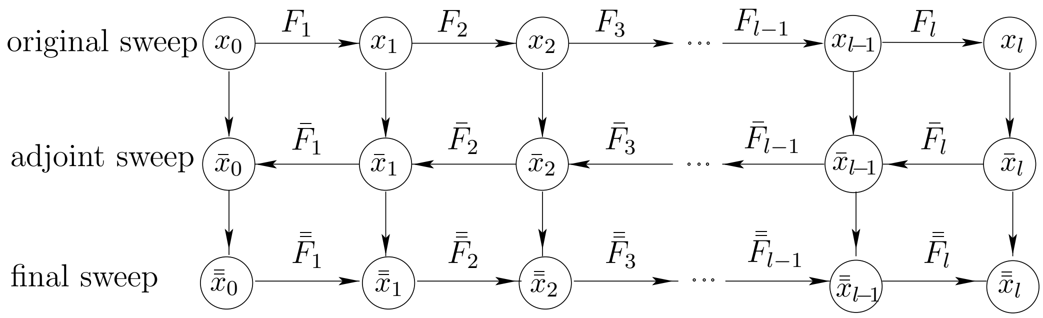

Figure 4.4: Multiple sweep evolution $\mathcal{E}_{3\times l}$

i.e. the direction of horizontal arrows within a single sweep. It shows the corresponding information flow between neighbouring 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 4.4. For the sake of simplicity, the following notations are introduced: we define the first, the second, and the third sweeps of the evolution $\mathcal{E}_{3\times l}$ by the **original**, the **adjoint**, and the **final** sweeps, respectively. Therefore, corresponding time steps and intermediate states are called original steps and original states, adjoint steps and adjoint states, and final steps and final states for original, adjoint, and final sweeps, respectively.

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

$$(4.23)\quad x_i = F_i(x_{i-1}), \quad \bar{x}_{i-1} = \bar{F}_i(x_{i-1}, \bar{x}_i), \quad \bar{\bar{x}}_i = \bar{\bar{F}}_i(\bar{x}_i, \bar{\bar{x}}_{i-1}), \quad 1 \leq i \leq l.$$

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

$$(4.24)\quad d \equiv dim\,\{x_i\}, \quad \bar{d} \equiv dim\,\{\bar{x}_i\}, \quad \bar{\bar{d}} \equiv dim\,\{\bar{\bar{x}}_i\}, \quad 0 \leq i \leq l.$$

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

$$(4.25)\quad t \equiv TIME(F_i), \quad \bar{t} \equiv TIME(\bar{F}_i), \quad \bar{\bar{t}} \equiv TIME(\bar{\bar{F}}_i), \quad 1 \leq i \leq l.$$

Furthermore, we assume that

$$(4.26)\qquad \bar{d} \gg d \qquad \text{and} \qquad \bar{t} \gg t.$$

Thus, the size $\bar{d}$ of the adjoint state is much higher than the size d of the original state. Correspondingly, the evaluation of adjoint steps is much more extensive than the evaluation of original steps. The assumptions (4.26) agree with the scenario presented in Algorithm 3.7, 3.8, and 3.9.

The aim is to implement an evolution $\mathcal{E}_{3\times l}$ using nested checkpointing. The problem is how to place different checkpoints to implement the evolution $\mathcal{E}_{3\times l}$ most efficiently. We call each possible strategy as **nested reversal schedule** because checkpoints are set and released at two different levels. Since the information to be stored on the original sweep differs from that needed on the adjoint sweep, we have two classes of checkpoints. Hence, we call the checkpoints **thin** on the original sweep and **fat** on the adjoint sweep.

Thus, it is obviously 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. 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 original sweep to store thin checkpoints, i.e. checkpoints of the size d, to reduce the total number of evaluated original steps F_i. On the adjoint sweep we remove thin checkpoints sequentially and store fat checkpoints instead of them. This has to be made as soon as required memory is available, i.e. as soon as a sufficient number of thin checkpoints is removed. At this point and throughout the sequel of this thesis we use the notation **S** for any admissible nested reversal schedule which can be applied to a multiple sweep evolution $\mathcal{E}_{3\times l}$ with any arbitrary but fixed size distribution $\vec{\mathbf{d}}_{\mathbf{3}\times\mathbf{l}} = (d, \bar{d}, \bar{\bar{d}})^\top$ and a given number C of fat checkpoints. More formally, we use the following definition.

Definition 4.5 (Nested Reversal Schedule S) *Consider an evolution $\mathcal{E}_{3\times l}$ traversing l time steps in three alternative sweeps. 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, assume that checkpoints can be stored during original and adjoint sweeps, provided sufficient memory is available. Assume that c thin checkpoints can be stored in place of a single fat one, i.e. $\bar{d} = c\,d$. Then, a* **nested reversal schedule S** *initialises $j = 0$, $\bar{j} = l$, $\check{l}_0 = 0$, and subsequently performs a sequence of the following basic actions*

A	$\equiv$	*Carry out the original step F_{j+1}. Increment j by 1,*
$\bar{A}$	$\equiv$	*If $\bar{j} - j = 1$, carry out the adjoint step $\bar{F}_{\bar{j}}$. Decrement $\bar{j}$ by 1,*
W_i	$\equiv$	*Store data of the original state x_j into the thin checkpoint $i \in \{0, 1, \ldots, cC\}$. Set $\check{l}_i = j$,*
$\bar{W}_{\bar{i}}$	$\equiv$	*Store data of the adjoint state $\bar{x}_{\bar{j}}$ into the fat checkpoint $\bar{i} \in \{0, 1, \ldots, C\}$. Set $\hat{l}_{\bar{i}} = \bar{j}$,*
R_i	$\equiv$	*Reset original state x_j from the thin checkpoint $i \in \{0, 1, \ldots, cC\}$,*
$\bar{R}_{\bar{i}}$	$\equiv$	*Reset adjoint state $\bar{x}_{\bar{j}}$ from the fat checkpoint $\bar{i} \in \{0, 1, \ldots, C\}$,*

$D \;\equiv\;$ *If* $\bar{j} = \check{l}_0 + 1$, *carry out the final step* $\bar{\bar{F}}_{\bar{j}}$ *and store data of the original state* $x_{\bar{j}}$ *into the* 0*th thin checkpoint. Set* $\check{l}_0 = \check{l}_0 + 1$,

until $\check{l}_0$ *has been increased up to* l, *i.e. until* $l - \check{l}_0 = 0$.

In Definition 4.5 and in the sequel of this thesis we use the following notations:

$\check{l}_i \;\equiv$ a counter j of an original state x_j which data is stored into the ith thin checkpoint ($i = 0, \ldots, cC$, $j = 0, \ldots, l$, $\check{l}_0 = 0$, $\check{l}_i < \check{l}_{i+1}$, i.e. thin checkpoints are numbered starting from the beginning),

$\hat{l}_{\bar{i}} \;\equiv$ a counter $\bar{j}$ of an adjoint state $\bar{x}_{\bar{j}}$ which data is stored into the $\bar{i}$th fat checkpoint ($\bar{i} = 0, \ldots, C$, $\bar{j} = 0, \ldots, l$, $\hat{l}_0 = l$, $\hat{l}_i > \hat{l}_{i+1}$, i.e. fat checkpoints are numbered starting from the end).

We normally assume that all nested reversal schedules terminate successfully in that $l - \check{l}_0 = 0$ at the end. We then say that the schedule is admissible for the given combination $(\mathbf{d_{3\times l}}, C)$. Without loss of generality, we assume that each admissible nested reversal schedule S begins with the action R_0, so that the original state x_0 is read out from the thin checkpoint 0. Moreover, immediately after the first evaluation of the adjoint state $\bar{x}_l$, in most cases after performing the entire original sweep for the first time, data of the adjoint state $\bar{x}_l$ has to be stored into the additional, the so called 0th fat checkpoint. Data of the adjoint state $\bar{x}_l$ has to be stored in the additional fat checkpoint during the overall execution of a nested reversal schedule $\mathbf{S}$.

Definition 4.6 (Repetition Numbers for S) *Consider a nested reversal schedule* $\mathbf{S}$, *so that* $\mathbf{S}$ *is admissible for an evolution* $\mathcal{E}_{3\times l}$. *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*

$$r, \bar{r}, \bar{\bar{r}} : [1, l] \to \mathbb{N}, \tag{4.27}$$

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

Provided a schedule is admissible in the sense that for given $\vec{\mathbf{d}}_{\mathbf{3}\times\mathbf{l}}$, C, and the initial l, it successfully increases $\check{l}_0$ from 0 up to l, its total runtime complexity can be computed from the additional problem parameters $\vec{\mathbf{t}}_{\mathbf{3}\times\mathbf{l}} = (t, \bar{t}, \bar{\bar{t}})^\top$. The temporal complexity of a nested reversal schedule $\mathbf{S}$, i.e. the run-time effort required to execute this nested reversal schedule can be computed by

$$\mathbf{T}(\mathbf{S}) = t\sum_{i=1}^{l} r_i + \bar{t}\sum_{i=1}^{l} \bar{r}_i + \bar{\bar{t}}\sum_{i=1}^{l} \bar{\bar{r}}_i. \tag{4.28}$$

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

$$\mathbf{T}_{min}(l, C) \equiv \min \{\mathbf{T}(\mathbf{S}),\ \mathbf{S} \text{ is admissible}\}. \tag{4.29}$$

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

$$\mathbf{T}(\mathbf{S}_{min}(l, C)) \equiv \mathbf{T}_{min}(l, C). \tag{4.30}$$

Now, we face the task of constructing an appropriate optimal nested reversal schedule $\mathbf{S}_{min}(l, C)$.

4.2.2 Structural Properties

To construct optimal nested reversal schedules it is convenient to give an impression how thin and fat checkpoints are placed and exchanged.

As mentioned above, thin checkpoints are placed along the original, fat checkpoints along the adjoint sweep. During the final sweep no checkpoints are required to be stored.

If C fat checkpoints are available and c thin checkpoints can be stored in place of a single fat one, then, obviously, there is memory for cC thin checkpoints available. Recall, the initial original state x_0 is stored into the additional thin checkpoint. Similarly, the last adjoint state $\bar{x}_l$ is stored into the additional fat checkpoint. During the execution of a nested reversal schedule the content of the additional thin checkpoint is changing, namely, after performing each final step $\bar{\bar{F}}_i$, $i = 1, \ldots, l, 1$ the initial original state obtains the value of x_i. Therefore, the previous content of the additional thin checkpoint, i.e. data of the original state x_{i-1}, is to be overwritten with data of the original state x_i. The content of the additional fat checkpoint, namely, data of the adjoint state $\bar{x}_l$, is not changing during the overall execution of a nested reversal schedule. A corresponding initial original state is to be kept in the additional thin checkpoint during the overall execution of a nested reversal schedule.

In order to initialise the adjoint state $\bar{x}_l$, we have to execute the entire original sweep at least one time. After storing the adjoint state $\bar{x}_l$ into the additional fat checkpoint, memory for C fat or cC thin checkpoints is available there. Obviously, we can already use this memory during the original sweep to store thin checkpoints along the execution of this sweep. In this way, the overall evaluation cost of a corresponding nested reversal schedule can be reduced.

Moreover, each nested reversal schedule $\mathbf{S}$ can be modified without increasing the evaluation cost $\mathbf{T}(\mathbf{S})$, so that each final step $\bar{\bar{F}}_i$, $i = 1, \ldots, l$, is executed exactly once.

To establish a search algorithm for the construction of optimal nested reversal schedules with uniform step costs, these schedules will be decomposed into smaller substructures which will be regarded separately. For that end, it is necessary to prove the following two assertions.

Theorem 4.1 (Nested Checkpoint Persistence) *Consider an evolution $\mathcal{E}_{3\times l}$ traversing l time steps in three alternative sweeps. $\vec{\mathbf{t}}_{\mathbf{3}\times\mathbf{l}}$ and $\vec{\mathbf{d}}_{\mathbf{3}\times\mathbf{l}}$ are the 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 $\bar{x}_i$, $0 \leq i \leq l$. Assume that c thin checkpoints can be stored in place of a single fat one, i.e. $\bar{d} = c\,d$. Let $\mathbf{S}$ be an admissible nested reversal schedule which can be applied to the implementation of the evolution $\mathcal{E}_{3\times l}$ using the given parameters. Then, $\mathbf{S}$ can be modified without increasing its evaluation cost $\mathbf{T}(\mathbf{S})$, so that the following properties hold. After the thin checkpoint writing W_i, copying the original state x_j into the thin checkpoint i, the next action W_i occurs only when $\bar{j}$ has already been reduced to j or below during the next adjoint sweep. Moreover, until that time no actions are taken involving the original states between x_0 and x_j.*

Furthermore, $\mathbf{S}$ can be modified without increasing its evaluation cost $\mathbf{T}(\mathbf{S})$, so that the following properties hold. After the fat checkpoint writing $\bar{W}_{\bar{i}}$, copying the adjoint state $\bar{x}_{\bar{j}}$ into the fat checkpoint $\bar{i}$, the next action $\bar{W}_{\bar{i}}$ occurs only when the overall number l of final steps has already been reduced to $(l-\bar{j})$ or below, i.e. when $\check{l}_0 \geq \bar{j}$. Moreover, until that time no actions are taken involving the original states between $x_{\bar{j}}$ and x_l and the adjoint states between $\bar{x}_{\bar{j}}$ and $\bar{x}_l$.

Proof:

The concept of **checkpoint persistence** has already been introduced for serial reversal schedules in Section 2 as well as in [35, 91]. The proof of Theorem 4.1 relies on the proof given in [35] or in [91].

To prove the first of the two assertions, we consider two first sweeps of the evolution $\mathcal{E}_{3\times l}$, namely, the original and the adjoint sweep, and omit the final sweep for a moment. First, prove that after the thin checkpoint writing W_i, copying the original state x_j into the thin checkpoint i, no actions are taken, involving the original states between x_0 and x_j, before $\bar{j}$ has been reduced to j. Assume that this assertion is not true, i.e. some actions are taken, involving the original states between x_0 and x_j, before $\bar{j}$ has been reduced to j. Since all actions regarding adjoint steps or states between x_0 and x_j are excluded, the only actions, which can be taken, are A, W_k, and R_k. It is reasonable to consider a combinations of these actions, since the action

R_k, performed without a consecutive action A, can be omitted immediately. Similarly, it is clear that each action W_k or A requires some actions R_k and A to be performed imminent before. Otherwise, this action W_k can also be omitted.

Therefore, we consider two combinations. The first one is

$$R_k + \underbrace{A + \cdots + A}_{j_2 - j_1} + W_g,$$

so that $k \in \{0, \ldots, i-1\}$ and $g \in \{i, \ldots, cC\}$. This means that the original state x_{j_1} with $j_1 < j$ is read out from the thin checkpoint k. Moreover, $j_2 - j_1$ actions A are performed, and, finally, data of the currently evaluated original state x_{j_2} is stored into the thin checkpoint g. If $j_2 > j$, then, obviously, we can improve the resulting nested reversal schedule in terms of its temporal complexity (this means that the number of evaluated original steps can be reduced), if we replace the combination above by the following one:

$$R_i + \underbrace{A + \cdots + A}_{j_2 - j} + W_g.$$

If $j_2 < j$, then, obviously, we can improve the resulting nested reversal schedule in terms of its temporal complexity, if we perform the combination $R_k + \underbrace{A + \cdots + A}_{j_2 - j_1} + W_g$ after $\bar{j}$ has been reduced to j, and until this point we use the thin checkpoint g in order to reduce the temporal complexity; or, if we store the original state x_{j_2} into the thin checkpoint g before the original state x_j is stored into the checkpoint i.

The second situation is

$$R_k + \underbrace{A + \cdots + A}_{j_2 - j_1} + \bar{A},$$

so that $k \in 0, \ldots, i-1$. This means that the original state x_{j_1} with $j_1 < j$ is read out from the thin checkpoint k. Moreover, $j_2 - j_1$ actions A are performed, and, finally, the adjoint step $\bar{F}_{\bar{j}}$ is performed, since the equation $\bar{j} - j_2 = 1$ is satisfied. Since in this case the relation $j_2 > j$ is satisfied (otherwise $\bar{j}$ has already been reduced to j), then, obviously, we can improve the resulting nested reversal schedule in terms of its temporal complexity, if we replace the combination above by the following one:

$$R_i + \underbrace{A + \cdots + A}_{j_2 - j} + \bar{A}.$$

Therefore, we have proved that after the thin checkpoint writing W_i, copying the original state x_j into the thin checkpoint i, no actions are taken, involving the original states between x_0 and x_j, before $\bar{j}$ has been reduced to j.

Next, we prove that after the thin checkpoint writing W_i, copying the original state x_j into the thin checkpoint i, the next action W_i occurs only when $\bar{j}$ has already been reduced to j or below. Assume that this is not satisfied, i.e. after the thin checkpoint writing W_i, copying the original state x_j into the thin checkpoint i, the original state x_{j_1} is stored into the thin checkpoint i. Because it was proved recently that after the thin checkpoint writing W_i no actions are taken, involving the original states between x_0 and x_j, before $\bar{j}$ has been reduced to j, the relation $j_1 > j$ is satisfied. Then, obviously, the first action W_i, copying the original state x_j into the thin checkpoint i, can be omitted without increasing the overall temporal complexity.

To prove the second assertion of Theorem 4.1, we consider the adjoint sweep and the final sweep of the evolution $\mathcal{E}_{3\times l}$, and omit the original sweep. Then, we proceed as above for proving the first assertion of Theorem 4.1. First, prove that after the fat checkpoint writing $\bar{W}_{\bar{i}}$, copying the adjoint state $\bar{x}_{\bar{j}}$ into the fat checkpoint $\bar{i}$, no actions are taken involving the original states between $x_{\bar{j}}$ and x_l and the adjoint states between $\bar{x}_{\bar{j}}$ and $\bar{x}_l$ before the overall number l of final steps has already been reduced to $(l-\bar{j})$ or below, i.e. before $\check{l}_0 \geq \bar{j}$. Assume that this assertion is not true, i.e. some actions, involving the original states between $x_{\bar{j}}$ and x_l and the adjoint states between $\bar{x}_{\bar{j}}$ and $\bar{x}_l$, are taken before the overall number l of final steps has already been reduced to $(l-\bar{j})$ or below, i.e. before $\check{l}_0 \geq \bar{j}$. Without loss of generality, we can consider all actions regarding original steps or states as additional help-actions for performing adjoint steps. If we prove that no actions have to be taken, involving the adjoint states between $\bar{x}_{\bar{j}}$ and $\bar{x}_l$, before the overall number l of final steps has already been reduced to $(l - \bar{j})$ or below, then, it implies that no actions, involving the original states between $x_{\bar{j}}$ and x_l, are required also. Therefore, the only actions, which can be taken, are $\bar{A}$, $\bar{W}_{\bar{k}}$, and $\bar{R}_{\bar{k}}$. As before, it is reasonable to consider a combinations of these actions, since the action $\bar{R}_{\bar{k}}$, performed without a consecutive action $\bar{A}$, can be omitted immediately. Similarly, it is clear that each action $\bar{W}_{\bar{k}}$ or $\bar{A}$ requires some actions $\bar{R}_{\bar{k}}$ and $\bar{A}$ to be performed imminent before. Otherwise, this action $\bar{W}_{\bar{k}}$ can also be omitted.

Therefore, we consider two combinations. The first one is

$$\bar{R}_{\bar{k}} + \underbrace{\bar{A} + \cdots + \bar{A}}_{\bar{j}_1 - \bar{j}_2} + \bar{W}_{\bar{g}},$$

so that $\bar{k} \in \{0, \ldots, \bar{i} - 1\}$ and $g \in \{\bar{i}, \ldots, C\}$. This means that the adjoint state $\bar{x}_{\bar{j}_1}$ with $\bar{j}_1 > \bar{j}$ is read out from the fat checkpoint $\bar{k}$. Moreover, $\bar{j}_1 - \bar{j}_2$ actions $\bar{A}$ are performed, and, finally, data of the currently evaluated adjoint state $\bar{x}_{\bar{j}_2}$ is stored into the fat checkpoint $\bar{g}$. Obviously, each adjoint step $\bar{A}$ requires some number of original steps, that are considered as an additional cost. If $\bar{j}_2 < \bar{j}$, then, obviously, we can improve the resulting nested reversal schedule in terms of its temporal complexity, if we replace the combination above by the following one

$$\bar{R}_{\bar{i}} + \underbrace{\bar{A} + \cdots + \bar{A}}_{\bar{j}-\bar{j}_2} + \bar{W}_{\bar{g}}.$$

If $\bar{j}_2 > \bar{j}$, then, obviously, we can improve the resulting nested reversal schedule in terms of its temporal complexity, if we perform the combination $\bar{R}_{\bar{k}} + \underbrace{\bar{A} + \cdots + \bar{A}}_{\bar{j}_1-\bar{j}_2} + \bar{W}_{\bar{g}}$ after the overall number l of final steps has been reduced to $(l - \bar{j})$ or below, and until this point we use the fat checkpoint $\bar{g}$ in order to reduce the overall temporal complexity; or, if we store the adjoint state $\bar{x}_{\bar{j}_2}$ into the fat checkpoint $\bar{g}$ before the adjoint state $\bar{x}_{\bar{j}}$ is stored into the checkpoint $\bar{i}$.

The second situation is

$$\bar{R}_{\bar{k}} + \underbrace{\bar{A} + \cdots + \bar{A}}_{\bar{j}_1-\bar{j}_2} + D,$$

so that $\bar{k} \in 0, \ldots, \bar{i} - 1$. This means that the adjoint state $\bar{x}_{\bar{j}_1}$ with $\bar{j}_1 > \bar{j}$ is read out from the fat checkpoint $\bar{k}$. Moreover, $\bar{j}_1 - \bar{j}_2$ actions $\bar{A}$ are performed, and, finally, the final step $\bar{\bar{F}}_{\bar{j}_2}$ is performed, since the equation $\bar{j}_2 = \check{l}_0 + 1$ is satisfied. Since in this case the relation $\bar{j}_2 < \bar{j}$ is fulfilled (otherwise the overall number l of final steps has already been reduced to $(l - \bar{j})$ or below), then, obviously, we can improve the resulting nested reversal schedule in terms of its temporal complexity, if we replace the combination above by the following one

$$\bar{R}_{\bar{i}} + \underbrace{\bar{A} + \cdots + \bar{A}}_{\bar{j}-\bar{j}_2} + D.$$

Therefore, we have proved that after the fat checkpoint writing $\bar{W}_{\bar{i}}$, copying the adjoint state $\bar{x}_{\bar{j}}$ into the fat checkpoint $\bar{i}$, no actions are taken, involving the original states between $x_{\bar{j}}$ and x_l and the adjoint states between $\bar{x}_{\bar{j}}$ and $\bar{x}_l$, before the overall number l of final steps has already been reduced to $(l - \bar{j})$ or below.

Next, we prove that after the fat checkpoint writing $\bar{W}_{\bar{i}}$, copying the adjoint state $\bar{x}_{\bar{j}}$ into the fat checkpoint $\bar{i}$, the next action $\bar{W}_{\bar{i}}$ occurs only when the overall number l of final steps has already been reduced to $(l - \bar{j})$ or below, i.e. when $\check{l}_0 \geq \bar{j}$. Assume that this is not satisfied, i.e. after the fat checkpoint writing $\bar{W}_{\bar{i}}$, the adjoint state $\bar{x}_{\bar{j}_1}$ is stored into the fat checkpoint $\bar{i}$. Because it was recently proved that after the fat checkpoint writing $\bar{W}_{\bar{i}}$, copying the adjoint state $\bar{x}_{\bar{j}}$ into the fat checkpoint $\bar{i}$, no actions are taken, involving the original states between $x_{\bar{j}}$ and x_l and the adjoint states between $\bar{x}_{\bar{j}}$ and $\bar{x}_l$, before the overall number l of final steps has already been reduced to $(l - \bar{j})$, the relation $\bar{j}_1 < \bar{j}$ is satisfied. Then, obviously, the first action $\bar{W}_{\bar{i}}$, copying the adjoint state $\bar{x}_{\bar{j}}$ into the fat checkpoint $\bar{i}$, can be omitted without increasing the overall temporal complexity.

Therefore, Theorem 4.1 is proven.

□

Recall, fat checkpoints can be stored only if required memory is available. If we store cC thin checkpoints during the original sweep, then, there is no memory available for fat checkpoints. Thus, we have to remove thin checkpoints during the adjoint sweep in order to provide required memory for fat checkpoints. The next theorem proves that, in order to reduce the overall evaluation cost for the implementation of an evolution $\mathcal{E}_{3\times l}$, it is more convenient to remove all thin checkpoints and to store as many fat checkpoints as possible.

Theorem 4.2 *Consider an evolution $\mathcal{E}_{3\times l}$ traversing l time steps in three alternative sweeps. $\vec{\mathbf{t}}_{\mathbf{3}\times\mathbf{l}}$ and $\vec{\mathbf{d}}_{\mathbf{3}\times\mathbf{l}}$ are the 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 $\bar{x}_i$, $0 \leq i \leq l$. Assume that c thin checkpoints can be stored in place of a single fat one, i.e. $\bar{d} = c\,d$. Let $\mathbf{S}$ be an admissible nested reversal schedule which can be applied to the implementation of the evolution $\mathcal{E}_{3\times l}$ using the given parameters. Then, $\mathbf{S}$ can be modified without increasing the evaluation cost $\mathbf{T}(\mathbf{S})$, so that the following properties hold. During the execution of the nested reversal schedule $\mathbf{S}$ all thin checkpoints are stored behind the corresponding last fat checkpoint. Moreover, during the evaluation of each final step $\bar{\bar{F}}_i$, $1 \leq i \leq l$, all stored checkpoints are fat checkpoints.*

Proof: Suppose, there is a nested reversal schedule $\mathbf{S}$ for the evolution $\mathcal{E}_{3\times l}$, so that during its execution at least a single thin checkpoint is left stored before the corresponding fat checkpoint or during performing of a final step $\bar{\bar{F}}_i$. In Figure 4.5 we consider a subset of an evolution $\mathcal{E}_{3\times l}$ between the original states $x_b, \ldots, x_e$, the adjoint states $\bar{x}_b, \ldots, \bar{x}_e$, and the final states $\bar{\bar{x}}_b, \ldots, \bar{\bar{x}}_e$. Suppose, during the execution of a nested reversal schedule $\mathbf{S}$, applied to the

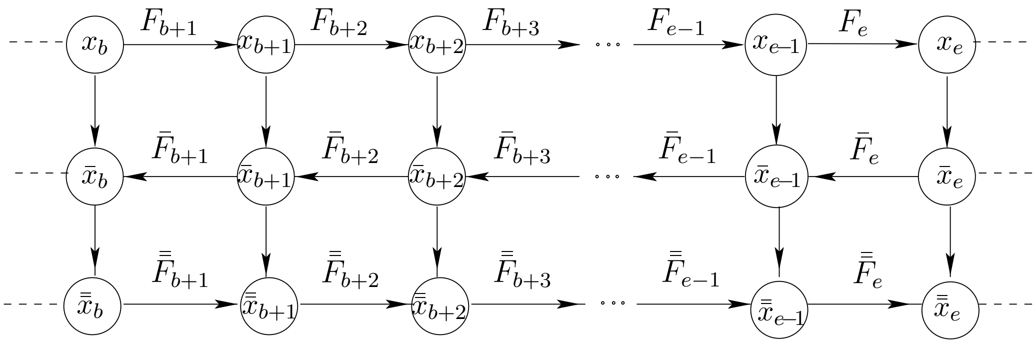

Figure 4.5: A subset of a multiple sweep evolution $\mathcal{E}_{3\times l}$

evolution $\mathcal{E}_{3\times l}$, the original state x_b is stored into a thin checkpoint, the adjoint state $\bar{x}_e$ is stored into a fat checkpoint. Moreover, there is only a single adjoint state between $\bar{x}_b$ and $\bar{x}_e$ which can be stored into a fat checkpoint.

Without loss of generality, we can define $b = 0$ and $e = l$. Moreover, one fat checkpoint (or c thin checkpoints) is (are) available altogether, i.e. $C = 1$. The initial original state is stored into the additional thin checkpoint. The last adjoint state is stored into the additional fat checkpoint immediately after performing the entire original sweep for the first time. Moreover, there is at least a single $\check{l}$, $0 < \check{l} < l$, so that the original state $x_{\check{l}}$ is stored into a thin checkpoint during the execution of the original sweep, and is to be left stored there during the execution of the final steps $\bar{\bar{F}}_1$ and $\bar{\bar{F}}_2$. Thus, we compare the minimal evaluation costs, resulting in the case of stored thin checkpoints and in the case of removed thin checkpoints. In this manner, we prove that there is no optimal reversal schedule with thin checkpoints stored before the corresponding fat checkpoint or during performing the corresponding final step. If all thin checkpoints are removed during the adjoint sweep and available memory is used for storing fat checkpoints, the resulting evaluation cost will be reduced. In the following, we compare the minimal costs, resulting from the evaluation of two final steps $\bar{\bar{F}}_1$ and $\bar{\bar{F}}_2$, for these two cases. Firstly, we consider the simplest case.

First case: $C = 1, c = 1$

In this case one fat checkpoint is available. Moreover, one thin checkpoint can be stored in place of a single fat one. An example for thin and fat checkpoint distribution taking place after performing the adjoint sweep is depicted in Figure 4.6. Figure 4.6 illustrates two sweeps, the original and the adjoint one. Intermediate states, stored as checkpoints, are identified with boxes; thin checkpoints with thin boxes, fat checkpoints with fat ones. In Figure 4.6 we assume that during the adjoint sweep the single thin checkpoint, containing

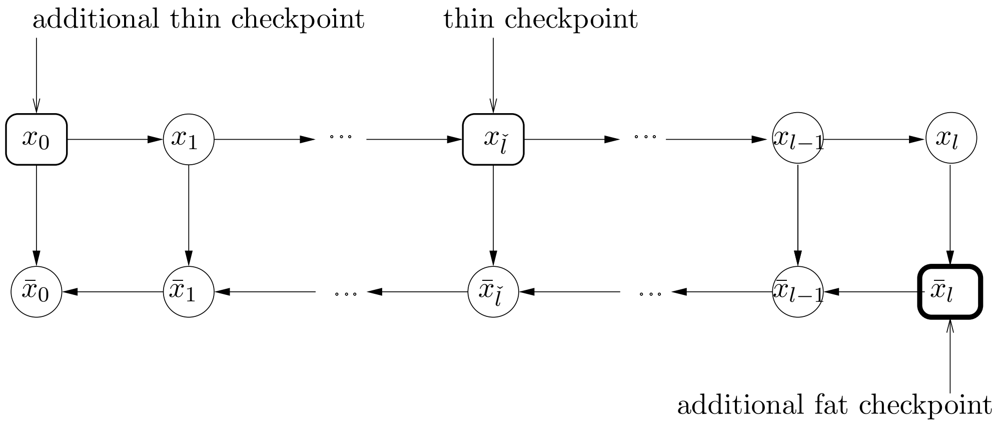

Figure 4.6: Example for a checkpoint distribution with $C = 1$ and $c = 1$, so that the thin checkpoint is left stored after performing the adjoint sweep

data of the original state $x_{\check{l}}$, is not removed. Therefore, memory needed for storing the fat checkpoint is not available. The only fat checkpoint, which is stored, is the additional fat checkpoint. The minimal temporal complexity $\mathbf{T}^1_{min}(\bar{\bar{F}}_1 + \bar{\bar{F}}_2)$, resulting from the evaluation of the final steps $\bar{\bar{F}}_1$ and $\bar{\bar{F}}_2$ for this case, can be determined by

$$\begin{aligned} \mathbf{T}^1_{min}(\bar{\bar{F}}_1 + \bar{\bar{F}}_2) &= \mathbf{T}_f t + (l-1)\bar{t} + \bar{\bar{t}} + \\ &+ \left[\mathbf{T}_{bin}(l - \check{l}, 1) + \mathbf{T}_{bin}(\check{l} - 1, 1)\right] t + (l-2)\bar{t} + \bar{\bar{t}}. \end{aligned} \tag{4.31}$$

Another situation is depicted in Figure 4.7. Here, the thin checkpoint, stored during the original sweep, is removed during the adjoint sweep. The fat checkpoint is stored in place of the thin one throughout performing the adjoint sweep. The minimal temporal complexity $\mathbf{T}^2_{min}(\bar{\bar{F}}_1 + \bar{\bar{F}}_2)$, resulting from the evaluation of the final steps $\bar{\bar{F}}_1$ and $\bar{\bar{F}}_2$ for this case, can be determined by

$$\begin{aligned} \mathbf{T}^2_{min}(\bar{\bar{F}}_1 + \bar{\bar{F}}_2) &= \mathbf{T}_f t + (l-1)\bar{t} + \bar{\bar{t}} + \\ &+ \left[\mathbf{T}_{bin}(\check{l} - 1, 1)\right] t + (\check{l}-2)\bar{t} + \bar{\bar{t}}. \end{aligned} \tag{4.32}$$

$\mathbf{T}_{bin}(l, c)$ in (4.31) and (4.32) is the minimal number of original steps required for the reversal of a sequence of l original steps, so that up to c checkpoints can be accommodated at any time (see Chapter 2). $\mathbf{T}_f$ in (4.31) and (4.32) corresponds to the number of original steps which are performed in order to evaluate the final step $\bar{\bar{F}}_1$. The difference between $\mathbf{T}^1_{min}(\bar{\bar{F}}_1 + \bar{\bar{F}}_2)$ and $\mathbf{T}^2_{min}(\bar{\bar{F}}_1 + \bar{\bar{F}}_2)$ is that in the first case the original state $x_{\check{l}}$ is stored into the thin checkpoint, in the second case the adjoint state $\bar{x}_{\check{l}}$ is stored into the fat checkpoint. Obviously, the number $\mathbf{T}_f$ remains the same in both cases.

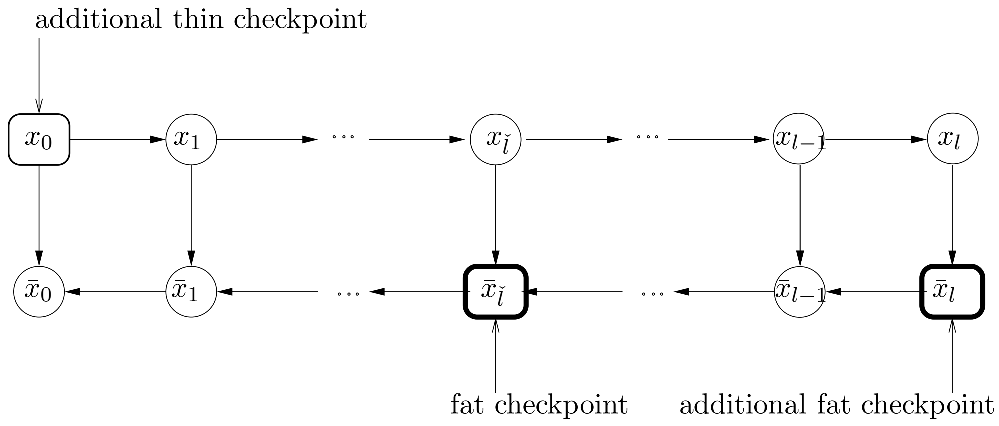

Figure 4.7: Example for a checkpoint distribution with $C = 1$ and $c = 1$, so that the thin checkpoint is removed during performing the adjoint sweep. The fat checkpoint is stored in place of it

In both equations the first lines, i.e.

$$\mathbf{T}_f t + (l-1)\bar{t} + \bar{\bar{t}},$$

correspond to the temporal complexity, resulting from evaluating the first final step $\bar{\bar{F}}_1$. The second lines of the equations (4.31) and (4.32) correspond to the temporal complexity, resulting from evaluating the second final step $\bar{\bar{F}}_2$. Obviously,

$$\left[\mathbf{T}_{bin}(\check{l}-1,1)\right] t + (\check{l}-2)\bar{t} + \bar{\bar{t}} < \left[\mathbf{T}_{bin}(l-\check{l},1) + \mathbf{T}_{bin}(\check{l}-1,1)\right] t + (l-2)\bar{t} + \bar{\bar{t}},$$

which implies, that $\mathbf{T}^2_{min}(\bar{\bar{F}}_1 + \bar{\bar{F}}_2) < \mathbf{T}^1_{min}(\bar{\bar{F}}_1 + \bar{\bar{F}}_2)$.

Second case: $C = 1$, c arbitrary

In this case one fat checkpoint is available. Moreover, c thin checkpoint can be stored in place of a single fat one. First, consider the case, so that a single thin checkpoint is left stored after performing the adjoint sweep (see Figure 4.6). Therefore, memory needed for storing the fat checkpoint is not available. For the minimal temporal complexity $\mathbf{T}^1_{min}(\bar{\bar{F}}_1 + \bar{\bar{F}}_2)$, resulting from the evaluation of the final steps $\bar{\bar{F}}_1$ and $\bar{\bar{F}}_2$, the following estimation is satisfied

$$\begin{aligned} \mathbf{T}^1_{min}(\bar{\bar{F}}_1 + \bar{\bar{F}}_2) \;&\geq\; \left[\mathbf{T}_{bin}(l+1,c+1)\right] t + (l-1)\bar{t} + \bar{\bar{t}} + \\ &+\; \left[\mathbf{T}_{bin}(l-\check{l},c) + \mathbf{T}_{bin}(\check{l}-1,c)\right] t + (l-2)\bar{t} + \bar{\bar{t}} = \mathbf{T}^{low}_1. \end{aligned} \tag{4.33}$$

Transform the situation described by Figure 4.6 and equation (4.33) in the following way. First, choose a value $\hat{l}$, $0 < \hat{l} < \check{l}$. Then, remove all thin checkpoints, including the single one, where data of the original state $x_{\check{l}}$ is stored, during performing the adjoint sweep. Therefore, there is memory available for setting the fat checkpoint. Finally, store the adjoint state $\bar{x}_{\hat{l}}$ into the fat checkpoint. The minimal temporal complexity $\mathbf{T}^2_{min}(\bar{\bar{F}}_1 + \bar{\bar{F}}_2)$, resulting from the evaluation of the final steps $\bar{\bar{F}}_1$ and $\bar{\bar{F}}_2$ for this case, can be determined by

$$\begin{aligned} \mathbf{T}^2_{min}(\bar{\bar{F}}_1 + \bar{\bar{F}}_2) &= \Big[\mathbf{T}_{bin}(\hat{l}+1,1) + \mathbf{T}_{bin}(l-\hat{l}+1,c)\Big]\, t + (l-1)\bar{t} + \bar{\bar{t}} + \\ &+ \Big[\mathbf{T}_{bin}(\hat{l}-1,1)\Big]\, t + (\hat{l}-2)\bar{t} + \bar{\bar{t}}. \end{aligned} \tag{4.34}$$

If we compare $\mathbf{T}^{low}_1$ and $\mathbf{T}^2_{min}(\bar{\bar{F}}_1 + \bar{\bar{F}}_2)$ in (4.33) and (4.34) and recall the assumption $\bar{t} >> t$, we obtain

$$\begin{aligned} \mathbf{T}^{low}_1 &- \mathbf{T}^2_{min}(\bar{\bar{F}}_1 + \bar{\bar{F}}_2) = \big[\mathbf{T}_{bin}(l+1,c+1) + \mathbf{T}_{bin}(l-\check{l},c) + \mathbf{T}_{bin}(\check{l}-1,c) - \\ &- \mathbf{T}_{bin}(\hat{l}+1,1) - \mathbf{T}_{bin}(l-\hat{l}+1,c) - \mathbf{T}_{bin}(\hat{l}-1,1)\Big]\, t + (l-\hat{l})\bar{t} \ge 0, \end{aligned} \tag{4.35}$$

since we can choose $\hat{l}$ sufficiently small. (4.35) implies that $\mathbf{T}^2_{min}(\bar{\bar{F}}_1 + \bar{\bar{F}}_2) \le \mathbf{T}^{low}_1 \le \mathbf{T}^1_{min}(\bar{\bar{F}}_1 + \bar{\bar{F}}_2)$.

Obviously, if the evaluation of the final steps $\bar{\bar{F}}_1$ and $\bar{\bar{F}}_2$ is more convenient with the removing of all thin checkpoints during the adjoint sweep, then, this result remains true for the evaluation of other final steps. Therefore, the assertion of the theorem is proved, i.e. each nested reversal schedule **S** can be modified without increasing the evaluation cost $\mathbf{T}(\mathbf{S})$, so that thin checkpoints can be stored only behind the corresponding fat checkpoint and during performing each final step only fat checkpoints are stored.

□

Theorem 4.1 and Theorem 4.2 imply the following powerful conclusion for any multiple sweep evolution $\mathcal{E}_{3\times l}$: a nested reversal schedule **S** for the implementation of a multiple sweep evolution $\mathcal{E}_{3\times l}$, using up to C fat checkpoints, can be decomposed into subschedules regarding each original sweep. After restoring an original state from the additional thin checkpoint, the original sweep to the original state $x_{\check{l}}$ and the action W_1 are performed. Then, the sequence of original steps $F_{\check{l}+1}, \ldots, F_l$ is reversed, using the remaining thin checkpoints. After that, the action $\bar{A}$ causes the adjoint step $\bar{F}_{\check{l}+1}$ during the release of the thin checkpoint storing the original state $x_{\check{l}}$. Finally, the reversal of the sequence $F_1, \ldots, F_{\check{l}}$ is performed, using the remaining thin checkpoints. This situation is depicted in Figure 4.8. This decomposition is similar to the decomposition of serial reversal schedules (see Chapter 2). The main difference is that, during the performance of the adjoint steps

$\bar{F}_{\check{l}+1}, \ldots, \bar{F}_l$, some adjoint states can be stored into fat checkpoints. Therefore, during the reversal of the sequence $F_1, \ldots, F_{\check{l}}$, the initial number of cC available thin checkpoints may not be provided any more.

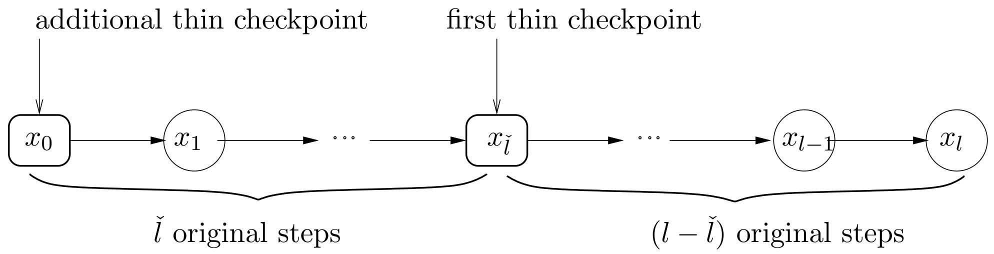

Figure 4.8: Decomposition of a multiple sweep evolution $\mathcal{E}_{3\times l}$ regarding the original sweep

Moreover, a nested reversal schedule $\mathbf{S}$ for the implementation of a multiple sweep evolution $\mathcal{E}_{3\times l}$, using up to C fat checkpoints, can be decomposed into subschedules regarding the adjoint sweep. After storing the adjoint state $\bar{x}_l$ into the additional fat checkpoint, the adjoint sweep to some state $\bar{x}_{\hat{l}}$ and the action $\bar{W}_1$ are performed. 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 action D causes the final step $\bar{\bar{F}}_{\hat{l}}$ during the release of the fat checkpoint storing the adjoint state $\bar{x}_{\hat{l}}$. Finally, the reversal of the sequence $\bar{F}_{\hat{l}+1}, \ldots, \bar{F}_l$ is performed, using C fat checkpoints again. This situation is depicted in Figure 4.9. This decomposition is similar to the decomposition of serial reversal schedules (see Chapter 2). The difference is that in order to perform each adjoint step $\bar{F}_i, 1 \leq i \leq l$, we need initially to perform original steps. Moreover, during the original sweep, some original states can be stored into thin checkpoints. Therefore, each time we start from the most convenient thin checkpoint. Thus, a number of original steps, which have to be performed in order to evaluate a corresponding adjoint step, each time varies whenever a corresponding adjoint step is performed. This is because the thin checkpoint distribution can be changed with each restarting of the original sweep.

4.2.3 Optimal Nested Reversal Schedules

An optimal nested reversal schedule $\mathbf{S}_{min}(l, C)$ for a given evolution $\mathcal{E}_{3\times l}$, step cost distribution $\vec{\mathbf{t}}_{\mathbf{3}\times\mathbf{l}}$, size distribution $\vec{\mathbf{d}}_{\mathbf{3}\times\mathbf{l}}$, and C available fat checkpoints, for which Theorem 4.1 and Theorem 4.2 are satisfied, can be established by

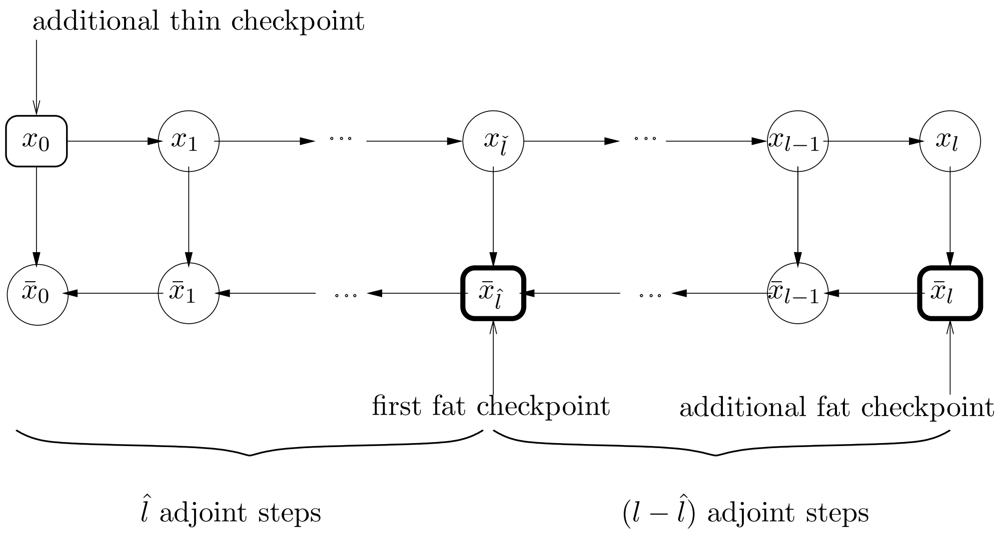

Figure 4.9: Decomposition of a multiple sweep evolution $\mathcal{E}_{3\times l}$ regarding the adjoint sweep

minimising the temporal complexity $\mathbf{T}_{min}(l, C)$ being defined by

$$\begin{aligned}\mathbf{T}_{min}(l,C) &= \min_{\substack{\check{l}_1<\ldots<\check{l}_{cC}\\ \hat{l}_C<\ldots<\hat{l}_1}} \Big\{\mathbf{T}_f(l,cC)t + (l-1)\bar{t} + \bar{\bar{t}} + \\ &+ \sum_{i=1}^{C} \mathbf{T}^1_{min}(\hat{l}_{i-1}-\hat{l}_i, C-i+1) + \mathbf{T}^1_{min}(\hat{l}_C - 1, 0)\Big\},\end{aligned} \tag{4.36}$$

$$\begin{aligned}\mathbf{T}^1_{min}(l,C) &= \min_{\substack{\check{l}_1<\ldots<\check{l}_{cC}\\ \hat{l}_C<\ldots<\hat{l}_1}} \Big\{\mathbf{T}^1_f(l,cC)t + (l-1)\bar{t} + \bar{\bar{t}} + \\ &+ \sum_{i=1}^{C} \mathbf{T}^1_{min}(\hat{l}_{i-1}-\hat{l}_i, C-i+1) + \mathbf{T}^1_{min}(\hat{l}_C - 1, 0)\Big\},\end{aligned} \tag{4.37}$$

$$\mathbf{T}_{min}(l,0) = lt + \sum_{i=1}^{l} \{\mathbf{T}_{bin}(i,1)t + (i-1)\bar{t}\} + l\bar{\bar{t}}, \tag{4.38}$$

$$\mathbf{T}^1_{min}(l,0) = \sum_{i=1}^{l} \{\mathbf{T}_{bin}(i,1)t + (i-1)\bar{t}\} + l\bar{\bar{t}}, \tag{4.39}$$

$$\mathbf{T}_{min}(l,C) = lt + (l-1)\bar{t} + l\bar{\bar{t}} \quad \text{if} \quad (l \le C+1), \tag{4.40}$$

$$\text{(4.41)} \qquad \mathbf{T}^1_{min}(l,C) \;=\; (l-1)t+(l-1)\bar{t}+l\bar{\bar{t}} \quad \text{if} \quad (l \le C+1),$$

with $\check{l}_1,\ldots,\check{l}_{cC},\hat{l}_1,\ldots,\hat{l}_C$ in (4.36) - (4.41) being chosen in the following way

$$\text{(4.42)} \qquad \begin{array}{c} 0=\check{l}_0<\check{l}_1<\ldots<\check{l}_{cC}<l, \\ \hat{l}_0=l, \\ \hat{l}_1<\check{l}_{c(C-1)+1}+1, \\ \vdots \\ \hat{l}_i<\min\{\hat{l}_{i-1},\check{l}_{c(C-i)+1}+1\}, \\ \vdots \\ \hat{l}_C<\min\{\hat{l}_{C-1},\check{l}_1+1\}. \end{array}$$

$\mathbf{T}_f(l,cC)$ and $\mathbf{T}^1_f(l,cC)$ in (4.36) and (4.37), respectively, denote the minimal number of original steps, leading up to the first final step $\bar{\bar{F}}_1$. Here, a sequence of l original steps is given, so that up to cC thin checkpoints can be accommodated at any time. Note, thin and fat checkpoints have a corresponding distribution, which is to be established by minimising the overall sum $\mathbf{T}_f(l,cC)$ and $\mathbf{T}^1_f(l,cC)$. To achieve the minimal numbers $\mathbf{T}_f(l,cC)$ and $\mathbf{T}^1_f(l,cC)$, binomial reversal schedules from Section 2 can be used.

The evaluation costs $\mathbf{T}_{min}(l,C)$ and $\mathbf{T}_f(l,cC)$ correspond to the case in which the last adjoint state $\bar{x}_l$ is not initialised and, consequently, its data is not stored into the additional fat checkpoint. Therefore, the original step F_l is to perform in order to initialise the adjoint state $\bar{x}_l$. The evaluation costs $\mathbf{T}^1_{min}(l,C)$ and $\mathbf{T}^1_f(l,cC)$ correspond to the case in which the adjoint state $\bar{x}_l$ is already initialised and its data is stored into the additional fat checkpoint. Therefore, there is no need to perform the last original step F_l.

Fat checkpoints can be stored as soon as required memory is available, i.e. as soon as at least c thin checkpoints are removed. Assume, there is a corresponding distribution for thin checkpoints

$$0=\check{l}_0<\check{l}_1<\ldots<\check{l}_{cC}<l,$$

and the additional fat checkpoint is stored ($\hat{l}_0=l$). The next fat checkpoint can be stored only if we remove at least c thin checkpoints. Thus, $0<\hat{l}_1<\check{l}_{c(C-1)+1}+1$. Note, it is not necessary to store this fat checkpoint immediately after c thin checkpoints are removed. If it is more convenient, fat checkpoints may be stored later. Moreover, the next fat checkpoint can be stored only if the previous one has already been stored, i.e. $0<\hat{l}_i<\hat{l}_{i-1},\ i=1,\ldots,C$.

Therefore, summarising, we obtain the following relation for $\hat{l}_i$: $0 < \hat{l}_i < \min\left\{\hat{l}_{i-1}, \check{l}_{c(C-i)+1} + 1\right\}, \; i = 1, \ldots, C$.

According to (4.36) - (4.41), thin and fat checkpoints are to be placed most preferably in order to achieve the minimal evaluation cost $\mathbf{T}_{min}(l, C)$, i.e. to construct an optimal nested reversal schedule $\mathbf{S}_{min}(l, C)$.

An optimal nested reversal schedule $\mathbf{S}_{min}(l, C)$ for a given evolution $\mathcal{E}_{3\times l}$, step cost distribution $\vec{\mathbf{t}}_{\mathbf{3}\times\mathbf{l}}$, size distribution $\vec{\mathbf{d}}_{\mathbf{3}\times\mathbf{l}}$, and C available fat checkpoints can be established using Algorithm 4.1. This algorithm constructs an optimal nested reversal schedule $\mathbf{S}_{min}(l, C)$, using exhaustive search techniques. The general steps of Algorithm 4.1 are presented and discussed below. The following notations are to be used.

i	counter of currently stored thin checkpoints,
$\bar{i}$	counter of currently stored fat checkpoints,
c_{av}	counter of currently available thin checkpoints.

Algorithm 4.1 works recursively. This begins with the first call of the procedure **SCHEMA** (l, C, $\mathbf{T}$, $\check{l}$, $\hat{l}$) with the corresponding parameters initialised before. In the procedure **THIN** (l, c_{av}, C, i, $\mathbf{T}$, $\check{l}$, $\hat{l}$) of Algorithm 4.1 thin checkpoints are to be set. In the procedure **FAT** (l, c_{av}, C, i, $\bar{i}$, $\mathbf{T}$, $\check{l}$, $\hat{l}$) fat checkpoints are to be set. In the statement **if** ($\mathbf{T} \leq \mathbf{T}_{min}$) of Algorithm 4.1 an appropriate nested reversal schedule is to be chosen, so that its evaluation cost $\mathbf{T}$ is reduced compared to the cost $\mathbf{T}_{min}$ of a reversal schedule in the previous cycle of the algorithm. The procedure **initialisation** (c_{av}, i, $\bar{i}$, $\check{l}$, $\hat{l}$) sets values of $\check{l}_k$, $\hat{l}_i \leq \check{l}_k \leq \hat{l}_{i-1}$, to $\check{l}_k = -1$. This corresponds to the removal of a certain number of thin checkpoints. Moreover, appropriate values of c_{av} and i are to be defined in this procedure. The function $\mathbf{T}_f(\hat{l}_{\bar{i}-1} - \hat{l}_{\bar{i}}, c_{av})$ in the procedure **FAT** (l, c_{av}, C, i, $\bar{i}$, $\mathbf{T}$, $\check{l}$, $\hat{l}$) of Algorithm 4.1 gives the number of original steps, necessary to perform in order to reverse the sequence of $(\hat{l}_{\bar{i}-1} - \hat{l}_{\bar{i}})$ original steps using c_{av} checkpoints. Moreover, the distribution of thin checkpoints between $\hat{l}_{\bar{i}}$ and $\hat{l}_{\bar{i}-1}$ has to be taken into account.

Implementing Algorithm 4.1, variables evaluated throughout different steps of this algorithm have to be stored locally, so that previous values of corresponding variables can be restored if required. Algorithm 4.1 terminates as soon as all opportunities for setting thin and fat checkpoints are exhausted. The best choice of these settings is to be selected. Obviously, the implementation of Algorithm 4.1 is very extensive in terms of run-time and memory complexity. For the implementation of Algorithm 4.1 there is a software available **optimal**, which is coded using the C++ programming language environment. Among various evaluations- and run-time tests, this is utilised

Algorithm 4.1 Exhaustive Search

Input: $\mathcal{E}_{3\times l}$, C, $\vec{\mathbf{t}}_{\mathbf{3}\times\mathbf{1}}$, $\vec{\mathbf{d}}_{\mathbf{3}\times\mathbf{1}}$, $\mathbf{T}=0$,
$\quad(\check{l},\,\hat{l}) : \check{l}_0 = \cdots = \check{l}_{cC} = \hat{l}_0 = \cdots = \hat{l}_C = -1$

Output: $\mathbf{S}_{min}(l,C)$, $\mathbf{T}_{min}(l,C)$

SCHEMA $(l,\, C,\, \mathbf{T},\, \check{l},\, \hat{l})$
- **if** $l \le C+1$ & $\hat{l}_0 = -1$
  - $\mathbf{T} = lt + (l-1)\bar{t} + l\bar{\bar{t}}$, **return**
- **if** $l \le C+1$ & $\hat{l}_0 = l$
  - $\mathbf{T} = (l-1)t + (l-1)\bar{t} + l\bar{\bar{t}}$, **return**
- **if** $C = 0$ & $\hat{l}_0 = -1$
  - $\mathbf{T} = lt + \sum_{i=1}^{l}\{\mathbf{T}_{bin}(i,1)t + (i-1)\bar{t}\} + l\bar{\bar{t}}$, **return**
- **if** $C = 0$ & $\hat{l}_0 = l$
  - $\mathbf{T} = \sum_{i=1}^{l}\{\mathbf{T}_{bin}(i,1)t + (i-1)\bar{t}\} + l\bar{\bar{t}}$, **return**
- $i=0$, $\check{l}_0 = 0$, $c_{av} = cC$, **THIN** $(l,\, c_{av},\, C,\, 0,\, \mathbf{T},\, \check{l},\, \hat{l})$, **return**

THIN $(l,\, c_{av},\, C,\, i,\, \mathbf{T},\, \check{l},\, \hat{l})$
- $\mathbf{T}_{min} = MAXREAL$
- **if** $(c_{av} > 0$ & $\check{l}_i + 1 < l)$
  - **for** $\check{l}_{i+1} = \check{l}_i + 1, \ldots, l-1$ **do**
    - $\mathbf{T} = \mathbf{T} + (\check{l}_{i+1} - \check{l}_i)$, **THIN** $(l,\, c_{av}-1,\, i+1,\, \mathbf{T},\, \check{l},\, \hat{l})$
    - **if** $(\mathbf{T} \le \mathbf{T}_{min})$
      - $\mathbf{T}_{min} = \mathbf{T}$
  - $\mathbf{T} = \mathbf{T}_{min}$, **return**
- **else**
  - **if** $\hat{l}_0 = l$
    - $\mathbf{T} = \mathbf{T} + (l - \check{l}_i - 1)$,
  - **else**
    - $\mathbf{T} = \mathbf{T} + (l - \check{l}_i)$,
  - $\bar{i} = 0$, $\hat{l}_0 = l$, **FAT** $(l,\, c_{av},\, C,\, i,\, \bar{i},\, \mathbf{T},\, \check{l},\, \hat{l})$, **return**

FAT $(l,\, c_{av},\, C,\, i,\, \bar{i},\, \mathbf{T},\, \check{l},\, \hat{l})$
- $\mathbf{T}_{min} = MAXREAL$
- **if** $(\bar{j} < C$ & $\hat{l}_{\bar{i}} - 1 > \check{l}_0)$
  - **for** $\hat{l}_{\bar{i}+1} = \min\{\hat{l}_{\bar{i}} - 1;\ \check{l}_{c(C-(\bar{i}+1))+1+c_{av}}\}, \ldots, \check{l}_0 + 1$ **do**
    - $\bar{i} = \bar{i} + 1$
    - $\mathbf{T} = \mathbf{T} + (\hat{l}_{\bar{i}-1} - \hat{l}_{\bar{i}})\bar{t} + \mathbf{T}_f(\hat{l}_{\bar{i}-1} - \hat{l}_{\bar{i}}, c_{av})t$
    - **initialisation** $(c_{av},\, i,\, \bar{i},\, \check{l},\, \hat{l})$
    - **FAT** $(l,\, c_{av} - c,\, C,\, i,\, \bar{i},\, \mathbf{T},\, \check{l},\, \hat{l})$
    - **if** $(\mathbf{T} \le \mathbf{T}_{min})$
      - $\mathbf{T}_{min} = \mathbf{T}$
  - $\mathbf{T} = \mathbf{T}_{min}$, **return**
- **else**
  - **for** $k = 0, \ldots, \bar{i} - 1$ **do**
    - $\mathbf{T_1} = 0$, **SCHEMA** $(\hat{l}_k - \hat{l}_{k+1},\, C-k,\, \mathbf{T_1},\, \check{l},\, \hat{l})$
    - $\mathbf{T} = \mathbf{T} + \mathbf{T_1}$
  - $\mathbf{T_1} = 0$, **SCHEMA** $(\hat{l}_{\bar{i}},\, C - \bar{i},\, \mathbf{T_1},\, \check{l},\, \hat{l})$
  - $\mathbf{T} = \mathbf{T} + \mathbf{T_1}$, **return**

to establish the next example.

Figure 4.10 shows one example for an optimal nested reversal schedule $\mathbf{S}_{min}(9,2)$ 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 $\vec{\mathbf{d}}_{\mathbf{3}\times\mathbf{9}} = (1,1,1)^\top$. $\mathcal{E}_{3\times 9}$ 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 additional fat checkpoint. Moreover, one thin checkpoint can be stored in place of a single fat one. Furthermore, the initial original state x_0 is stored into the additional thin checkpoint. In Figure 4.10 time steps

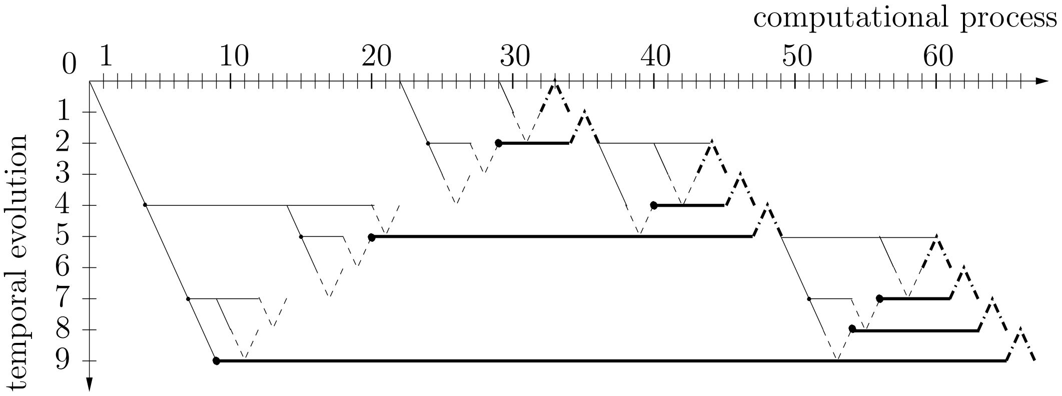

Figure 4.10: An optimal nested reversal schedules $\mathbf{S}_{min}(9,2)$ constructed using Algorithm 4.1

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 original steps $F_i,\ i = 1,\ldots,9$, whereas adjoint steps $\bar{F}_i,\ i = 1,\ldots,9$, are visualised by dotted slanted lines. Final steps $\bar{\bar{F}}_i,\ i = 1,\ldots,9$, are drawn by slanted dashed-dotted thick lines.

Nested reversal schedule $\mathbf{S}_{min}(9,2)$ starts with the action R_0 restoring the initial original state x_0 from the 0th thin checkpoint. Four actions A are executed by performing four original steps F_1, F_2, F_3, and F_4, consecutively. The original state x_4 is stored into the first thin checkpoint by the action W_1. Now, again, three actions A are applied to perform three original steps F_5, F_6, and F_7, and the original state x_7 is stored into the second thin checkpoint by the action W_2. Furthermore, the original steps F_8 and F_9 are executed by two actions A and the adjoint state $\bar{x}_9$ is initialised. The adjoint state $\bar{x}_9$

is stored into the additional fat checkpoint by the action $\bar{W}_0$. The adjoint sweep is started by this action.

Furthermore, data of the original state x_7 is restored from the second thin checkpoint by the action R_2, the original step F_8 is performed, and the adjoint states $\bar{x}_8$ and $\bar{x}_7$ can be evaluated by the application of the two actions $\bar{A}$. Then, the original state x_4 is restored from the first thin checkpoint by the action R_1, the original state x_5 is reevaluated by the action A, and its data is stored into the second thin checkpoint by the action W_2. After that, the original state x_6 is evaluated by the action A. Then, the adjoint states $\bar{x}_6$ and $\bar{x}_5$ are evaluated by the application of the action $\bar{A}$ two times. The adjoint state $\bar{x}_5$ is stored into the first fat checkpoint by the action $\bar{W}_1$. Then, we go back to the adjoint state $\bar{x}_2$ by reevaluating required original states and storing the adjoint state $\bar{x}_2$ into the second fat checkpoint by the action $\bar{W}_2$. Furthermore, 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. Furthermore, we store the current initial original state x_1 into the additional 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 optimal nested reversal schedule $\mathbf{S}_{min}(9,2)$ in Figure 4.10, it is necessary to perform 23 original steps F_i, 13 adjoint steps $\bar{F}_i$, and nine final steps $\bar{\bar{F}}_i$. The optimal nested reversal schedule $\mathbf{S}_{min}(9,2)$ and its minimal evaluation cost $\mathbf{T}_{min}(9,2)$ are determined using Algorithm 4.1 and routine **optimal**.

4.2.4 Heuristic

According to Section 4.2.3, 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 $\mathbf{S}_h(l,C)$ (h means a heuristic), so that its evaluation cost $\mathbf{T}_h(l,C)$ varies not too much from the minimal evaluation cost. Moreover, the construction of a nested reversal schedule $\mathbf{S}_h(l,C)$ can 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 $\mathbf{S}_h(l,C)$, is developed in the present section.

Consider an evolution $\mathcal{E}_{3\times l}$ traversing l time steps in three alternative sweeps. $\vec{\mathbf{t}}_{\mathbf{3}\times\mathbf{l}} = (t, \bar{t}, \bar{\bar{t}})^\top$ and $\vec{\mathbf{d}}_{\mathbf{3}\times\mathbf{l}} = (d, \bar{d}, \bar{\bar{d}})^\top$ are the 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 $\bar{x}_i$, $0 \leq i \leq l$. Assume that c thin checkpoints can be stored in place of a single fat one, i.e. $\bar{d} = c\, d$. By Algorithm 4.1 in Section 4.2.3 an optimal nested reversal schedule can be constructed using an exhaustive search algorithm. Using this approach, 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. In contrast to the situations for simple reversals involving only an original and an adjoint sweep, it was not possible to find a closed form characterisation of optimal reversal schedules. Therefore, a heuristic for the construction of appropriate nested reversal schedules was developed. The intent of this heuristic is to slightly restrict the placements of thin checkpoints. Due to the assumption (4.26), it is more convenient to reduce the movement freedom of thin checkpoints. This is because even a considerable increment in the number of evaluated original 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 (4.26) and (4.28)). 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 $\mathbf{S}_h(l, C)$ can be recursively constructed by minimising the evaluation cost

$$\begin{aligned} \mathbf{T}_h(l, C) &= \min_{0<\hat{l}<l} \Big\{ \mathbf{T}_{bin}(l - \hat{l} + 1, c)t + (l - \hat{l})\bar{t} + \\ &+ \mathbf{T}_h^1(l - \hat{l}, C) + \mathbf{T}_h(\hat{l}, , C - 1) \Big\}, \end{aligned} \tag{4.43}$$

with

$$\begin{aligned} \mathbf{T}_h^1(l, C) &= \min_{0<\hat{l}<l} \Big\{ \mathbf{T}_{bin}(l - \hat{l}, c)t + (l - \hat{l})\bar{t} + \\ &+ \mathbf{T}_h^1(l - \hat{l}, C) + \mathbf{T}_h(\hat{l}, C - 1) \Big\}, \end{aligned} \tag{4.44}$$

$$\mathbf{T}_h(l, 0) = lt + \sum_{i=1}^{l} \{\mathbf{T}_{bin}(i, 1)t + (i-1)\bar{t}\} + l\bar{\bar{t}}, \tag{4.45}$$

$$\mathbf{T}_h^1(l, 0) = \sum_{i=1}^{l} \{\mathbf{T}_{bin}(i, 1)t + (i-1)\bar{t}\} + l\bar{\bar{t}}, \tag{4.46}$$

$$(4.47) \qquad \mathbf{T}_h(l,C) \;=\; lt + (l-1)\bar{t} + l\bar{\bar{t}} \quad \text{if} \quad (l \le C+1),$$

and

$$(4.48) \qquad \mathbf{T}_h^1(l,C) \;=\; (l-1)t + (l-1)\bar{t} + l\bar{\bar{t}} \quad \text{if} \quad (l \le C+1).$$

Obviously, formulae (4.45) - (4.48) for $\mathbf{T}_h(l,0)$, $\mathbf{T}_h^1(l,0)$, $\mathbf{T}_h(l,C)$, and $\mathbf{T}_h^1(l,C)$ with $l \le C+1$ are equivalent to (4.38) - (4.41) for $\mathbf{T}_{min}(l,0)$, $\mathbf{T}_{min}^1(l,0)$, $\mathbf{T}_{min}(l,C)$, and $\mathbf{T}_{min}^1(l,C)$, respectively. The construction of a nested reversal schedule $\mathbf{S}_h(l,C)$, according to (4.43) - (4.48), leads to the following powerful conclusion: a nested reversal schedule $\mathbf{S}_h(l,C)$ for the implementation of a multiple sweep evolution $\mathcal{E}_{3\times l}$, using up to C fat checkpoints, can be decomposed into subschedules $\mathbf{S}_h(l-\hat{l},C)$ and $\mathbf{S}_h(\hat{l},C-1)$. $\mathbf{S}_h(l-\hat{l},C)$ and $\mathbf{S}_h(\hat{l},C-1)$ 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 additional fat checkpoint, the adjoint sweep to the adjoint state $\bar{x}_{\hat{l}}$ and the action $\bar{W}_1$ are performed. 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 action D causes the final step $\bar{\bar{F}}_{\hat{l}}$ during 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. Therefore, a nested reversal schedule can be decomposed into two nested subschedules and one simple reversal schedule as shown in Figure 4.11. Here, $\mathbf{S}_h(l,C)$ denotes a corre-

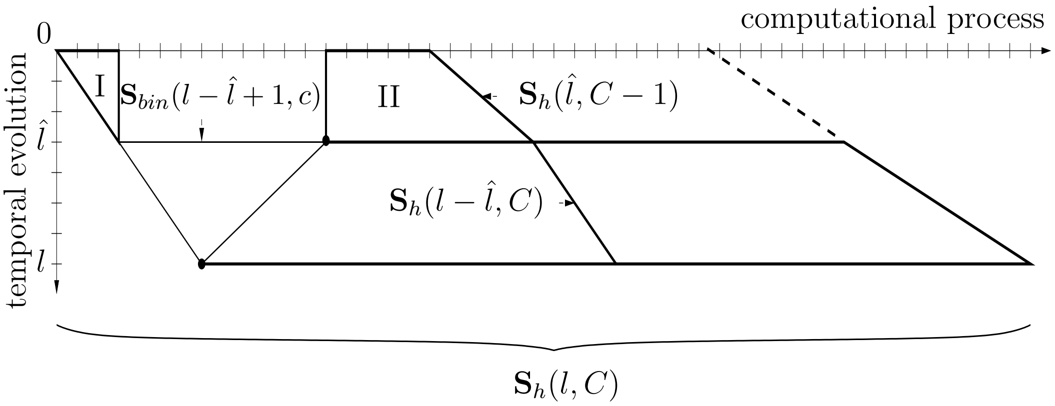

Figure 4.11: Decomposition of a nested reversal schedule $\mathbf{S}_h(l,C)$

sponding nested reversal schedule, which is constructed using the heuristic described above. This schedule is decomposed into two parts $\mathbf{S}_h(\hat{l},C-1)$ and $\mathbf{S}_h(l-\hat{l},C)$ as shown in Fig. 4.11. This happens 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 is applied to the reversal of $(l - \hat{l} + 1)$ original steps, using minimal run-time and memory requirement.

Since the execution of the nested reversal schedule $\mathbf{S}_h(\hat{l}, C-1)$ is interrupted by the execution of the binomial reversal schedule $\mathbf{S}_{bin}(l-\hat{l}+1, c)$, the nested reversal schedule $\mathbf{S}_h(\hat{l}, C-1)$ consists of two parts, that are denoted I and II, as it is shown in Figure 4.11. Therefore, the first part I of the nested reversal schedule $\mathbf{S}_h(\hat{l}, C-1)$ is to be evaluate firstly. Then, the binomial reversal schedule $\mathbf{S}_{bin}(l-\hat{l}+1, c)$ is performed. After that, the execution of the nested reversal schedule $\mathbf{S}_h(\hat{l}, C-1)$ is continued, i.e. the part II is performed. As it is shown in Figure 4.11, the part I of the nested reversal schedule $\mathbf{S}_h(\hat{l}, C-1)$ consists of the actions regarding original steps $F_1, \ldots, F_{\hat{l}}$, and actions regarding a subset of the original states $x_1, \ldots, x_{\hat{l}-1}$, that are to be stored into $c(C-1)$ thin checkpoints. The part II of the nested reversal schedule $\mathbf{S}_h(\hat{l}, C-1)$ consists of the actions regarding original steps $F_1, \ldots, F_{\hat{l}}$, original states $x_1, \ldots, x_{\hat{l}-1}$, adjoint steps $\bar{F}_1, \ldots, \bar{F}_{\hat{l}}$, adjoint states $\bar{x}_1, \ldots, \bar{x}_{\hat{l}-1}$, and action regarding final steps $\bar{\bar{F}}_1, \ldots, \bar{\bar{F}}_{\hat{l}}$, and final states $\bar{\bar{x}}_1, \ldots, \bar{\bar{x}}_{\hat{l}-1}$.

Consider the slopes of the slanted lines in the subschedules $\mathbf{S}_h(\hat{l}, C-1)$ and $\mathbf{S}_h(l-\hat{l}, C)$. The slopes of the left slanted lines in the part I of $\mathbf{S}_h(\hat{l}, C-1)$ and in the $\mathbf{S}_h(l-\hat{l}, C)$ are exactly one. This is because these lines represent the evaluation of the original steps $F_1, \ldots, F_{\hat{l}}$ and $F_{\hat{l}+1}, \ldots, F_l$, respectively, only once. The slopes of the right slanted lines in the part II of the nested reversal schedule $\mathbf{S}_h(\hat{l}, C-1)$ and in the $\mathbf{S}_h(l-\hat{l}, C)$ are greater than one. This is because these lines represent the evaluation of the final steps $\bar{\bar{F}}_1, \ldots, \bar{\bar{F}}_{\hat{l}}$ and $\bar{\bar{F}}_{\hat{l}+1}, \ldots, \bar{\bar{F}}_l$, respectively. Obviously, to evaluate a single final step, one has to perform generally a significant number of original and adjoint steps. Therefore, the slopes of corresponding slanted lines is greater than one.

According to the decomposition in Figure 4.11, a distribution of thin checkpoints depends on a distribution of fat checkpoints. Thus, if the adjoint state $\bar{x}_{\hat{l}}$ is stored into the first fat checkpoint, then, during the original sweep up to $c(C-1)$ thin checkpoints can be accommodated between the original states $x_0, \ldots, x_{\hat{l}}$. The initial original state x_0 is stored into the additional thin checkpoint. Moreover, up to c thin checkpoints can be accommodated between the original states $x_{\hat{l}}, \ldots, x_{l-1}$. The original state $x_{\hat{l}}$ is stored into one of the c available thin checkpoints.

Therefore, during the original sweep, the sequence of l original steps is divided into $(C+1)$ parts. The situation is illustrated graphically in Figure 4.12. In Figure 4.12, up to c thin checkpoints are available for each of $(C+1)$ parts of a step sequence (except for the first one). The initial original state of each part is stored into a thin checkpoint. The distribution of thin checkpoints

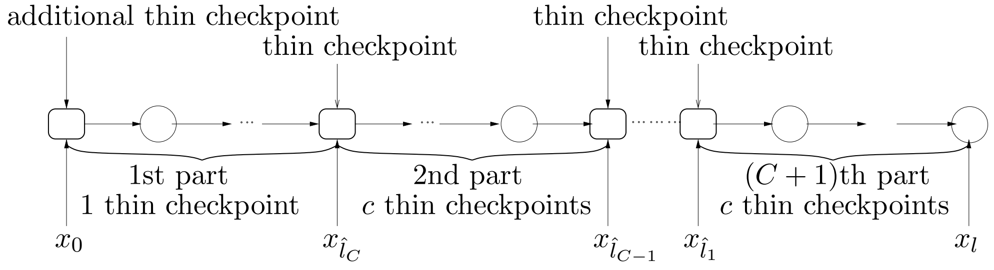

Figure 4.12: Partition of the original sweep of a multiple sweep evolution $\mathcal{E}_{3\times l}$ corresponding to setting fat checkpoints

within each part corresponds to the binomial distribution (see Section 2.2.1). From (4.43) - (4.48) it is obviously that the evaluation cost $\mathbf{T}_h(l, C)$ and, consequently, a corresponding nested reversal schedule $\mathbf{S}_h(l, C)$ can be evaluated using dynamic programming. The run-time and memory requirement, required for these evaluations using dynamic approach, 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 convexity properties described in the following. Thus, we have developed a so-called **Linear Method**. Basic properties of this method relies on information given in [59].

4.2.5 Linear Method

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

$$\begin{aligned} \mathbf{T}_h^{\hat{l}}(l, C) &= \mathbf{T}_{bin}(l - \hat{l} + 1, c + 1)t + (l - \hat{l})\bar{t} + \\ &+ \mathbf{T}_h^1(l - \hat{l}, C) + \mathbf{T}_h(\hat{l}, C - 1), \end{aligned} \tag{4.49}$$

$$\begin{aligned} \mathbf{T}_h^{1,\hat{l}}(l, C) &= \mathbf{T}_{bin}(l - \hat{l}, c + 1)t + (l - \hat{l})\bar{t} + \\ &+ \mathbf{T}_h^1(l - \hat{l}, C) + \mathbf{T}_h(\hat{l}, C - 1). \end{aligned} \tag{4.50}$$

$\mathbf{T}_h^{\hat{l}}(l, C)$ and $\mathbf{T}_h^{1,\hat{l}}(l, C)$ are functions of $\hat{l}$, so that l and C are fixed. First, we consider the function $\mathbf{T}_h^{\hat{l}}(l, C)$. The main property of this function, which provides the accuracy of Linear Method, is the convexity of $\mathbf{T}_h^{\hat{l}}(l, C)$ with respect to its parameter $\hat{l}$. This is due to the construction of $\mathbf{T}_h^{\hat{l}}(l, C)$ as a sum of convex functions according to (4.49). Therefore, each local minima of the function $\mathbf{T}_h^{\hat{l}}(l, C)$ is its global minima. The convexity property of the function $\mathbf{T}_h^{\hat{l}}(l, C)$ is exploited for the development of Linear Method. Obviously, the function $\mathbf{T}_h^{\hat{l}}(l, C)$ is constructed, so that

$$\mathbf{T}_h(l, C) = \min_{0<\hat{l}<l} \left\{ \mathbf{T}_h^{\hat{l}}(l, C) \right\}. \tag{4.51}$$

Figure 4.13 illustrates three examples for $\mathbf{T}_h^{\hat{l}}(100, C)$ as a function from $\hat{l}$ for $l = 100$, step cost distribution $\vec{\mathbf{t}}_{\mathbf{3}\times\mathbf{100}} = (1, 1, 1)^\top$, size distributions $\vec{\mathbf{d}}_{\mathbf{3}\times\mathbf{100}} = (1, 3, 1)^\top$, and different numbers C of fat checkpoints. In Figure 4.13 values

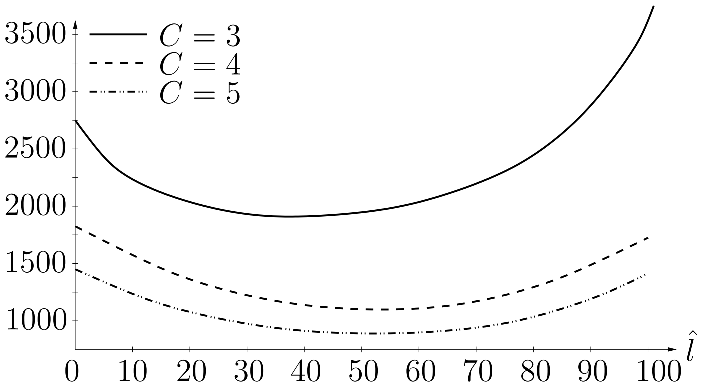

Figure 4.13: $\mathbf{T}_h^{\hat{l}}(100, C)$ as a function on $\hat{l}$

of the function $\mathbf{T}_h^{\hat{l}}(100, C)$ 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 4.13 it can be also seen that the function $\mathbf{T}_h^{\hat{l}}(100, C)$ is convex.

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

$$\mathbf{T}_h^{\hat{l}_\star}(l, C) = \mathbf{T}_h(l, C) = \min_{0<\hat{l}<l} \mathbf{T}_h^{\hat{l}}(l, C). \tag{4.52}$$

Note, for $\hat{l}$ satisfying $\hat{l} \in (0, \hat{l}_\star)$ the function $\mathbf{T}_h^{\hat{l}}(l, C)$ decreases. For $\hat{l}$ satisfying $\hat{l} \in (\hat{l}_\star, l)$ the function $\mathbf{T}_h^{\hat{l}}(l, C)$ increases. This monotonic property is also used to develop Linear Method.

In the continuous case in order to find extrema points of a convex function depending on a single variable, it is required to find zero points of the first derivative of this function. In the discrete case, defined by equation (4.52), evaluating an extrema point $\hat{l}_\star$ is equivalent to finding $\hat{l}_\star$, so that

$$\mathbf{T}_h^{\hat{l}}(l, C) \text{ decreases for } \hat{l} \in (0, \hat{l}_\star), \tag{4.53}$$

and

$$\mathbf{T}_h^{\hat{l}}(l,C) \text{ increases for } \hat{l} \in (\hat{l}_\star, l). \tag{4.54}$$

Therefore, we have to find an interval, containing a single or more points $\hat{l}_\star$, so that the function $\mathbf{T}_h^{\hat{l}}(l,C)$ neither increases nor decreases in this interval, i.e. it takes its minimal value. In the sequel, consider a difference defined by

$$\mathbf{T}_h^{\hat{l}+1}(l,C) - \mathbf{T}_h^{\hat{l}}(l,C). \tag{4.55}$$

For $\hat{l}$ satisfying $\hat{l} \in (0, \hat{l}_\star)$, the difference (4.55) is negative. For $\hat{l}$ satisfying $\hat{l} \in \left[\hat{l}_\star, l\right)$, the difference (4.55) is positive. For the interval $\left[\hat{l}_\star^l, \hat{l}_\star^u\right)$ of optimal values $\hat{l}_\star$ the difference (4.55) is zero. If the interval of optimal values contains only one value, $\hat{l}_\star$ can be defined, so that $\left(\mathbf{T}_h^{\hat{l}_\star}(l,C) - \mathbf{T}_h^{\hat{l}_\star-1}(l,C)\right)$ is negative and $\left(\mathbf{T}_h^{\hat{l}_\star+1}(l,C) - \mathbf{T}_h^{\hat{l}_\star}(l,C)\right)$ is positive. Obviously, considerations above concerning the function $\mathbf{T}_h^{\hat{l}}(l,C)$ can be straightforwardly transformed to the function $\mathbf{T}_h^{1,\hat{l}}(l,C)$. This is because the functions $\mathbf{T}_h^{\hat{l}}(l,C)$ and $\mathbf{T}_h^{1,\hat{l}}(l,C)$ differ from each over only in the argument for the function $\mathbf{T}_{min}$ (recall (4.49) and (4.50)).

To establish a search algorithm for the construction of admissible nested reversal schedules $\mathbf{S}_h(l,C)$, the optimal point $\hat{l}_\star$ can be evaluated recursively. To do this, it is necessary to prove the following two assertions.

Lemma 4.1 *Consider an evolution $\mathcal{E}_{3\times l}$ traversing l time steps in three alternative sweeps. $\vec{\mathbf{t}}_{\mathbf{3}\times(\mathbf{l-1})} = \vec{\mathbf{t}}_{\mathbf{3}\times\mathbf{l}} = \vec{\mathbf{t}}_{\mathbf{3}\times(\mathbf{l+1})} = (t, \bar{t}, \bar{\bar{t}})^\top$ and $\vec{\mathbf{d}}_{\mathbf{3}\times(\mathbf{l-1})} = \vec{\mathbf{d}}_{\mathbf{3}\times\mathbf{l}} = \vec{\mathbf{d}}_{\mathbf{3}\times(\mathbf{l+1})} = (d, \bar{d}, \bar{\bar{d}})^\top$ 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*

$$\mathbf{T}_h(l+1,C) - 2\mathbf{T}_h(l,C) + \mathbf{T}_h(l-1,C) \ge 0, \tag{4.56}$$

$$\mathbf{T}_h^1(l+1,C) - 2\mathbf{T}_h^1(l,C) + \mathbf{T}_h^1(l-1,C) \ge 0. \tag{4.57}$$

Proof:

Left hand sides of inequalities (4.56) and (4.57) correspond to the second discrete derivatives of the functions $\mathbf{T}_h(\vec{\mathbf{t}}_{3(m)}, \vec{\mathbf{d}}_{3(m)}, C)$ and $\mathbf{T}_h^1(\vec{\mathbf{t}}_{3(m)}, \vec{\mathbf{d}}_{3(m)}, C)$, respectively, with respect to m at the point $m = l$.

From the definition of the function $\mathbf{T}_h(l, C)$, according to (4.47), for $l+1 \leq C+1$ we obtain

$$\mathbf{T}_h(l+1, C) - 2\mathbf{T}_h(l, C) + \mathbf{T}_h(l-1, C) = 0. \tag{4.58}$$

Similarly, for the function $\mathbf{T}_h^1(l, C)$, according to (4.48), we have

$$\mathbf{T}_h^1(l+1, C) - 2\mathbf{T}_h^1(l, C) + \mathbf{T}_h^1(l-1, C) = 0. \tag{4.59}$$

Moreover, for $C = 0$ from (4.43) we obtain

$$\begin{aligned} &\mathbf{T}_h(l+1, 0) - 2\mathbf{T}_h(l, 0) + \mathbf{T}_h(l-1, 0) = \\ &= (\mathbf{T}_{bin}(l+1, 1) - \mathbf{T}_{bin}(l, 1) - (\mathbf{T}_{bin}(l, 1) - \mathbf{T}_{bin}(l-1, 1)))\, t = \\ &= (r_{bin}(l+1, 1) - r_{bin}(l, 1))\, t \geq 0. \end{aligned} \tag{4.60}$$

Analog, for the function $\mathbf{T}_h^1(\vec{\mathbf{t}}_{3(1)}, \vec{\mathbf{d}}_{3(1)}, C)$ with $C = 0$ we obtain

$$\begin{aligned} &\mathbf{T}_h^1(l+1, 0) - 2\mathbf{T}_h^1(l, 0) + \mathbf{T}_h^1(l-1, 0) = 0 \\ &= (r_{bin}(l+1, 1) - r_{bin}(l, 1))\, t \geq 0. \end{aligned} \tag{4.61}$$

Therefore, for $l+1 \leq C+1$ and $C = 0$ the functions $\mathbf{T}_h(l, C)$ and $\mathbf{T}_h^1(l, C)$ are monotonic increasing and their second derivatives with respect to l, defined by (4.56) - (4.57), are non-negative.

Assume that this is also valid for all $\tilde{l}$ and a fixed number C, so that $\tilde{l} < l$ with l being arbitrary positive values. Then, according to (4.43) and (4.44), functions $\mathbf{T}_h(l, C)$ and $\mathbf{T}_h^1(l, C)$ are constructed as a minimum of the sum of functions with non-negative second derivatives (due to the assumption). Therefore, the second derivatives of the functions $\mathbf{T}_h(l, C)$ and $\mathbf{T}_h^1(l, C)$ are also non-negative. Particularly, we have to consider four possible cases. First, we carry out a detailed prove for the relation (4.56).

Case 1: $\hat{m}_\star = \hat{l}_\star$ and $\hat{l}_\star = \hat{n}_\star$ with $m = l+1$ and $n = l-1$. From (4.56) and (4.43) we obtain

$$\begin{aligned} &\mathbf{T}_h(l+1, C) - 2\mathbf{T}_h(l, C) + \mathbf{T}_h(l-1, C) = \\ &= \mathbf{T}_{bin}(l - \hat{l}_\star + 2, c)t + (l + 1 - \hat{l}_\star)\bar{t} + \mathbf{T}_h^1(l + 1 - \hat{l}_\star, C) + \mathbf{T}_h(\hat{l}_\star, C-1) - \\ &-2\mathbf{T}_{bin}(l - \hat{l}_\star + 1, c)t - 2(l - \hat{l}_\star)\bar{t} - 2\mathbf{T}_h^1(l - \hat{l}_\star, C) - 2\mathbf{T}_h(\hat{l}_\star, C-1) + \\ &+\mathbf{T}_{bin}(l - \hat{l}_\star, c)t + (l - 1 - \hat{l}_\star)\bar{t} + \mathbf{T}_h^1(l - 1 - \hat{l}_\star, C) + \mathbf{T}_h(\hat{l}_\star, C-1) = \\ &= r_{bin}(l - \hat{l}_\star + 2, c)t - r_{bin}(l - \hat{l}_\star + 1, c)t + \\ &+\mathbf{T}_h^1(l + 1 - \hat{l}_\star, C) - 2\mathbf{T}_h^1(l - \hat{l}_\star, C) + \mathbf{T}_h^1(l - 1 - \hat{l}_\star, C) \geq 0. \end{aligned} \tag{4.62}$$

Relation (4.62) is satisfied because the function $r_{bin}(l,c)$ is monotone increasing with respect to l and due to the induction assumption. The induction assumption can be used here, since $l - \hat{l}_\star < l$.

Case 2: $\hat{m}_\star = \hat{l}_\star + 1$ and $\hat{l}_\star - 1 = \hat{n}_\star$ with $m = l+1$ and $n = l-1$. From (4.56) and (4.43) we obtain

$$\begin{aligned}
&\mathbf{T}_h(l+1,C) - 2\mathbf{T}_h(l,C) + \mathbf{T}_h(l-1,C) = \\
&= \mathbf{T}_{bin}(l-\hat{l}_\star+1,c)t + (l-\hat{l}_\star)\bar{t} + \mathbf{T}_h^1(l-\hat{l}_\star,C) + \mathbf{T}_h(\hat{l}_\star+1,C-1) - \\
&-2\mathbf{T}_{bin}(l-\hat{l}_\star+1,c)t - 2(l-\hat{l}_\star)\bar{t} - 2\mathbf{T}_h^1(l-\hat{l}_\star,C) - 2\mathbf{T}_h(\hat{l}_\star,C-1) + \\
&+\mathbf{T}_{bin}(l-\hat{l}_\star+1,c)t + (l-\hat{l}_\star)\bar{t} + \mathbf{T}_h^1(l-\hat{l}_\star,C) + \mathbf{T}_h(\hat{l}_\star-1,C-1) = \\
&= \mathbf{T}_h(\hat{l}_\star+1,C-1) - 2\mathbf{T}_h(\hat{l}_\star,C-1) + \mathbf{T}_h(\hat{l}_\star-1,C-1) \geq 0.
\end{aligned} \tag{4.63}$$

Relation (4.63) is satisfied due to the induction assumption. The induction assumption can be used here, since $\hat{l}_\star < l$.

Case 3: $\hat{m}_\star = \hat{l}_\star + 1$ and $\hat{l}_\star = \hat{n}_\star$ with $m = l+1$ and $n = l-1$. From (4.56) and (4.43) we obtain

$$\begin{aligned}
&\mathbf{T}_h(l+1,C) - 2\mathbf{T}_h(l,C) + \mathbf{T}_h(l-1,C) = \\
&= \mathbf{T}_{bin}(l-\hat{l}_\star+1,c)t + (l-\hat{l}_\star)\bar{t} + \mathbf{T}_h^1(l-\hat{l}_\star,C) + \mathbf{T}_h(\hat{l}_\star+1,C-1) - \\
&-2\mathbf{T}_{bin}(l-\hat{l}_\star+1,c)t - 2(l-\hat{l}_\star)\bar{t} - 2\mathbf{T}_h^1(l-\hat{l}_\star,C) - 2\mathbf{T}_h(\hat{l}_\star,C-1) + \\
&+\mathbf{T}_{bin}(l-\hat{l}_\star,c)t + (l-1-\hat{l}_\star)\bar{t} + \mathbf{T}_h^1(l-1-\hat{l}_\star,C) + \mathbf{T}_h(\hat{l}_\star,C-1) = \\
&= -r_{bin}(l-\hat{l}_\star+1,c)t - \bar{t} + \mathbf{T}_h(\hat{l}_\star+1,C-1) - \mathbf{T}_h(\hat{l}_\star,C-1) - \\
&-\mathbf{T}_h^1(l-\hat{l}_\star,C) + \mathbf{T}_h^1(l-\hat{l}_\star-1,C) \geq 0.
\end{aligned} \tag{4.64}$$

Relation (4.64) is satisfied because, due to the assumption, $\hat{l}_\star$ is an optimal point for the function $\mathbf{T}_h(l,C)$. Therefore, using (4.49), we obtain

$$\begin{aligned}
&0 \leq \mathbf{T}_h^{\hat{l}_\star+1}(l,C) - \mathbf{T}_h^{\hat{l}_\star}(l,C) = \\
&= -r_{bin}(l-\hat{l}_\star+1,c)t - \bar{t} + \mathbf{T}_h(\hat{l}_\star+1,C-1) - \mathbf{T}_h(\hat{l}_\star,C-1) - \\
&-\mathbf{T}_h^1(l-\hat{l}_\star,C) + \mathbf{T}_h^1(l-\hat{l}_\star-1,C),
\end{aligned} \tag{4.65}$$

which implies (4.64).

Case 4: $\hat{m}_\star = \hat{l}_\star$ and $\hat{l}_\star - 1 = \hat{n}_\star$ with $m = l+1$ and $n = l-1$. From (4.56) and (4.43) we obtain

$$\begin{aligned}
&\mathbf{T}_h(l+1,C) - 2\mathbf{T}_h(l,C) + \mathbf{T}_h(l-1,C) = \\
&= \mathbf{T}_{bin}(l-\hat{l}_\star+2,c)t + (l+1-\hat{l}_\star)\bar{t} + \mathbf{T}_h^1(l+1-\hat{l}_\star,C) + \mathbf{T}_h(\hat{l}_\star,C-1) - \\
&-2\mathbf{T}_{bin}(l-\hat{l}_\star+1,c)t - 2(l-\hat{l}_\star)\bar{t} - 2\mathbf{T}_h^1(l-\hat{l}_\star,C) - 2\mathbf{T}_h(\hat{l}_\star,C-1) + \\
&+\mathbf{T}_{bin}(l-\hat{l}_\star+1,c)t + (l-\hat{l}_\star)\bar{t} + \mathbf{T}_h^1(l-\hat{l}_\star,C) + \mathbf{T}_h(\hat{l}_\star-1,C-1) = \\
&= r_{bin}(l-\hat{l}_\star+2,c)t + \bar{t} + \mathbf{T}_h^1(l-\hat{l}_\star+1,C) - \mathbf{T}_h^1(l-\hat{l}_\star,C) - \\
&-\mathbf{T}_h(\hat{l}_\star,C-1) + \mathbf{T}_h(\hat{l}_\star-1,C-1) \geq 0.
\end{aligned} \tag{4.66}$$

Relation (4.66) is satisfied because, due to the assumption, $\hat{l}_\star$ is an optimal point for the function $\mathbf{T}_h(l, C)$. Therefore, using (4.49), we obtain

$$\begin{aligned} & 0 \le \mathbf{T}_h^{\hat{l}_\star - 1}(l, C) - \mathbf{T}_h^{\hat{l}_\star}(l, C) = \\ & = r_{bin}(l - \hat{l}_\star + 2, c)t + \bar{t} + \mathbf{T}_h^1(l - \hat{l}_\star + 1, C) - \mathbf{T}_h^1(l - \hat{l}_\star, C) - \\ & - \mathbf{T}_h(\hat{l}_\star, C - 1) + \mathbf{T}_h(\hat{l}_\star - 1, C - 1), \end{aligned} \tag{4.67}$$

which implies (4.66).

Therefore, the relation (4.56) is proved. The second relation (4.57) can be shown in the same way, namely, considering four cases as above and using corresponding formulae for $\mathbf{T}_h^{1,\hat{l}}(l, C)$. The difference is merely in changing $r_{bin}(l, c)$ by $r_{bin}(l - 1, c)$ in each equation.

Thus, the assertion of Lemma 4.1 is proved. □

Lemma 4.2 *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 $\hat{l}_\star^1$ satisfying*

$$\mathbf{T}_h^{\hat{l}_\star}(l, C) \;=\; \min_{0<\hat{l}<l} \mathbf{T}_h^{\hat{l}}(l, C). \tag{4.68}$$

and

$$\mathbf{T}_h^{1,\hat{l}_\star^1}(l, C) \;=\; \min_{0<\hat{l}<l} \mathbf{T}_h^{1,\hat{l}}(l, C). \tag{4.69}$$

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 corresponding 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

$$\hat{m}_\star \in \left[\hat{l}_\star, \ldots, \hat{l}_\star + (m - l)\right]. \tag{4.70}$$

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

$$\hat{m}_\star^1 \in \left[\hat{l}_\star^1, \ldots, \hat{l}_\star^1 + (m - l)\right]. \tag{4.71}$$

Proof:

It is sufficient to prove the assertion of Lemma 4.2 for $m = l+1$. Then, the result can be extended for any arbitrary m. First, prove the property (4.70).

Consider first the trivial case $l+1 \leq C+1$. Obviously, the relation (4.70) is satisfied for this case. Furthermore, assume that the assertion of Lemma 4.2 is true for all $\tilde{l}$ with $\tilde{l} \leq l$. Now, it will be shown that the property (4.70) is valid for $(l+1)$.

Since $\hat{l}_\star$ satisfies (4.52), the following relations are also satisfied

$$\mathbf{T}_h^{\hat{l}_\star}(l,C) - \mathbf{T}_h^{\hat{l}_\star-1}(l,C) \leq 0, \qquad (\hat{l}_\star > 1), \tag{4.72}$$

and

$$\mathbf{T}_h^{\hat{l}_\star+1}(l,C) - \mathbf{T}_h^{\hat{l}_\star}(l,C) \geq 0, \qquad (\hat{l}_\star < l-1). \tag{4.73}$$

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

$$\mathbf{T}_h^{\hat{l}_\star}(l+1,C) - \mathbf{T}_h^{\hat{l}_\star-1}(l+1,C) \leq 0, \ (\Leftrightarrow \hat{m}_\star \geq \hat{l}_\star > 1), \tag{4.74}$$

and

$$\mathbf{T}_h^{\hat{l}_\star+2}(l+1,C) - \mathbf{T}_h^{\hat{l}_\star+1}(l+1,C) \geq 0, \ (\Leftrightarrow \hat{m}_\star \leq \hat{l}_\star + 1 < l). \tag{4.75}$$

Firstly, we point out that (4.74) is fulfilled. For this purpose, we evaluate the difference between (4.72) and (4.74). After that, we prove that this difference is non-negative. Therefore, because of relation (4.72), the inequality (4.74) is satisfied. Thus,

$$\begin{aligned} &\mathbf{T}_h^{\hat{l}_\star}(l,C) - \mathbf{T}_h^{\hat{l}_\star-1}(l,C) - \mathbf{T}_h^{\hat{l}_\star}(l+1,C) + \mathbf{T}_h^{\hat{l}_\star-1}(l+1,C) = \\ &= \Big\{\mathbf{T}_{bin}(l-\hat{l}_\star+1,c) - 2\mathbf{T}_{bin}(l-\hat{l}_\star+2,c) + \mathbf{T}_{bin}(l-\hat{l}_\star+3,c)\Big\}\, t + \\ &+\mathbf{T}_h^1(l-\hat{l}_\star+2,C) - 2\mathbf{T}_h^1(l-\hat{l}_\star+1,C) + \mathbf{T}_h^1(l-\hat{l}_\star,C) \geq 0. \end{aligned} \tag{4.76}$$

The inequality (4.76) is proved, using Lemma 4.1. This is because the assertion of Lemma 4.2 is satisfied for $l - \hat{l}_\star + 2 \leq l$ with $\hat{l}_\star > 1$ due to the induction assumption. Moreover,

$$\begin{aligned} &\mathbf{T}_{bin}(l-\hat{l}_\star+1,c) - 2\mathbf{T}_{bin}(l-\hat{l}_\star+2,c) + \mathbf{T}_{bin}(l-\hat{l}_\star+3,c) = \\ &= \mathbf{T}_{bin}(l-\hat{l}_\star+3,c) - \mathbf{T}_{bin}(l-\hat{l}_\star+2,c) - \\ &-(\mathbf{T}_{bin}(l-\hat{l}_\star+2,c) - \mathbf{T}_{bin}(l-\hat{l}_\star+1,c)) = \\ &= r_{bin}(l-\hat{l}_\star+3,c) - r_{bin}(l-\hat{l}_\star+2,c) \geq 0. \end{aligned} \tag{4.77}$$

Here, in (4.77) we use the monotonic property of binomial repetition numbers $r_{bin}(l, c)$ with respect to l (see Chapter 2). Therefore, (4.76) and, respectively, (4.74) are satisfied. Then, we point out that (4.75) is fulfilled. For this purpose, we evaluate the difference between (4.75) and (4.73). After that, we prove that this difference is non-negative. Therefore, because of relation (4.73), the inequality (4.75) is satisfied. Thus,

$$\begin{aligned}\mathbf{T}_h^{\hat{l}_\star+2}(l+1, C) - \mathbf{T}_h^{\hat{l}_\star+1}(l+1, C) - \mathbf{T}_h^{\hat{l}_\star+1}(l, C) + \mathbf{T}_h^{\hat{l}_\star}(l, C) = \\ = \mathbf{T}_h(\hat{l}_\star + 2, C-1) - 2\mathbf{T}_h(\hat{l}_\star + 1, C-1) + \mathbf{T}_h(\hat{l}_\star, C-1) \geq 0.\end{aligned} \tag{4.78}$$

The inequality (4.78) is proved, using Lemma 4.1. This is because the assertion of Lemma 4.2 is satisfied for $\hat{l}_\star + 2 \leq l$ with $\hat{l}_\star < l - 1$ due to the induction assumption. Therefore, for $m = l + 1$ we obtain: $\hat{m}_\star = \hat{l}_\star + 1$ or $\hat{m}_\star = \hat{l}_\star$. Thus, the property (4.70) 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 (4.70) is proved for any m.

Obviously, consideration above can be applied to the function $\mathbf{T}_h^1(l, C)$ also. They can be straightforwardly transformed by replacing function $\mathbf{T}_h(l, C)$ by function $\mathbf{T}_h^1(l, C)$, so that they are valid for the function $\mathbf{T}_h^1(l, C)$. Therefore, the property (4.71) is also proved. This concludes the proof of Lemma 4.2.

□

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

$$\mathbf{F}_h(l, C) = \mathbf{T}_h(l, C) - \mathbf{T}_h(l - 1, C), \tag{4.79}$$

and

$$\mathbf{F}_h^1(l, C) = \mathbf{T}_h^1(l, C) - \mathbf{T}_h^1(l - 1, C). \tag{4.80}$$

Here, the functions $\mathbf{T}_h(l, C)$ and $\mathbf{T}_h^1(l, C)$ are given by the formulae (4.43) - (4.48). 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 $\mathbf{F}_h(l, C)$ and $\mathbf{F}_h^1(l, C)$ can be evaluated recursively. Simultaneously, the appropriate nested reversal schedule $\mathbf{S}_h(l, C)$ is constructed. The temporal complexity of this nested reversal schedule, i.e. its evaluation cost is equal to $\mathbf{T}_h(l, C)$.

Procedure 4.1 Evaluation $\mathbf{F}_h(l+1,C)$ and $\mathbf{F}_h^1(l+1,C)$

a). Evaluation $\mathbf{F}_h(l+1,C)$.

Initial case: If $l+1 \le C+1$, $\mathbf{F}_h(l+1,C) = t + \bar{t} + \bar{\bar{t}}$.

$\mathbf{C = 0}$:

$$\begin{gathered}\mathbf{F}_h(l+1,0) = \mathbf{T}_h(l+1,0) - \mathbf{T}_h(l,0) = \\ = (l+1)t + \sum_{i=1}^{l+1} \{\mathbf{T}_{bin}(i,1)t + (i-1)\bar{t}\} + (l+1)\bar{\bar{t}} - \\ -lt - \sum_{i=1}^{l} \{\mathbf{T}_{bin}(i,1)t + (i-1)\bar{t}\} - l\bar{\bar{t}} = \frac{l(l+1)+2}{2}t + l\bar{t} + \bar{\bar{t}}.\end{gathered}$$

$\mathbf{C \ge 1}$: Assume, for a specified step cost distribution $\vec{\mathbf{t}}_{\mathbf{3\times 1}}$, size distribution $\vec{\mathbf{d}}_{\mathbf{3\times 1}}$, and C available fat checkpoints the optimal value $\hat{l}_\star$ satisfying $\mathbf{T}_h^{\hat{l}_\star}(l,C) = \min\limits_{0<\hat{l}<l} \mathbf{T}_h^{\hat{l}}(l,C)$ is determined. Then,

$$\begin{gathered}\mathbf{F}_h(l+1,C) = \mathbf{T}_h(l+1,C) - \mathbf{T}_h(l,C) = \mathbf{T}_h(l+1,C) - \\ -\mathbf{T}_{bin}(l-\hat{l}_\star+1,c)t - \mathbf{T}_h^1(l-\hat{l}_\star,C) - (l-\hat{l}_\star)\bar{t} - \mathbf{T}_h(\hat{l}_\star,C-1).\end{gathered}$$

To evaluate $\mathbf{F}_h(\vec{\mathbf{d}}_{\mathbf{3\times(l+1)}},C)$, apply Lemma 4.2 with $m = l+1$. Therefore, we have to consider two cases: $\hat{m}_\star = \hat{l}_\star + 1$ and $\hat{m}_\star = \hat{l}_\star$.

First case: $\hat{m}_\star = \hat{l}_\star + 1$ for $m = l+1$.

$$\begin{gathered}\mathbf{F}_h(l+1,C) = \mathbf{T}_{bin}(l-\hat{l}_\star+1,c)t + \mathbf{T}_h^1(l-\hat{l}_\star,C) + \\ +\mathbf{T}_h(\hat{l}_\star+1,C-1) + (l+1-\hat{l}_\star-1)\bar{t} - \\ -\mathbf{T}_{bin}(l-\hat{l}_\star+1,c)t - \mathbf{T}_h^1(l-\hat{l}_\star,C) - \mathbf{T}_h(\hat{l}_\star,C-1) - (l-\hat{l}_\star)\bar{t} = \\ = \mathbf{T}_h(\hat{l}_\star+1,C-1) - \mathbf{T}_h(\hat{l}_\star,C-1) = \mathbf{F}_h(\hat{l}_\star+1,C-1).\end{gathered}$$

Second case: $\hat{m}_\star = \hat{l}_\star$ for $m = l+1$.

$$\begin{gathered}\mathbf{F}_h(l+1,C) = \mathbf{T}_{bin}(l+1-\hat{l}_\star+1,c)t + \mathbf{T}_h^1(l+1-\hat{l}_\star,C) + \\ +\mathbf{T}_h(\hat{l}_\star,C-1) + (l+1-\hat{l}_\star)\bar{t} - \\ -\mathbf{T}_{bin}(l-\hat{l}_\star+1,c)t - \mathbf{T}_h^1(l-\hat{l}_\star,C) - \\ -\mathbf{T}_h(\hat{l}_\star,C-1) - (l-\hat{l}_\star)\bar{t} = \\ = \mathbf{T}_h^1(l+1-\hat{l}_\star,C) - \mathbf{T}_h^1(l-\hat{l}_\star,C) + \bar{t} + \\ +\mathbf{T}_{bin}(l+2-\hat{l}_\star,c)t - \mathbf{T}_{bin}(l-\hat{l}_\star+1,c)t = \\ = r(l-\hat{l}_\star+2,c)t + \mathbf{F}_h^1(l+1-\hat{l}_\star,C) + \bar{t} = \\ = r(l-\hat{l}_\star+2,c)t + \mathbf{F}_h^1(l-\hat{l}_\star+1,C) + \bar{t}.\end{gathered}$$

Recall the definition of the function $\mathbf{F}_h(l+1, C)$, according to (4.79). Since the function $\mathbf{T}_h(l+1, C)$ takes its minimum between $\mathbf{T}_h^{\hat{l}_\star}(l+1, C)$ and $\mathbf{T}_h^{\hat{l}_\star+1}(l+1, C)$, the function $\mathbf{F}_h(l+1, C)$ can be evaluated by

$$\mathbf{F}_h(l+1, C) = \min \Big\{ \mathbf{F}_h(\hat{l}_\star + 1, C-1);$$
$$r(l - \hat{l}_\star + 2, c)t + \mathbf{F}_h^1(l - \hat{l}_\star + 1, C) + \bar{t} \Big\} .$$

b). Evaluation $\mathbf{F}_h^1(l+1, C)$.

Initial case: If $l+1 \leq C+1$, $\mathbf{F}_h^1(l+1, C) = t + \bar{t} + \bar{\bar{t}}$.

$\mathbf{C = 0}$:

$$\mathbf{F}_h^1(l+1, 0) = \mathbf{T}_h^1(l+1, 0) - \mathbf{T}_h^1(l, 0) =$$
$$= \sum_{i=1}^{l+1} \{\mathbf{T}_{bin}(i, 1)t + (i-1)\bar{t}\} + (l+1)\bar{\bar{t}} -$$
$$- \sum_{i=1}^{l} \{\mathbf{T}_{bin}(i, 1)t + (i-1)\bar{t}\} - l\bar{\bar{t}} = \frac{l(l+1)}{2} t + l\bar{t} + \bar{\bar{t}}.$$

$\mathbf{C \geq 1}$: Assume, for a specified step cost distribution $\vec{\mathbf{t}}_{\mathbf{3\times 1}}$, size distribution $\vec{\mathbf{d}}_{\mathbf{3\times 1}}$, and C available fat checkpoints the optimal value $\hat{l}_\star^1$, satisfying $\mathbf{T}_h^{1,\hat{l}_\star^1}(l, C) = \min\limits_{0<\hat{l}<l} \mathbf{T}_h^{1,\hat{l}}(l, C)$ is determined. Therefore,

$$\mathbf{F}_h^1(l+1, C) = \mathbf{T}_h^1(l+1, C) - \mathbf{T}_h^1(l, C) =$$
$$= \mathbf{T}_h^1(l+1, C) - \mathbf{T}_{bin}(l - \hat{l}_\star^1, c)t - \mathbf{T}_h^1(l - \hat{l}_\star^1, C) - (l - \hat{l}_\star^1)\bar{t} - \mathbf{T}_h(\hat{l}_\star^1, C-1).$$

To evaluate $\mathbf{F}_h^1(\vec{\mathbf{d}}_{\mathbf{3\times(l+1)}}, C)$, apply Lemma 4.2 with $m = l+1$. Therefore, we have to consider two cases: $\hat{m}_\star^1 = \hat{l}_\star^1 + 1$ and $\hat{m}_\star^1 = \hat{l}_\star^1$.

<u>*First case*</u>: $\hat{m}_\star^1 = \hat{l}_\star^1 + 1$ for $m = l+1$.

$$\mathbf{F}_h^1(l+1, C) = \mathbf{T}_{bin}(l - \hat{l}_\star^1, c)t + \mathbf{T}_h^1(l - \hat{l}_\star^1, C) +$$
$$+ \mathbf{T}_h(\hat{l}_\star^1 + 1, C-1) + (l + 1 - \hat{l}_\star^1 - 1)\bar{t} -$$
$$- \mathbf{T}_{bin}(l - \hat{l}_\star^1, c)t - \mathbf{T}_h^1(l - \hat{l}_\star^1, C) - \mathbf{T}_h(\hat{l}_\star^1, C-1) - (l - \hat{l}_\star^1)\bar{t} =$$
$$= \mathbf{T}_h(\hat{l}_\star^1 + 1, C-1) - \mathbf{T}_h(\hat{l}_\star^1, C-1) = \mathbf{F}_h(\hat{l}_\star^1 + 1, C-1).$$

<u>Second case</u>: $\hat{m}^1_\star = \hat{l}^1_\star$ for $m = l + 1$.

$$\begin{aligned}\mathbf{F}^1_h(l+1, C) = \mathbf{T}_{bin}(l+1-\hat{l}^1_\star, c)t + \mathbf{T}^1_h(l+1-\hat{l}^1_\star, C) + \\ +\mathbf{T}_h(\hat{l}^1_\star, C-1) + (l+1-\hat{l}^1_\star)\bar{t} - \\ -\mathbf{T}_{bin}(l-\hat{l}^1_\star, c)t - \mathbf{T}^1_h(l-\hat{l}^1_\star, C) - \\ -\mathbf{T}_h(\hat{l}^1_\star, C-1) - (l-\hat{l}^1_\star)\bar{t} = \\ = \mathbf{T}^1_h(l+1-\hat{l}^1_\star, C) - \mathbf{T}^1_h(l-\hat{l}^1_\star, C) + \bar{t} + \\ +\mathbf{T}_{bin}(l+1-\hat{l}^1_\star, c)t - \mathbf{T}_{bin}(l-\hat{l}^1_\star, c)t = \\ = r(l-\hat{l}^1_\star+1, c)t + \mathbf{F}^1_h(l-\hat{l}^1_\star+1, C) + \bar{t}.\end{aligned}$$

Similar to the computation $\mathbf{F}_h(l+1, C)$, according to (4.2.5), the function $\mathbf{F}^1_h(l+1, C)$ can be evaluated by

$$\mathbf{F}^1_h(l+1, C) = \min\Big\{\mathbf{F}_h(\hat{l}^1_\star+1, C-1); \\ r(l-\hat{l}^1_\star+1, c)t + \mathbf{F}^1_h(l-\hat{l}^1_\star+1, C) + \bar{t}\Big\}.$$

□

As a conclusion of Lemma 4.1, Lemma 4.2 and Procedure 4.1, the following Algorithm 4.2 is established. Algorithm 4.2 constructs an admissible nested reversal schedule $\mathbf{S}_h(l, C)$. This can be applied to the implementation of a multiple sweep evolution $\mathcal{E}_{3\times l}$ with a corresponding step cost distribution $\vec{\mathbf{t}}_{\mathbf{3\times l}} = (t, \bar{t}, \bar{\bar{t}})^\top$, a size distribution $\vec{\mathbf{d}}_{\mathbf{3\times l}} = (d, \bar{d}, \bar{\bar{d}})^\top$, so that $\bar{d} = cd$, and a given number C of fat checkpoints.

The memory and run-time requirement for the implementation of Algorithm 4.2 and, consequently, for the construction of an appropriate nested reversal schedule $\mathbf{S}_h(l, C)$ is of order $\mathcal{O}(lC)$. For the implementation of Algorithm 4.2 there is a software available **heuristic**, which is coded using the C++ programming language environment. Among various evaluations- and run-time tests, this is utilised to establish the next example and the following run-time tests.

Figure 4.14 shows one example for a nested reversal schedule $\mathbf{S}_h(9, 2)$ for an evolution $\mathcal{E}_{3\times 9}$ with the step cost distribution $\vec{\mathbf{t}}_{\mathbf{3\times 9}} = (1, 5, 1)^\top$ and the size distribution $\vec{\mathbf{d}}_{\mathbf{3\times 9}} = (1, 1, 1)^\top$. $\mathcal{E}_{3\times 9}$ 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 additional fat checkpoint. Moreover, one thin checkpoint can be stored in place of a single fat one. Furthermore, the initial original state x_0 is stored into the additional thin checkpoint. Nested reversal schedule $\mathbf{S}_h(9, 2)$ is constructed using the heuristic in Algorithm 4.2. In Figure 4.14 time steps

Algorithm 4.2 Construction of $\mathbf{S}_h(l, C)$

Input: $\mathcal{E}_{3\times l}$ - *multiple sweep evolution*

$\vec{\mathbf{t}}_{\mathbf{3}\times\mathbf{1}} = (t, \bar{t}, \bar{\bar{t}})^\top$ - *step cost distribution*

$\vec{\mathbf{d}}_{\mathbf{3}\times\mathbf{1}} = (d, \bar{d}, \bar{\bar{d}})^\top$ - *size distribution*

C - *number of available fat checkpoints*

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

For $(m = 0; m < l; m++)$

  For $(n = 0; n \leq C; n++)$

    If $(m + 1 \leq n + 1)$

     $\hat{g}_\star = m$ *for* $g = m + 1$

     $\hat{g}^1_\star = m$ *for* $g = m + 1$

     $\mathbf{T}_h(m+1, n) = (m+1)t + m\bar{t} + (m+1)\bar{\bar{t}}$

     $\mathbf{F}_h(m+1, n) = t + \bar{t} + \bar{\bar{t}}$

     $\mathbf{T}^1_h(m+1, n) = mt + m\bar{t} + (m+1)\bar{\bar{t}}$

     $\mathbf{F}^1_h(m+1, n) = t + \bar{t} + \bar{\bar{t}}$

    Else

     If $(n == 0)$

$$\mathbf{T}_h(m+1, 0) = (m+1)t + \sum_{i=1}^{m+1} \{\mathbf{T}_{bin}(i,1)t + (i-1)\bar{t}\} + (m+1)\bar{\bar{t}}$$

$$\mathbf{F}_h(m+1, 0) = \tfrac{m(m+1)+2}{2}t + m\bar{t} + \bar{\bar{t}}$$

$$\mathbf{T}^1_h(m+1, 0) = \sum_{i=1}^{m+1} \{\mathbf{T}_{bin}(i,1)t + (i-1)\bar{t}\} + (m+1)\bar{\bar{t}}$$

$$\mathbf{F}^1_h(m+1, 0) = \tfrac{m(m+1)}{2}t + m\bar{t} + \bar{\bar{t}}$$

     Else

      If $\left(\mathbf{F}_h(\hat{m}_\star + 1, n-1) < r(m - \hat{m}_\star + 2, c)t + \mathbf{F}^1_h(m - \hat{m}_\star + 1, n) + \bar{t}\right)$

       $\hat{g}_\star = \hat{m}_\star + 1$ *for* $g = m + 1$

       $\mathbf{F}_h(m+1, n) = \mathbf{F}_h(\hat{m}_\star + 1, n-1)$

      Else

       $\hat{g}_\star = \hat{m}_\star$ *for* $g = m + 1$

       $\mathbf{F}_h(m+1, n) = r(m - \hat{m}_\star + 2, c)t + \mathbf{F}^1_h(m - \hat{m}_\star + 1, n) + \bar{t}$

      If $\left(\mathbf{F}_h(\hat{m}^1_\star + 1, n-1) < r(m - \hat{m}^1_\star + 1, c)t + \mathbf{F}^1_h(m - \hat{m}^1_\star + 1, n) + \bar{t}\right)$

       $\hat{g}^1_\star = \hat{m}^1_\star + 1$ *for* $g = m + 1$

       $\mathbf{F}^1_h(m+1, n) = \mathbf{F}_h(\hat{m}^1_\star + 1, n-1)$

      Else

       $\hat{g}^1_\star = \hat{m}^1_\star$ *for* $g = m + 1$

       $\mathbf{F}^1_h(m+1, n) = r(m - \hat{m}^1_\star + 1, c)t + \mathbf{F}^1_h(m - \hat{m}^1_\star + 1, n) + \bar{t}$

      $\mathbf{T}_h(m+1, n) = \mathbf{T}_h(m, n) + \mathbf{F}_h(m+1, n)$

      $\mathbf{T}^1_h(m+1, n) = \mathbf{T}^1_h(m, n) + \mathbf{F}^1_h(m+1, n)$

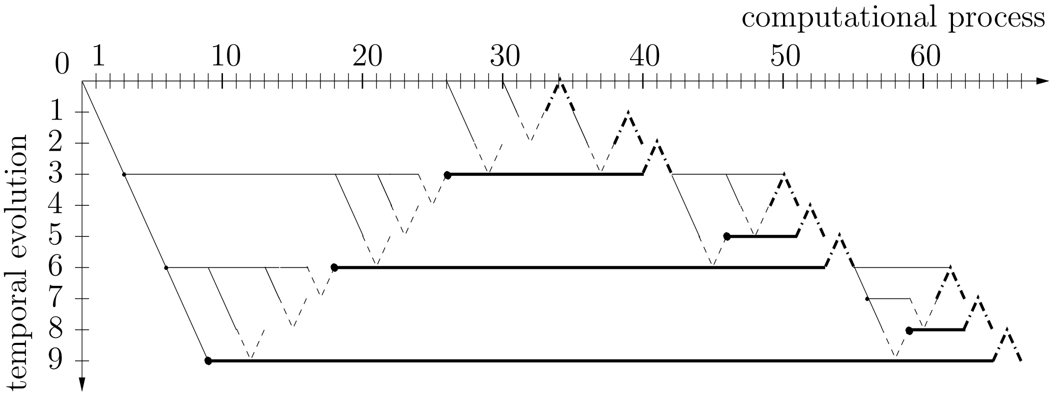

Figure 4.14: Nested reversal schedules $\mathbf{S}_h(9,2)$ constructed using the heuristic in Algorithm 4.2

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 original steps F_i, $i = 1,\ldots,9$, whereas adjoint steps $\bar{F}_i$, $i = 1,\ldots,9$, are visualised by dotted slanted lines. Final steps $\bar{\bar{F}}_i$, $i = 1,\ldots,9$, are drawn by slanted dashed-dotted thick lines.

Nested reversal schedule $\mathbf{S}_h(9,2)$ starts with the action R_0 restoring the initial original state x_0 from the 0th thin checkpoint. Three actions A are executed by performing three original steps F_1, F_2, and F_3, consecutively. The original state x_3 is stored into the first thin checkpoint by the action W_1. Now, again, three actions A are applied to perform three original steps F_4, F_5, and F_6, and the original state x_6 is stored into the second thin checkpoint by the action W_2. Furthermore, the original steps F_7, F_8, and F_9 are executed by three actions A and the adjoint state $\bar{x}_9$ is initialised. The adjoint state $\bar{x}_9$ is stored into the additional fat checkpoint by the action $\bar{W}_0$. The adjoint sweep is started by this action.

Furthermore, data of the original state x_6 is restored from the second thin checkpoint by the action R_2, the original steps F_7 and F_8 are performed, and the adjoint state $\bar{x}_8$ can be evaluated by the application of the action $\bar{A}$. Then, the original state x_6 is anew restored from the second thin checkpoint by the action R_2, the original state x_7 is reevaluated by the action A, the adjoint states $\bar{x}_7$ and $\bar{x}_6$ are evaluated by the application of the action $\bar{A}$

two times. The adjoint state $\bar{x}_6$ is stored into the first fat checkpoint by the action $\bar{W}_1$. Then, we go back to the adjoint state $\bar{x}_3$ by reevaluating required original states and storing the adjoint state $\bar{x}_3$ into the second fat checkpoint by the action $\bar{W}_2$. Furthermore, 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. Furthermore, we store the current initial original state x_1 into the additional 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 $\mathbf{S}_h(9, 2)$ in Figure 4.14, it is necessary to perform 24 original steps F_i, 13 adjoint steps $\bar{F}_i$, and nine final steps $\bar{\bar{F}}_i$. The nested reversal schedule $\mathbf{S}_h(9, 2)$ and its minimal evaluation cost $\mathbf{T}_h(9, 2)$ are determined using Algorithm 4.2 and routine **heuristic**.

If we compare Figure 4.10 and Figure 4.14, we see that nested reversal schedules shown in these figures are different, although the parameters $\vec{\mathbf{d}}_{\mathbf{3}\times\mathbf{9}}$, $\vec{\mathbf{t}}_{\mathbf{3}\times\mathbf{9}}$, and $C = 2$ are the same for the both schedules. The nested reversal schedule $\mathbf{S}_h(9, 2)$ in Figure 4.14 can be represented as a decomposition of nested and binomial reversal schedules, according to Figure 4.11. Obviously, the optimal nested reversal schedule $\mathbf{S}_{min}(9, 2)$ in Figure 4.10 does not possess this property. Moreover, nested reversal schedules $\mathbf{S}_{min}(9, 2)$ and $\mathbf{S}_h(9, 2)$ in Figure 4.10 and in Figure 4.14, respectively, differ from each other in the number of original states, required for their execution. For performing $\mathbf{S}_{min}(9, 2)$ 23 original steps are needed. For performing $\mathbf{S}_h(9, 2)$ this number increases up to one.

The resulting schedule $\mathbf{S}_h(l, C)$, constructed by Linear Method, in general is not an optimal one. Figure 4.15 shows deviations of the corresponding evaluation cost $\mathbf{T}_h(l, C)$ from the minimal evaluation cost $\mathbf{T}_{min}(l, C)$ for different values of l, C, and c, using the step cost distribution $\vec{\mathbf{t}}_{\mathbf{3}\times\mathbf{l}} = (1, 5, 1)^\top$ and different size distributions $\vec{\mathbf{d}}_{\mathbf{3}\times\mathbf{l}} = (1, c, 1)^\top$, which depend on the parameter c. The horizontal axis denotes the number l of time steps. Deviations, evaluated by the formula

$$\frac{(\mathbf{T}_h(l, C) - \mathbf{T}_{min}(l, C))}{\mathbf{T}_{min}(l, C)} \times 100, \tag{4.81}$$

are plotted along the vertical axis. As we can see in Figure 4.15, 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 $\bar{t}/t \gg 5$. Due to the huge temporal complexity, needed to evaluate optimal nested reversal schedules $\mathbf{S}_{min}(l, C)$ and, respectively, to

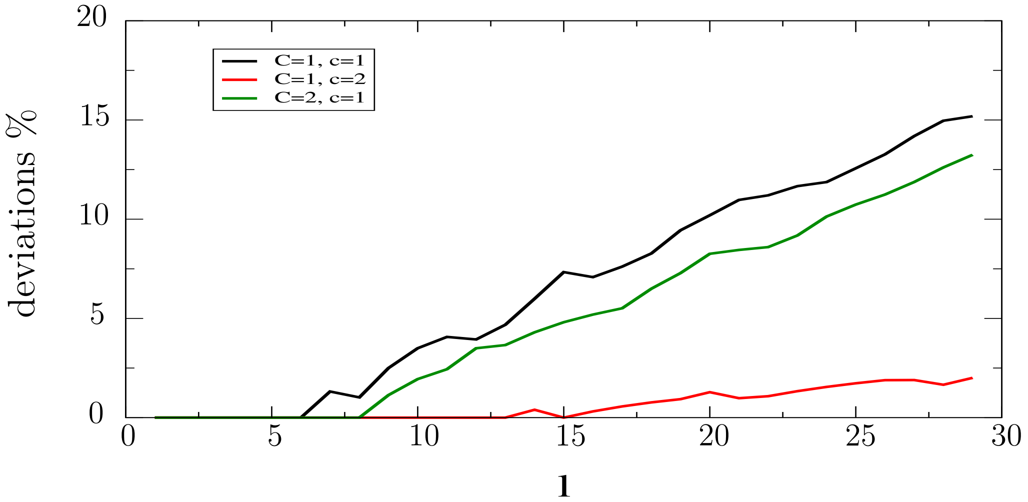

Figure 4.15: Deviations of $\mathbf{T}_h(l, C)$ from $\mathbf{T}_{min}(l, C)$

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 4.15. 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 the following section.

4.2.6 Lower Bound for Minimal Evaluation Cost

Consider an evolution $\mathcal{E}_{3\times l}$ traversing l time steps in three alternative sweeps. $\vec{\mathbf{t}}_{\mathbf{3}\times\mathbf{l}} = (t, \bar{t}, \bar{\bar{t}})^\top$ and $\vec{\mathbf{d}}_{\mathbf{3}\times\mathbf{l}} = (d, \bar{d}, \bar{\bar{d}})^\top$ are the 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 $\bar{x}_i$, $0 \leq i \leq l$. Assume that c thin checkpoints can be stored in place of a single fat one, i.e. $\bar{d} = cd$. According to Section 4.2.3, the construction of an optimal nested reversal schedule $\mathbf{S}_{min}(l, C)$, 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 convenient to determine an estimation for the minimal evaluation cost $\mathbf{T}_{min}(l, C)$ before an optimal nested reversal schedule is constructed. A lower bound for the minimal evaluation cost $\mathbf{T}_{min}(l, C)$ is denoted $\mathbf{L}_{min}(l, C)$, which can be constructed straightforwardly.

According to the definition of the minimal evaluation cost $\mathbf{T}_{min}(l, C)$, the following relation is satisfied

$$\mathbf{T}_{min}(l, C) = \mathbf{T}_f(l, C)t + \mathbf{T}_a(l, C)\bar{t} + l\bar{\bar{t}}, \tag{4.82}$$

with

$\mathbf{T}_f(l, C)$	$\equiv$	the overall number of original steps F_i, needed to perform during the execution of $\mathbf{S}_{min}(l, C)$,
$\mathbf{T}_a(l, C)$	$\equiv$	the overall number of adjoint steps $\bar{F}_i$, needed to perform during the execution of $\mathbf{S}_{min}(l, C)$.

Thus, the lower bound $\mathbf{L}_{min}(l, C)$ can be evaluated by

$$\mathbf{L}_{min}(l, C) = \mathbf{L}_f(l, C)t + \mathbf{L}_a(l, C)\bar{t} + l\bar{\bar{t}}, \tag{4.83}$$

with $\mathbf{L}_f(l, C)$ and $\mathbf{L}_a(l, C)$ being estimations for $\mathbf{T}_f(l, C)$ and $\mathbf{T}_a(l, C)$, respectively, so that the following relations are satisfied

$$\mathbf{L}_f(l, C) \le \mathbf{T}_f(l, C), \tag{4.84}$$

$$\mathbf{L}_a(l, C) \le \mathbf{T}_a(l, C). \tag{4.85}$$

Therefore, according to (4.83), we obtain the desired relation

$$\mathbf{L}_{min}(l, C) \le \mathbf{T}_{min}(l, C). \tag{4.86}$$

Thus, the task of finding the lower bound $\mathbf{L}_{min}(l, C)$ is reduced to the task of evaluating the corresponding estimations $\mathbf{L}_f(l, C)$ and $\mathbf{L}_a(l, C)$.

Firstly, to evaluate the estimation $\mathbf{L}_a(l, C)$, assume that the original cost t is neglectable small. Practically, this is often the case. Therefore, the original sweep can be omitted. The evolution $\mathcal{E}_{3\times l}$, consisting of 3 sweeps, is transformed into the evolution $\mathcal{E}_{2\times l}$, consisting of 2 sweeps, namely, the adjoint sweep and the final sweep. The minimal number of adjoint steps, needed to implement the evolution $\mathcal{E}_{2\times l}$, using up to C checkpoints with an uniform step cost distribution, corresponds to the minimal evaluation cost $\mathbf{T}_{bin}(l, C+1)$ of the binomial reversal schedule $\mathbf{S}_{bin}(l, C+1)$ (see Chapter 2). Thus,

$$\mathbf{L}_a(l, C) = \mathbf{T}_{bin}(l, C+1) = \sum_{i=1}^{l} r(C+1, i), \tag{4.87}$$

with $r(C+1, i) = r_{bin}(C+1, i)$. Obviously, (4.85) is satisfied, if $\mathbf{L}_a(l, C)$ is evaluated by (4.87).

To evaluate the estimation $\mathbf{L}_f(l, C)$ for the minimal value $\mathbf{T}_f(l, C)$, consider anew the transformation of the evolution $\mathcal{E}_{3\times l}$, consisting of 3 sweeps, into the evolution $\mathcal{E}_{2\times l}$. The evolution $\mathcal{E}_{2\times l}$ consists of 2 sweeps, namely, the adjoint sweep and the final sweep as before. The step cost distribution $\bar{\mathbf{w}}_\mathbf{l}$ of the adjoint sweep in the evolution $\mathcal{E}_{2\times l}$ is non-uniform in this case. This is defined by

$$\bar{\mathbf{w}}_\mathbf{l} = \langle w_1, \ldots, w_l \rangle \,. \tag{4.88}$$

Here, each weight $w_i,\ 1 \le i \le l$, gives an estimation for the number of original steps, needed to perform, in order to evaluate the corresponding adjoint step $\bar{F}_i$. Note, it is not necessary to determine exact numbers of evaluated original steps. This is sufficient to determine their lower bounds $w_i,\ 1 \le i \le l$.

To do this, recall the correspondence between original steps $F_i,\ 1 \le i \le l$, and repetition numbers according to Figure 2.11 and Lemma 2.1. Thus, for a sequence of l original steps with uniform step cost distribution and $\hat{c} = cC + 1$ checkpoints available, the minimal evaluation cost $\mathbf{T}_{bin}(l, \hat{c})$, needed to reverse this sequence, is given by $\mathbf{T}_{bin}(l, \hat{c}) = \sum\limits_{i=1}^{l} r(\hat{c}, i)$. Therefore, each original step $F_i,\ 1 \le i \le l$, can be associated with the repetition number $r(\hat{c}, i) = r_{bin}(\hat{c}, i)$. The repetition number $r_{bin}(\hat{c}, i)$ can be used as a lower bound w_i. Therefore, the step cost distribution $\bar{\mathbf{w}}_\mathbf{l}$ is defined by

$$\bar{\mathbf{w}}_\mathbf{l} = \langle r_{bin}(\hat{c}, 1), \ldots, r_{bin}(\hat{c}, l) \rangle \,. \tag{4.89}$$

Moreover, if one has a partition of l original steps according to Figure 4.16,

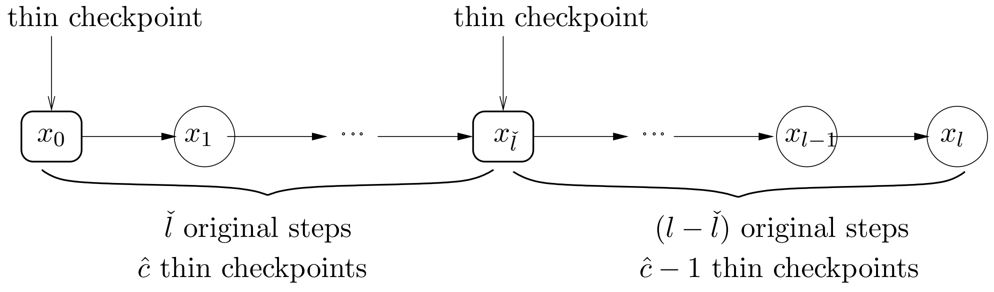

Figure 4.16: Decomposition of a multiple sweep evolution $\mathcal{E}_{3\times l}$ regarding the original sweep

then, the following relations are satisfied

$$\mathbf{T}_{bin}(l,\hat{c}) = \sum_{i=1}^{l} r_{bin}(\hat{c},i) \leq \check{l} + \mathbf{T}_{bin}(\check{l},\hat{c}) + \mathbf{T}_{bin}(l-\check{l},\hat{c}-1)$$

(4.90)

$$\leq \check{l} + \sum_{i=1}^{\check{l}} r_{bin}(\hat{c},i) + \mathbf{T}_{bin}(l-\check{l},\hat{c}-1).$$

Note, that if $\check{l}$ is evaluated using Theorem 2.3, i.e. using the binomial rule, then, the relations in (4.90) are satisfied with " $=$ " in both cases. Therefore, this implies

(4.91)

$$\sum_{i=1}^{l} r_{bin}(\hat{c},i) - \sum_{i=1}^{\check{l}} r_{bin}(\hat{c},i) = \sum_{i=\check{l}+1}^{l} r_{bin}(\hat{c},i) \leq \check{l} + \mathbf{T}_{bin}(l-\check{l},\hat{c}-1).$$

During constructing the lower bound $\mathbf{L}_f(l,C)$ the value $\check{l}$ corresponds to the value $\hat{l}_C$, i.e. to the number of adjoint state stored into the Cth fat checkpoint. Obviously, this value is initially unknown. Therefore, the sum $\sum_{i=\check{l}+1}^{l} r_{bin}(\hat{c},i)$ is to evaluate exactly in C steps during the construction of $\mathbf{L}_f(l,C)$, i.e.

(4.92)

$$\sum_{i=\check{l}+1}^{l} r_{bin}(\hat{c},i) = \sum_{i=\hat{l}_C+1}^{l} r_{bin}(\hat{c},i) = \sum_{i=\hat{l}_C+1}^{\hat{l}_{C-1}} r_{bin}(\hat{c},i) + \ldots + \sum_{i=\hat{l}_1+1}^{l=\hat{l}_0} r_{bin}(\hat{c},i).$$

Thus, the lower bound $\mathbf{L}_f(l,C) = \mathbf{L}_f(l,C,\hat{c})$ can be recursively constructed by minimising

$$\begin{aligned}\mathbf{L}_f(l,C,\hat{c}) &= \min_{0<\hat{l}<l} \Bigg\{ \sum_{i=\hat{l}+1}^{l+1} r_{bin}(\hat{c},i) + \mathbf{L}_f^1(l-\hat{l},C,cC+1) + \\ &+ \mathbf{L}_f^1(\hat{l},C-1,\hat{c}) \Bigg\},\end{aligned} \tag{4.93}$$

with

$$\begin{aligned}\mathbf{L}_f^1(l,C,\hat{c}) &= \min_{0<\hat{l}<l} \Bigg\{ \sum_{i=\hat{l}+1}^{l} r_{bin}(\hat{c},i) + \mathbf{L}_f^1(l-\hat{l},C,cC+1) + \\ &+ \mathbf{L}_f^1(\hat{l},C-1,\hat{c}) \Bigg\},\end{aligned} \tag{4.94}$$

$$\mathbf{L}_f(l,0,\hat{c}) \;=\; l + \sum_{i=1}^{l} \mathbf{T}_{bin}(i,1), \tag{4.95}$$

$$\mathbf{L}_f^1(l,0,\hat{c}) \;=\; \sum_{i=1}^{l} \mathbf{T}_{bin}(i,1), \tag{4.96}$$

$$\mathbf{L}_f(l,C,\hat{c}) \;=\; l \quad \text{if} \quad (l \le C+1), \tag{4.97}$$

and

$$\mathbf{L}_f^1(l,C,\hat{c}) \;=\; (l-1) \quad \text{if} \quad (l \le C+1). \tag{4.98}$$

To evaluate function $\mathbf{L}_f(l,C,\hat{c})$, use Linear Method from Section 4.2.5. This time, Linear Method has to be applied to the functions $\mathbf{L}_f(l,C,\hat{c})$ and $\mathbf{L}_f^1(l,C,\hat{c})$ instead of to the functions $\mathbf{T}_h(l,C)$ and $\mathbf{T}_h^1(l,C)$. Obviously, Lemma 4.1 and Lemma 4.2 can be straightforwardly transformed for the functions $\mathbf{L}_f(l,C,\hat{c})$ and $\mathbf{L}_f^1(l,C,\hat{c})$.

To find a minimal point $\hat{l}_\star$ consider the functions $\mathbf{F}_f(l,C,\hat{c})$ and $\mathbf{F}_f^1(l,C,\hat{c})$, defined by

$$\mathbf{F}_f(l,C,\hat{c}) = \mathbf{L}_f(l,C,\hat{c}) - \mathbf{L}_f(l-1,C,\hat{c}), \tag{4.99}$$

and

$$\mathbf{F}_f^1(l,C,\hat{c}) = \mathbf{L}_f^1(l,C,\hat{c}) - \mathbf{L}_f^1(l-1,C,\hat{c}). \tag{4.100}$$

The functions $\mathbf{F}_f(l,C,\hat{c})$ and $\mathbf{F}_f^1(l,C,\hat{c})$ can be evaluated recursively. This leads to Procedure 4.2.

Procedure 4.2 Evaluation $\mathbf{F}_f(l+1,C,\hat{c})$ and $\mathbf{F}_f^1(l+1,C,\hat{c})$

a). Evaluation $\mathbf{F}_f(l+1,C,\hat{c})$.

Initial case: If $l+1 \le C+1$, $\mathbf{F}_f(l+1,C,\hat{c}) = 1$.

$\mathbf{C=0}$:

$$\mathbf{F}_f(l+1,0,\hat{c}) = \mathbf{L}_f(l+1,0,\hat{c}) - \mathbf{L}_f(l,0,\hat{c}) = \tfrac{l(l+1)+2}{2}.$$

$\mathbf{C \ge 1}$: Assume, for specified l and C the optimal value $\hat{l}_\star$ is determined. Then,

$$\mathbf{F}_f(l+1,C,\hat{c}) = \mathbf{L}_f(l+1,C,\hat{c}) - \mathbf{L}_f(l,C,\hat{c}) =$$
$$= \mathbf{L}_f(l+1,C,\hat{c}) - \sum_{i=\hat{l}_\star+1}^{l+1} r_{bin}(\hat{c},i) - \mathbf{L}_f^1(l-\hat{l}_\star,C,cC+1) - \mathbf{L}_f^1(\hat{l}_\star,C-1,\hat{c}).$$

To evaluate $\mathbf{F}_f(l+1, C, \hat{c})$ consider two cases: $\hat{m}_\star = \hat{l}_\star + 1$ and $\hat{m}_\star = \hat{l}_\star$ for $m = l + 1$.

<u>First case</u>: $\hat{m}_\star = \hat{l}_\star + 1$ for $m = l + 1$.

$$\begin{aligned}\mathbf{F}_f(l+1, C, \hat{c}) = &\sum_{i=\hat{l}_\star+2}^{l+2} r_{bin}(\hat{c}, i) + \mathbf{L}_f^1(l - \hat{l}_\star, C, cC+1) + \mathbf{L}_f^1(\hat{l}_\star + 1, C-1, \hat{c}) - \\ &- \sum_{i=\hat{l}_\star+1}^{l+1} r_{bin}(\hat{c}, i) - \mathbf{L}_f^1(l - \hat{l}_\star, C, cC+1) - \mathbf{L}_f^1(\hat{l}_\star, C-1, \hat{c}) = \\ &= r_{bin}(\hat{c}, l+2) - r_{bin}(\hat{c}, \hat{l}_\star + 1) + \mathbf{F}_f^1(\hat{l}_\star + 1, C-1, \hat{c}).\end{aligned}$$

<u>Second case</u>: $\hat{m}_\star = \hat{l}_\star$ for $m = l + 1$.

$$\begin{aligned}\mathbf{F}_f(l+1, C, \hat{c}) = &\sum_{i=\hat{l}_\star+1}^{l+2} r_{bin}(\hat{c}, i) + \mathbf{L}_f^1(l + 1 - \hat{l}_\star, C, cC+1) + \mathbf{L}_f^1(\hat{l}_\star, C-1, \hat{c}) - \\ &- \sum_{i=\hat{l}_\star+1}^{l+1} r_{bin}(\hat{c}, i) - \mathbf{L}_f^1(l - \hat{l}_\star, C, cC+1) - \mathbf{L}_f^1(\hat{l}_\star, C-1, \hat{c}) = \\ &= r_{bin}(\hat{c}, l+2) + \mathbf{F}_f^1(l + 1 - \hat{l}_\star, C, cC+1).\end{aligned}$$

Summarising, the function $\mathbf{F}_f(l+1, C, \hat{c})$ can be evaluated by

$$\begin{aligned}\mathbf{F}_f(l+1, C, \hat{c}) \;&=\; \min\Big\{ r_{bin}(\hat{c}, l+2) - r_{bin}(\hat{c}, \hat{l}_\star + 1) + \mathbf{F}_f^1(\hat{l}_\star + 1, C-1, \hat{c}); \\ &\qquad r_{bin}(\hat{c}, l+2) + \mathbf{F}_f^1(l + 1 - \hat{l}_\star, C, cC+1) \Big\}.\end{aligned}$$

b). Evaluation $\mathbf{F}_f^1(l+1, C, \hat{c})$.

Initial case: If $l + 1 \le C + 1$, $\mathbf{F}_f^1(l+1, C, \hat{c}) = 1$.

$\mathbf{C = 0}$:

$$\mathbf{F}_f^1(l+1, 0, \hat{c}) = \mathbf{L}_f^1(l+1, 0, \hat{c}) - \mathbf{L}_f^1(l, 0, \hat{c}) = \tfrac{l(l+1)}{2}.$$

$\mathbf{C \ge 1}$: Assume, for specified l and C the optimal value $\hat{l}_\star^1$ is determined. Then,

$$\begin{aligned}&\mathbf{F}_f^1(l+1, C, \hat{c}) = \mathbf{L}_f^1(l+1, C, \hat{c}) - \mathbf{L}_f^1(l, C, \hat{c}) = \\ &= \mathbf{L}_f^1(l+1, C, \hat{c}) - \sum_{i=\hat{l}_\star^1+1}^{l} r_{bin}(\hat{c}, i) - \mathbf{L}_f^1(l - \hat{l}_\star^1, C, cC+1) - \mathbf{L}_f^1(\hat{l}_\star^1, C-1, \hat{c}).\end{aligned}$$

To evaluate $\mathbf{F}_f^1(l+1, C, \hat{c})$ consider two cases: $\hat{m}_\star^1 = \hat{l}_\star^1 + 1$ and $\hat{m}_\star^1 = \hat{l}_\star^1$ for $m = l + 1$.

<u>First case</u>: $\hat{m}^1_\star \;=\; \hat{l}^1_\star + 1$ for $m = l + 1$.

$$\begin{aligned}\mathbf{F}^1_f(l+1, C, \hat{c}) = \sum_{i=\hat{l}^1_\star+2}^{l+1} r_{bin}(\hat{c}, i) + \mathbf{L}^1_f(l - \hat{l}^1_\star, C, cC+1) + \mathbf{L}^1_f(\hat{l}^1_\star + 1, C-1, \hat{c}) - \\ - \sum_{i=\hat{l}^1_\star+1}^{l} r_{bin}(\hat{c}, i) - \mathbf{L}^1_f(l - \hat{l}^1_\star, C, cC+1) - \mathbf{L}^1_f(\hat{l}^1_\star, C-1, \hat{c}) = \\ = r_{bin}(\hat{c}, l+1) - r_{bin}(\hat{c}, \hat{l}^1_\star + 1) + \mathbf{F}^1_f(\hat{l}^1_\star + 1, C-1, \hat{c}).\end{aligned}$$

<u>Second case</u>: $\hat{m}^1_\star \;=\; \hat{l}^1_\star$ for $m = l + 1$.

$$\begin{aligned}\mathbf{F}^1_f(l+1, C, \hat{c}) = \sum_{i=\hat{l}^1_\star+1}^{l+1} r_{bin}(\hat{c}, i) + \mathbf{L}^1_f(l + 1 - \hat{l}^1_\star, C, cC+1) + \mathbf{L}^1_f(\hat{l}^1_\star, C-1, \hat{c}) - \\ - \sum_{i=\hat{l}^1_\star+1}^{l} r_{bin}(\hat{c}, i) - \mathbf{L}^1_f(l - \hat{l}^1_\star, C, cC+1) - \mathbf{L}^1_f(\hat{l}^1_\star, C-1, \hat{c}) = \\ = r_{bin}(\hat{c}, l+1) + \mathbf{F}^1_f(l + 1 - \hat{l}^1_\star, C, cC+1).\end{aligned}$$

Summarising, the function $\mathbf{F}^1_f(l+1, C, \hat{c})$ can be evaluated by

$$\begin{aligned}\mathbf{F}^1_f(l+1, C, \hat{c}) \;&=\; \min\Big\{ r_{bin}(\hat{c}, l+1) - r_{bin}(\hat{c}, \hat{l}^1_\star + 1) + \mathbf{F}^1_f(\hat{l}^1_\star + 1, C-1, \hat{c}); \\ &\qquad r_{bin}(\hat{c}, l+1) + \mathbf{F}^1_f(l + 1 - \hat{l}^1_\star, C, cC+1) \Big\}.\end{aligned}$$

□

As a conclusion of Procedure 4.2, the following Algorithm 4.3 is established. Algorithm 4.3 evaluates the functions $\mathbf{L}_f(l, C, \hat{c})$ and $\mathbf{L}^1_f(l, C, \hat{c})$ for a given number l of time steps and C checkpoints.

Therefore, using the estimations $\mathbf{L}_f(l, C) = \mathbf{L}_f(l, C, \hat{c})$ and $\mathbf{L}_a(l, C)$, evaluated by Algorithm 4.3 and (4.87), respectively, the lower bound $\mathbf{L}_{min}(l, C)$ for the minimal evaluation cost $\mathbf{T}_{min}(l, C)$ can be computed by

$$\mathbf{L}_{min}(l, C) \;=\; \mathbf{L}_f(l, C)t + \sum_{i=1}^{l} r_{bin}(C+1, i)\bar{t} + l\bar{\bar{t}}. \tag{4.101}$$

Figure 4.17 illustrates various examples for the minimal evaluation cost $\mathbf{T}_{min}(l, C)$ and its lower bound $\mathbf{L}_{min}(l, C)$. This is evaluated by (4.101) for different values l, C, and c, using the step cost distribution $\vec{\mathbf{t}}_{\mathbf{3}\times\mathbf{1}} = (1, 5, 1)^\top$ and different

Algorithm 4.3 Evaluation $\mathbf{L}_f(l, C, \hat{c})$ and $\mathbf{L}_f^1(l, C, \hat{c})$
Input: l - *number of time steps*
C - *number of available checkpoints*
Output: *values of* $\hat{m}_\star$, $\hat{m}_\star^1$, $\mathbf{L}_f(m,n,ck{+}1)$, $\mathbf{L}_f^1(m,n,ck{+}1)$, $\mathbf{F}_f(m,n,ck{+}1)$ *and* $\mathbf{F}_f^1(m,n,ck+1))$ *for* $0 < m \le l,\ 0 \le n \le C,$ *and* $0 \le k \le C$.
For $(m = 0;\ m < l;\ m++)$
  For $(n = 0;\ n \le C;\ n++)$
    For $(k = 0;\ k \le C;\ k++)$
      If $(m + 1 \le n + 1)$
        $\hat{g}_\star = m$ *for* $g = m + 1$
        $\hat{g}_\star^1 = m$ *for* $g = m + 1$
        $\mathbf{L}_f(m+1, n, ck+1) = m + 1$
        $\mathbf{F}_f(m+1, n, ck+1) = 1$
        $\mathbf{L}_f^1(m+1, n, ck+1) = m$
        $\mathbf{F}_f^1(m+1, n, ck+1) = 1$
      Else
        If $(n == 0)$
$$\mathbf{L}_f(m+1, 0, ck+1) = (m+1) + \sum_{i=1}^{m+1} \mathbf{T}_{bin}(i, 1)$$
$$\mathbf{F}_f(m+1, 0, ck+1) = \tfrac{m(m+1)+2}{2}$$
$$\mathbf{L}_f^1(m+1, 0, ck+1) = \sum_{i=1}^{m+1} \mathbf{T}_{bin}(i, 1)$$
$$\mathbf{F}_f^1(m+1, 0, ck+1) = \tfrac{m(m+1)}{2}$$
        Else
          If $\left\{\mathbf{F}_f^1(\hat{m}_\star + 1, n-1, ck+1) - r_{bin}(ck+1, \hat{m}_\star + 1) < \mathbf{F}_f^1(m + 1 - \hat{m}_\star, n, cn+1)\right\}$
            $\hat{g}_\star = \hat{m}_\star + 1$ *for* $g = m + 1$
            $\mathbf{F}_f(m+1, n, ck+1) = r_{bin}(ck+1, m+2) - r_{bin}(ck+1, \hat{m}_\star + 1) + \mathbf{F}_f^1(\hat{m}_\star + 1, n-1, ck+1)$
          Else
            $\hat{g}_\star = \hat{m}_\star$ *for* $g = m + 1$
            $\mathbf{F}_f(m+1, n, ck+1) = r_{bin}(ck+1, m+2) + \mathbf{F}_f^1(m+1-\hat{m}_\star, n, cn+1)$
          If $\left\{\mathbf{F}_f^1(\hat{m}_\star^1 + 1, n-1, ck+1) - r_{bin}(ck+1, \hat{m}_\star^1 + 1) < \mathbf{F}_f^1(m + 1 - \hat{m}_\star^1, n, cn+1)\right\}$
            $\hat{g}_\star^1 = \hat{m}_\star^1 + 1$ *for* $g = m + 1$
            $\mathbf{F}_f^1(m+1, n, ck+1) = r_{bin}(ck+1, m+1) - r_{bin}(ck+1, \hat{m}_\star^1 + 1) + \mathbf{F}_f^1(\hat{m}_\star^1 + 1, n-1, ck+1)$
          Else
            $\hat{g}_\star^1 = \hat{m}_\star^1$ *for* $g = m + 1$
            $\mathbf{F}_f^1(m+1, n, ck+1) = r_{bin}(ck+1, m+1) + \mathbf{F}_f^1(m+1-\hat{m}_\star^1, n, cn+1)$
          $\mathbf{L}_f(m+1, n, ck+1) = \mathbf{L}_f(m, n, ck+1) + \mathbf{F}_f(m+1, n, ck+1)$
          $\mathbf{L}_f^1(m+1, n, ck+1) = \mathbf{L}_f^1(m, n, ck+1) + \mathbf{F}_f^1(m+1, n, ck+1)$

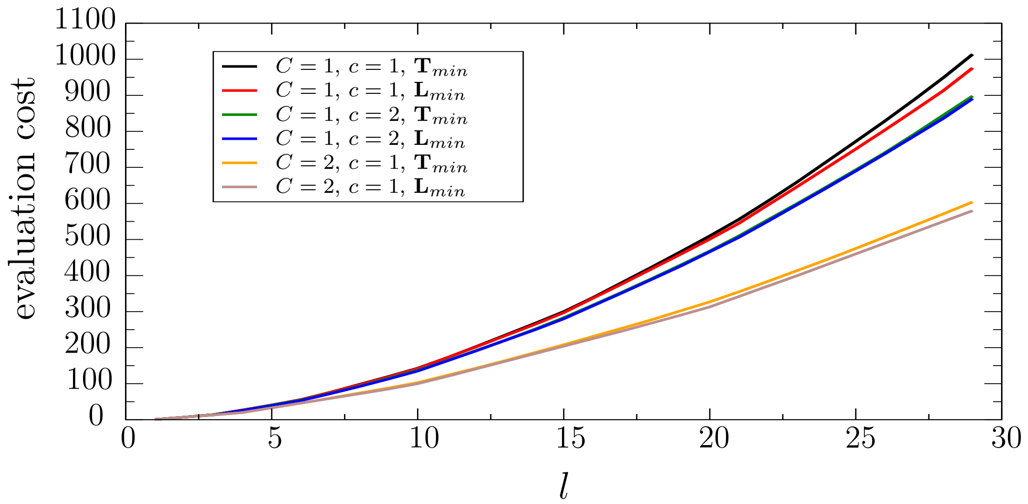

Figure 4.17: Examples for the minimal evaluation cost $\mathbf{T}_{min}(l, C)$ and its lower bound $\mathbf{L}_{min}(l, C)$

size distributions $\vec{\mathbf{d}}_{\mathbf{3}\times\mathbf{1}} = (1, c, 1)^\top$. These size distributions depend on parameter c. The horizontal axis denotes the number l of time steps. The vertical axis gives corresponding minimal evaluation costs and its lower bounds. As we can see in Figure 4.17, the lower bound $\mathbf{L}_{min}(l, C)$, evaluated by the formula (4.101), lies very close to the minimal evaluation cost $\mathbf{T}_{min}(l, C)$.

Figure 4.18 illustrates deviations of the minimal evaluation cost $\mathbf{T}_{min}(l, C)$ from its lower bound $\mathbf{L}_{min}(l, C)$. This is evaluated by (4.101) for different values l, C, and c, using the step cost distribution $\vec{\mathbf{t}}_{\mathbf{3}\times\mathbf{1}} = (1, 5, 1)^\top$ and different size distributions $\vec{\mathbf{d}}_{\mathbf{3}\times\mathbf{1}} = (1, c, 1)^\top$, which depend on parameter c. The horizontal axis denotes the number l of time steps.

Deviations are evaluated by the formula

$$\frac{(\mathbf{T}_{min}(l, C) - \mathbf{L}_{min}(l, C))}{\mathbf{T}_{min}(l, C)} \times 100. \tag{4.102}$$

As we can see in Figure 4.18, for different combinations between C and c, deviations are less than 5% w.r.t. the minimal evaluation cost $\mathbf{T}_{min}(l, C)$. Therefore, the lower bound $\mathbf{L}_{min}(l, C)$, evaluated by the formula (4.101), gives very precise estimation for the minimal evaluation cost $\mathbf{T}_{min}(l, C)$.

Because of the complexity resulting through the construction of optimal nested reversal schedules $\mathbf{S}_{min}(l, C)$ and computation minimal evaluation cost $\mathbf{T}_{min}(l, C)$, it is not possible to construct them for large numbers of l, C, and

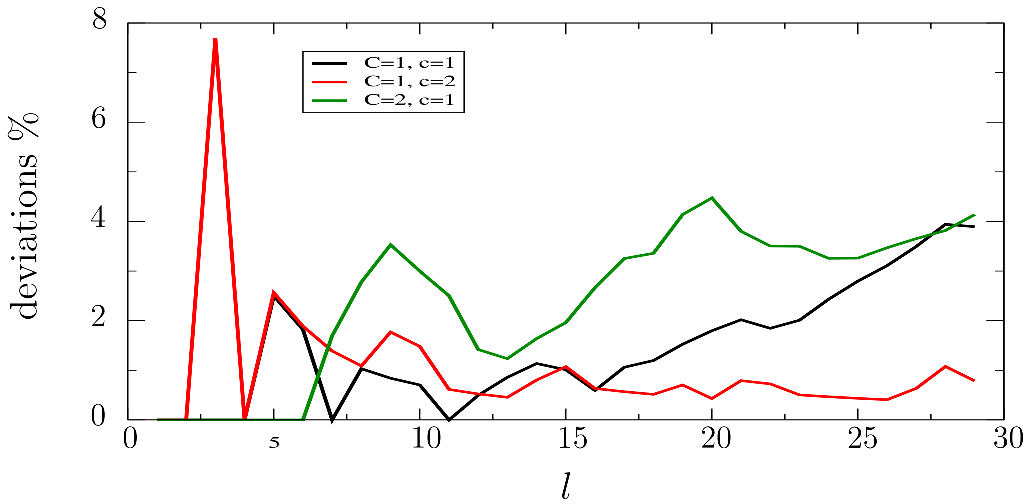

Figure 4.18: Discrepancies between the minimal evaluation cost $\mathbf{T}_{min}(l, C)$ and its lower bound $\mathbf{L}_{min}(l, C)$

c. Nevertheless, we can learn more about the quality of nested reversal schedules $\mathbf{S}_h(l, C)$ and the lower bound $\mathbf{L}_{min}(l, C)$ for the minimal evaluation cost for different parameters l, C, and c, if we compare the temporal complexity $\mathbf{T}_h(l, C)$ of nested reversal schedules $\mathbf{S}_h(l, C)$, resulting from the heuristic, and the lower bound $\mathbf{L}_{min}(l, C)$ for the minimal evaluation cost.

Figure 4.19 shows deviations of the corresponding evaluation cost $\mathbf{T}_h(l, C)$ from 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 step cost distribution $\vec{\mathbf{t}}_{\mathbf{3}\times\mathbf{1}} = (1, 5, 1)^\top$ and different size distributions $\vec{\mathbf{d}}_{\mathbf{3}\times\mathbf{1}} = (1, c, 1)^\top$, which depend on the parameter c. The horizontal axis denotes the number l of time steps. Deviations, evaluated by the formula

$$\frac{(\mathbf{T}_h(l, C) - \mathbf{L}_{min}(l, C))}{\mathbf{L}_{min}(l, C)} \times 100, \tag{4.103}$$

are plotted along the vertical axis. As we can see in Figure 4.19, deviations reach up to 20% w.r.t. the lower bound $\mathbf{L}_{min}(l, C)$. Consequently, deviations between the evaluation cost $\mathbf{T}_h(l, C)$ and the minimal evaluation cost $\mathbf{T}_{min}(l, C)$ are even smaller. In our 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. Two last examples with $c = 100$ show situations close to realistic. Here, deviations only reach up to 3% w.r.t. the lower bound $\mathbf{L}_{min}(l, C)$.

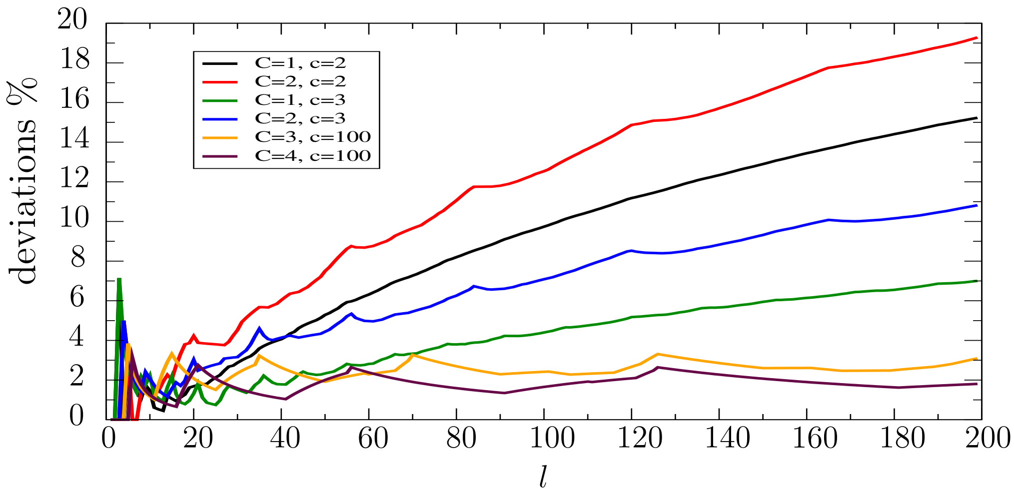

Figure 4.19: Deviations of $\mathbf{T}_h(l, C)$ from $\mathbf{L}_{min}(l, C)$

4.2.7 Logarithmic Growth of the Temporal Complexity

Summarising, we turn one more time to the important question of complexity. How much do nested reversal schedules cost in terms of storage and computation? The storage question is easier to answer. If C fat checkpoints are available, then, the storage requirement, needed to implement a corresponding nested reversal schedule, is given by

$$\begin{aligned} (C+1)\,\text{size(fat)} + \text{size(thin)} &= (C+1)c\,\text{size(thin)} + \text{size(thin)} = \\ &= (Cc + c + 1)\,\text{size(thin)}. \end{aligned} \tag{4.104}$$

Here, size(fat) and size(thin) denote the size of fat and thin checkpoints, respectively.

To estimate the computational cost of nested reversal schedules, we may use similar reasoning as in [34, 35, 38, 91]. It is convenient to measure these costs relatively to the computational cost of the basic approach. Recall, using the basic approach, all intermediate states are stored during the implementation of a multiple sweep evolution. Therefore, the storage requirement, needed to implement a corresponding multiple sweep evolution using the basic approach, is given by

$$l\ [\text{size(fat)} + \text{size(thin)}]\,. \tag{4.105}$$

The computational cost of the basic approach is $\mathcal{O}(l)$. Thus, to estimate the computational cost of nested reversal schedules, it is necessary to prove the following theorem.

Theorem 4.3 (Run-time Requirement Growth) *Consider an evolution* $\mathcal{E}_{3\times l}$ *traversing* l *time steps in three alternative sweeps.* $\vec{\mathbf{t}}_{\mathbf{3}\times\mathbf{l}} = (t, \bar{t}, \bar{\bar{t}})^\top$ *and* $\vec{\mathbf{d}}_{\mathbf{3}\times\mathbf{l}} = (d, \bar{d}, \bar{\bar{d}})^\top$ *are the 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* $\bar{x}_i$, $0 \leq i \leq l$. *Assume that* c *thin checkpoints can be stored in place of a single fat one, i.e.* $\bar{d} = c\,d$. *Let* $\mathbf{S}_h(l, C)$ *be an admissible nested reversal schedule.* $\mathbf{S}_h(\vec{\mathbf{t}}_{\mathbf{3}\times\mathbf{l}}, l, C)$ *is constructed, using the heuristic in Section 4.2.4 for the given parameters* $\vec{\mathbf{t}}_{\mathbf{3}\times\mathbf{l}}$, $\vec{\mathbf{d}}_{\mathbf{3}\times\mathbf{l}}$, *and* C. *Then, the growth of the run-time requirement* $\mathbf{T}_h(l, C)$ *for* $\mathbf{S}_h(l, C)$ *with respect to the minimal run-time requirement* $\mathbf{T}_{min}(l, l)$ *of the basic approach is to estimate as follows*

$$\frac{\mathbf{T}_h(l, C)}{\mathbf{T}_{min}(l, l)} = \mathcal{O}\left((\ln l)^2\right). \tag{4.106}$$

Proof:

To prove the relation (4.106), it is firstly required to evaluate an upper bound $\mathbf{U}_h(l, C)$ of the evaluation cost $\mathbf{T}_h(l, C)$, secondly to prove the equation

$$\frac{\mathbf{U}_h(l, C)}{\mathbf{T}_{min}(l, l)} = \mathcal{O}\left((\ln l)^2\right). \tag{4.107}$$

To evaluate the upper bound $\mathbf{U}_h(l, C)$, consider a binomial nested reversal schedule $\mathbf{S}^n_{bin}(l, C)$, defined by

$\mathbf{S}^n_{bin}(l, C)$ $\equiv$ an admissible nested reversal schedule, which can be applied to the implementation of a multiple sweep evolution $\mathcal{E}_{3\times l}$, step cost distribution $\vec{\mathbf{t}}_{\mathbf{3}\times\mathbf{l}}$, size distribution $\vec{\mathbf{d}}_{\mathbf{3}\times\mathbf{l}}$, and C available fat checkpoints, so that during the implementation of $\mathbf{S}^n_{bin}(l, C)$ fat checkpoints are placed using the binomial rule, thin checkpoints are placed using the heuristic (see Section 4.2.4).

Assume that $l = l_{max}(r, C+1)$ is the maximal number of steps, which can be reversed using up to $(C+1)$ checkpoints, so that each step is evaluated up to r times during this reversal (see Section 2.1.7). Initial thin and fat checkpoint distributions of a binomial nested reversal schedule $\mathbf{S}^n_{bin}(l, C)$ are illustrated in Figure 4.20. Obviously, the binomial nested reversal schedule $\mathbf{S}^n_{bin}(l, C)$ is

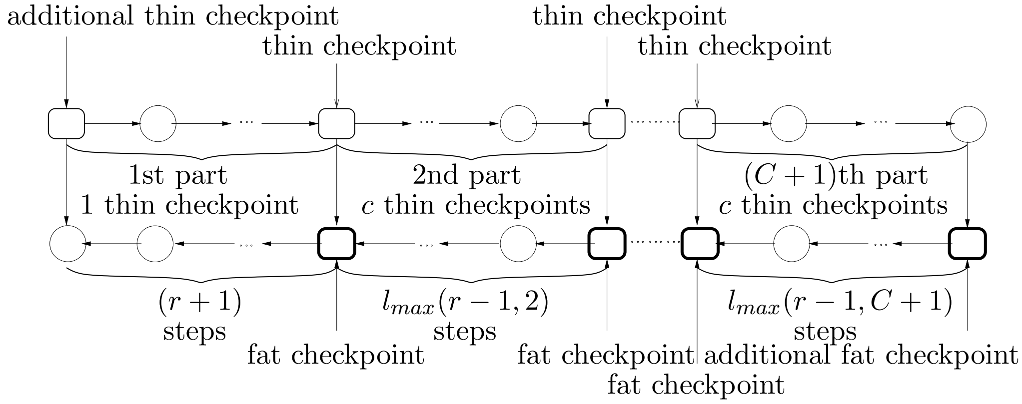

Figure 4.20: Partition of the original and the adjoint sweeps of a multiple sweep evolution $\mathcal{E}_{3\times l}$ using the binomial rule in Section 2.2.1 with $l = l_{max}(r, C+1)$

a special case of a nested reversal schedule $\mathbf{S}_h(l, C)$, constructed using the heuristic in Section 4.2.4. Therefore, for the evaluation cost $\mathbf{T}^n_{bin}(l, C)$ of the binomial nested reversal schedule $\mathbf{S}^n_{bin}(l, C)$ the following relation is satisfied

$$\mathbf{T}_h(l, C) \leq \mathbf{T}^n_{bin}(l, C). \tag{4.108}$$

Thus, the evaluation cost $\mathbf{T}^n_{bin}(l, C)$ of the binomial nested reversal schedule $\mathbf{S}^n_{bin}(l, C)$ can be used as an upper bound $\mathbf{U}_h(l, C)$ for the evaluation cost $\mathbf{T}_h(l, C)$. The evaluation cost $\mathbf{T}^n_{bin}(l, C)$ can be computed by

$$\mathbf{T}^n_{bin}(l, C) = \mathbf{T}^f_{bin}(l, C)t + \mathbf{T}^a_{bin}(l, C)\bar{t} + l\bar{\bar{t}}, \tag{4.109}$$

with

$\mathbf{T}^f_{bin}(l, C)$	being the overall number of original steps F_i, needed during the execution of $\mathbf{S}^n_{bin}(l, C)$,
$\mathbf{T}^a_{bin}(l, C)$	being the overall number of adjoint steps $\bar{F}_i$, needed during the execution of $\mathbf{S}^n_{bin}(l, C)$.

According to the construction of binomial nested reversal schedules, the overall number $\mathbf{T}^a_{bin}(l, C)$ of adjoint steps can be evaluated by

$$\mathbf{T}^a_{bin}(l, C) = \mathbf{T}_{bin}(l, C+1), \tag{4.110}$$

with $\mathbf{T}_{bin}(l, C+1)$ being the minimal evaluation cost, needed for the reversal of l steps with up to $C+1$ checkpoints available (see Chapter 2).

Since $\mathbf{T}_{bin}(l, C+1) \leq r(l, C+1)l$, each adjoint step is evaluated up to $r(l, C+1) = r_{bin}(l, C+1)$ times. Figure 4.20 indicates that an upper bound for the number of forward steps, needed to evaluate an adjoint step is

$$\max\{r(l, C+1), r(m, c)\},$$

with $m = l_{max}\left(r(l, C+1) - 1, C+1\right)$. $r(l, C+1)$ corresponds to the number of forward steps, needed to perform for evaluating an adjoint step from the 1st part. This part contains $r(l, C+1) + 1$ steps, so that 1 thin checkpoint is available. $r(m, c)$ corresponds to the number of forward steps, needed to evaluate an adjoint step from the $(C+1)$th part. This part contains $m = l_{max}\left(r(l, C+1) - 1, C+1\right)$ steps, so that c thin checkpoints are available.

Since $m < l$, the following relation is satisfied

$$\mathbf{T}^{f}_{bin}(l, C) \leq r(l, C+1) \max\left\{r(l, C+1), r(l, c)\right\} l. \tag{4.111}$$

Then, summarising (4.109), (4.110), and (4.111), we obtain

$$\begin{aligned} \mathbf{T}^{n}_{bin}(l, C) &\leq r(l, C+1) \max\left\{r(l, C+1), r(l, c)\right\} lt + \\ &+ r(l, C+1) l\bar{t} + l\bar{\bar{t}}. \end{aligned} \tag{4.112}$$

Using the definition of the minimal evaluation cost $\mathbf{T}_{min}(l, l)$, we obtain

$$\mathbf{T}_{min}(l, l) = l(t + \bar{t} + \bar{\bar{t}}). \tag{4.113}$$

Therefore, the ratio $\mathbf{T}^{n}_{bin}(l, C)/\mathbf{T}_{min}(l, l)$ can be estimated by

$$\begin{aligned} \frac{\mathbf{T}^{n}_{bin}(l, C)}{\mathbf{T}_{min}(l, l)} &\leq \frac{r(l, C+1) \max\left\{r(l, C+1), r(l, c)\right\} t}{t + \bar{t} + \bar{\bar{t}}} + \frac{r(l, C+1)\bar{t} + \bar{\bar{t}}}{t + \bar{t} + \bar{\bar{t}}} \leq \\ &\leq r(l, C+1) \max\left\{r(l, C+1), r(l, c)\right\} K + \mathcal{O}\left(r(l, C+1)\right) \leq \\ &\leq \begin{cases} r(l, C+1)^2 K + \mathcal{O}\left(r(l, C+1)\right), & \text{for } c \geq C+1, \\ r(l, C+1) r(l, c) K + \mathcal{O}\left(r(l, C+1)\right), & \text{for } c < C+1, \end{cases} \end{aligned} \tag{4.114}$$

with K being a constant ($K = \frac{t}{t + \bar{t} + \bar{\bar{t}}}$).

Recall Figure 2.10 (Chapter 2) in the case $r(l, C+1) \sim (C+1)$, the relative reversal cost $r(l, C+1)$, as well as the number $(C+1)$ of checkpoints available, grow in the same way as the logarithm $\ln l$ of the length l; for C fixed $r(l, C+1) = \mathcal{O}\left(l^{1/C}\right)$. Moreover, if c is fixed, then, $r(l, c) = \mathcal{O}\left(l^{1/c}\right)$. If $c \sim r(l, c)$, then, the relative reversal cost $r(l, c)$, as well as the number c of thin checkpoints available, grow in the same way as the logarithm $\ln l$ of the length l.

Figure 4.21 illustrates the growth of the run-time requirement for different r.

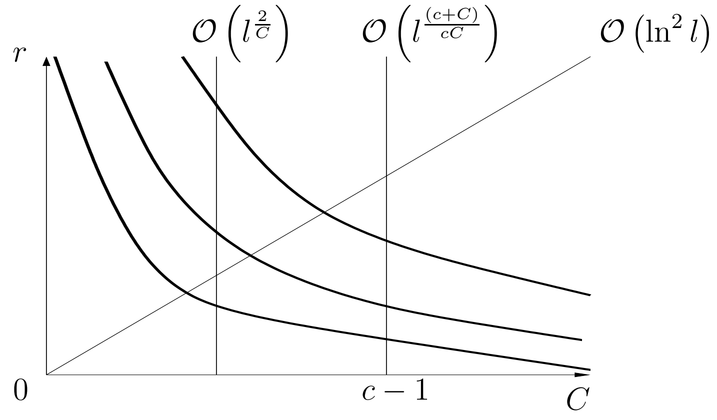

Figure 4.21: Growth of the run-time requirement

Therefore,

$$\frac{\mathbf{T}_h(l,C)}{\mathbf{T}_{min}(l,l)} \leq \frac{\mathbf{U}_h(l,C)}{\mathbf{T}_{min}(l,l)} = \mathcal{O}\left(\ln^2 l\right), \tag{4.115}$$

with $\mathbf{U}_h(l,C) = \mathbf{T}^n_{bin}(l,C)$, $r(l,C+1) \sim (C+1)$, and $r(l,c) \sim c$. Thus, the relation (4.106), as well as Theorem 4.3, are proved.

□

Theorem 4.3 implies the following powerful conclusion: when the dependence on number of time steps is arranged polylogarithmically by nested checkpoint strategies, namely

$$\frac{\text{Reversal Space}}{\text{Forward Space}} = \mathcal{O}(\ln^2 l), \tag{4.116}$$

then, the operations count, needed to implement a Newton step, also grows as a second power of $\ln l$, i.e.

$$\frac{\text{Reversal Time}}{\text{Forward Time}} = \mathcal{O}(\ln^2 l). \tag{4.117}$$

This is because, using the equation (4.104), the storage requirement, needed to implement a corresponding nested reversal schedule, is given by

$$\text{Reversal Space} = (Cc + c + 1)\ \text{size(thin)}.$$

The storage requirement for a forward simulation is given by

$$\text{Forward Space} = \text{size(thin)}.$$

This is because to implement a forward simulation a space for a single original state is needed, which size corresponds to the size of a thin checkpoint. Therefore,

$$\frac{\text{Reversal Space}}{\text{Forward Space}} = Cc + c + 1 = \mathcal{O}(\ln^2 l),$$

if $r(l, C+1) \sim C+1$ and $r(l, c) \sim c$. Moreover, run-time requirement for a forward simulation is given by

$$\text{Forward Time} = \mathcal{O}(l).$$

Run-time requirement, needed to implement a corresponding nested reversal schedule, is given by

$$\text{Reversal Time} = \mathbf{T}_h(l, C) \leq \mathbf{U}_h(l, C) = \mathcal{O}(l \ln^2 l),$$

if $r(l, C+1) \sim C+1$ and $r(l, c) \sim c$. This is because of Theorem 4.3. Therefore,

$$\frac{\text{Reversal Time}}{\text{Forward Time}} = \mathcal{O}(\ln^2 l).$$

Combining similar results in [35, 38], concerning single reversals, we can construct the following Table 4.1. Left column of Table 4.1 corresponds to

Single Reversal (Gradient) $r(l, s) \sim s$	Double Reversal (Newton Step) $r(l, C+1) \sim C+1$ and $r(l, c) \sim c$
$\frac{\text{Reversal Space}}{\text{Forward Space}} = \mathcal{O}(\ln l)$	$\frac{\text{Reversal Space}}{\text{Forward Space}} = \mathcal{O}(\ln^2 l)$
$\frac{\text{Reversal Time}}{\text{Forward Time}} = \mathcal{O}(\ln l)$	$\frac{\text{Reversal Time}}{\text{Forward Time}} = \mathcal{O}(\ln^2 l)$

Table 4.1: Temporal and spatial complexity for single and double reversal

the single reversal case, which produces a gradient for instance. If $r(l, s) = r_{bin}(l, s) \sim s$, then, the growth of the spatial and the temporal complexities, needed for the single reversal, can be estimated as shown in the left column of Table 4.1. Right column of Table 4.1 corresponds to the double reversal case, which produces a Newton step for instance. If $r(l, C+1) = r_{bin}(l, C+1) \sim C+1$ and $r(l, c) = r_{bin}(l, c) \sim c$, then, the growth of the spatial and the temporal complexities, needed for the double reversal, can be estimated as shown in the right column of Table 4.1. This proves that the dependence on l can be arranged polylogarithmically by nested checkpoint strategies. Consequently, the operations count also grows as a second power of $\ln l$.

4.3 Classification of Available Software

In this section a survey of software available to solve BVPs and optimal control problems is given. These software are developed based on theoretical investigations in these thesis.

4.3.1 Double Sweep Algorithms

Consider algorithms, discussed in Chapter 3, each iteration of which requires a double sweep through the discretised time interval. They are coded using the C++ programming language environment. For the memory efficient implementation of these algorithms the routine **revolve** from [38] is applied. Figure 4.22 gives an overview which routines have to be utilised to solve an optimal control problem or, consequently, a BVP resulting from its first order conditions, using a corresponding method. Moreover, Figure 4.22 gives an overview which routines have to be utilised to solve any arbitrary decoupled BVP using a corresponding method.

The scheme in Figure 4.22 consists of three rows. In order to implement an example for a BVP or an example for an optimal control problem, exactly one file from each row has to be chosen, so that there are connections between these files. Thus, if we have to solve an example of a BVP, we have firstly to chose the file **bvp** from the first row in Figure 4.22. This file contains a code for a corresponding BVP. There are three files which are connected with the **bvp** in the second row, each of which contains a code of a corresponding method, i.e. the compactification method (**compact**), the Riccati method (**riccati**), and the reduced compactification method or the stabilised march (**red-cmpt**). One of these files from the second line has to be chosen. As we can see, each file from the second row is connected with the file **revolve** from the third row. This can be used to produce binomial reversal schedules (see Chapter 2) and to control the reversal process for a given forward simulation.

In the case if we want to solve an optimal control problem, we should chose the file **control** from the first row, which contains a code for the corresponding example. Moreover, this file is connected with a single file from the second

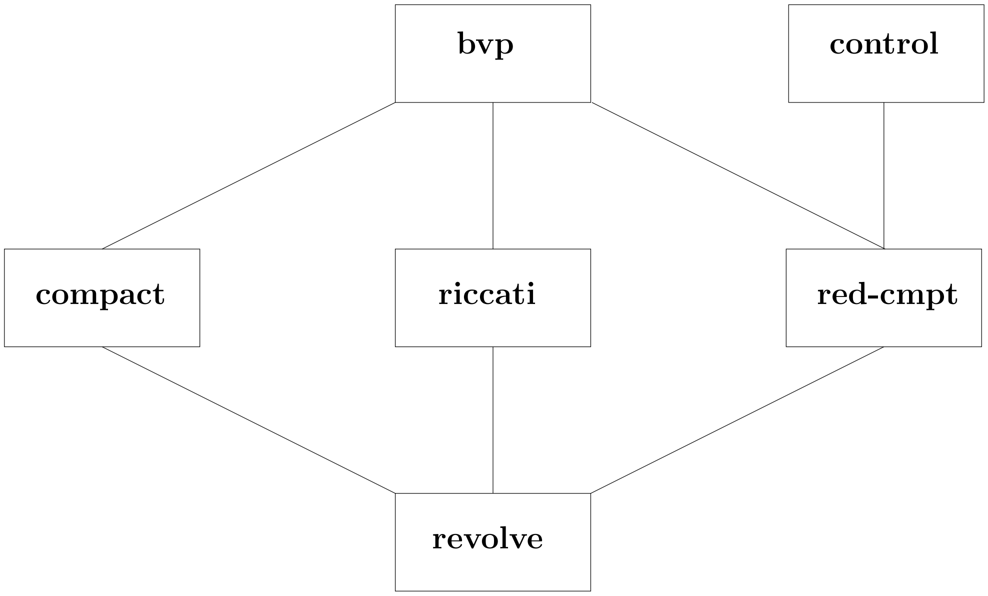

Figure 4.22: Software available for the memory reduced implementation of double sweep algorithms

row, namely, with the file **red-cmpt**. Thus, we can use the reduced compactification method or the stabilised march algorithm to solve an optimal control problem. Finally, we have to link the file **revolve** from the third row for the memory reduced implementation of the stabilised march algorithm.

4.3.2 Triple Sweep Algorithms

Consider algorithms, discussed in Chapter 3, each iteration of which requires a triple sweep through the discretised time interval. They are coded using the C++ programming language environment. For the memory efficient implementation of these algorithms the routine **nested-revolve** is applied. This is also coded using the C++ programming language environment. Figure 4.23 gives an overview which routines have to be utilised to solve an optimal control problem or, consequently, a BVP resulting from its first order conditions, using a corresponding method.

The scheme in Figure 4.23 consists of four rows. In order to implement an example for an optimal control problem, exactly one file from each row has to be chosen, so that there are connections between these files. Thus, if we have to solve an example for an optimal control problem, we should chose the file **control** from the first row in Figure 4.23. This file contains a code for a

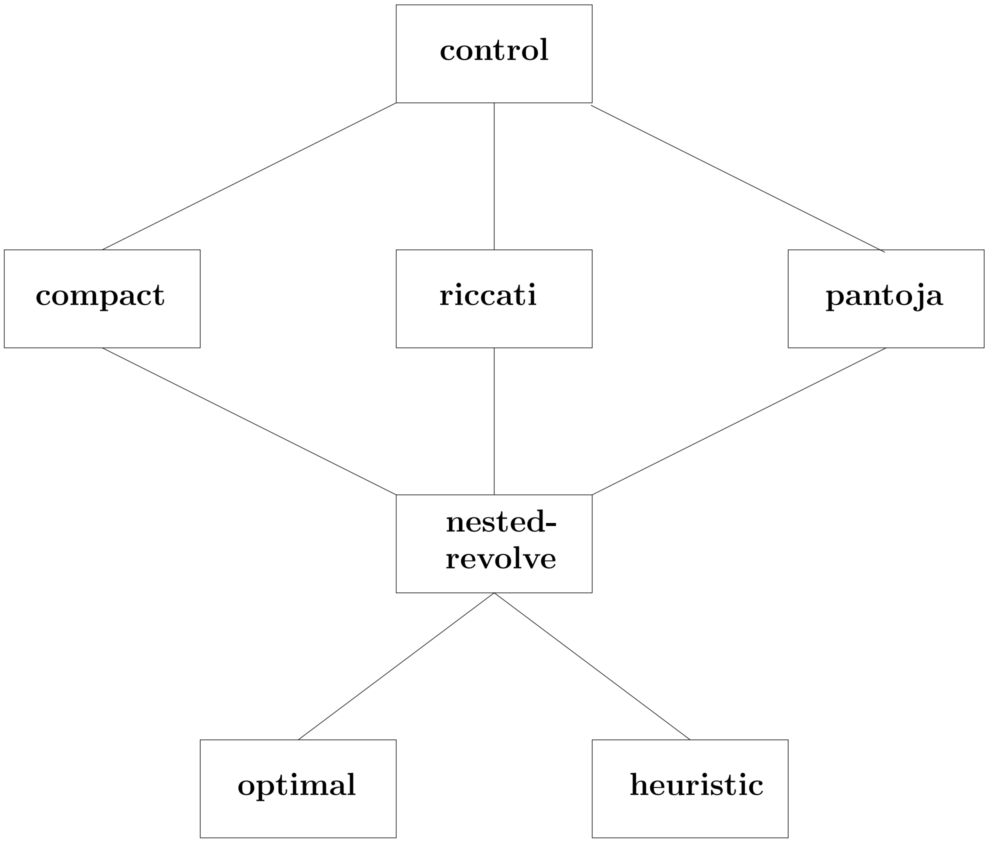

Figure 4.23: Software available for the memory reduced implementation of triple sweep algorithms

corresponding example. There are three files which are connected with the **control** in the second row, each of which contains a code of a corresponding method, i.e. the compactification method (**compact**), the Riccati method (**riccati**), and the Pantoja method (**pantoja**). One of these files from the second line has to be chosen. As we can see, each file from the second row is connected with the file **nested-revolve** from the third row. This can be used to control the implementation process for a given three sweeps evolution. The file **nested-revolve** is connected with both of two files from the fourth row in this scheme. These files contain codes for corresponding nested reversal schedules, which can be used for the memory efficient implementation of a triple sweep evolution. We can select the file **optimal** if we want to utilise optimal nested reversal schedules introduced in Section 4.2.3. We can also select the file **heuristic** if we want to utilise nested reversal schedules produced by the heuristic in Section 4.2.4.

4.4 Conclusions

Throughout this chapter evolutions containing more than two alternative sweeps are considered. Particularly, for evolutions containing three sweeps and a fixed number of checkpoints available throughout its implementation, it was possible to establish optimal nested reversal schedules. The schedules are designed to minimise the overall execution time given a certain total amount of storage for the checkpoints. While a closed form solution for this discrete optimisation problem have not been found, cheap heuristics for constructing nested reversals that are quite close to optimality have been developed. It was demonstrated that the dependence on l can be arranged polylogarithmically [34] by nested checkpoint strategies. Consequently, the operations count also grows as a second power of $\ln l$, which needs not result in an increase of the actual run-time due to memory effects.

Chapter 5

Numerical Examples

The following example is the first example from the set of optimal control problems given in [27].

5.1 Example 1

Consider the following optimal control problem
(5.1)

$$\min_{\mathbf{x},\mathbf{u}} J(\mathbf{x},\mathbf{u}) = \sum_{i=1}^{l-1}\left(\sum_{j=1}^{n}\left(\mathbf{x}_j^i + \frac{1}{4}\right)^4 + \sum_{j=1}^{m}\left(\mathbf{u}_j^i + \frac{1}{2}\right)^4\right) + \sum_{j=1}^{n}\left(\mathbf{x}_j^l + \frac{1}{4}\right)^4,$$

where the system is described by

$$\mathbf{x}^{i+1} = T_i(\mathbf{x}^i, \mathbf{u}^i), \qquad i = 1, ..., l-1. \tag{5.2}$$

with

$$T_i(\mathbf{x}^i, \mathbf{u}^i) = A\mathbf{x}^i + B\mathbf{u}^i + (\mathbf{x}^{i^\top} C\mathbf{u}^i)e, \qquad i = 1, ..., l-1, \tag{5.3}$$

$\mathbf{x}^1 = 0$, and $A \in \mathbf{R}^{n\times n}$ given by:

$$(A)_{i,j} = \begin{cases} 0.5 & if \quad i = j, \\ 0.25 & if \quad i = j+1, \\ -0.25 & if \quad i = j-1. \end{cases} \tag{5.4}$$

$B \in \mathbf{R}^{n\times m}$ given by:

$$(B)_{i,j} = \frac{i+j}{n+m}; \tag{5.5}$$

$C \in \mathbf{R}^{n\times m}$ given by:

$$(C)_{i,j} = \mu \frac{i+j}{n+m}; \tag{5.6}$$

e is the all-one vector in $\mathbf{R}^n$.

For this problem we take $n = 2$, $m = 1$, and $\mu = 0, 1/200, 1/20, 1/2, 1$. The larger μ, the more nonlinear the transformation functions. For each μ we choose $l = 50$ and $l = 100$. The initial point is $\mathbf{u}^0 = 0$. The numerical results are presented in Table 5.1 for $\mu = 1$ and $l = 100$. Pantoja Algorithm 3.8 from Section 3.5.4 takes 10 iterations and stops with $\Lambda(10) = 0.0$, so that $\Lambda(i) = \frac{\|\delta \mathbf{u}^i\|}{\|\delta \mathbf{u}^i + \mathbf{u}^i\|}$. The resulting trajectory $\mathbf{x}_1^{10}$, $\mathbf{x}_2^{10}$ and the optimal control

Table 5.1: Program data for Example 1

iter	$J(\mathbf{u}^i)$	$\|\nabla J(\mathbf{u}^i)\|_2$	$\nabla J(\mathbf{u}^i)\Delta\mathbf{u}^i$	$\Lambda(i)$
0	7.039062	6.826840	−0.000000	0.000000
1	2.843369	2.910754	−6.347026	3.012546
2	0.712435	0.982059	−3.426086	1.264369
3	0.152004	0.296074	−0.927362	0.448092
4	0.037062	0.087618	−0.190987	0.211389
5	0.014595	0.025244	−0.037497	0.115788
6	0.010626	0.006470	−0.006749	0.064842
7	0.010149	0.001096	−0.000855	0.030111
8	0.010130	0.000056	−0.000036	0.007348
9	0.010130	0.000000	−0.000000	0.000413
10	0.010130	0.000000	−0.000000	0.000000

$\mathbf{u}_1^{10}$ are depicted in Figure 5.1. For the memory efficient implementation of this example nested reversal schedules are used. Therefore, the memory requirement is reduced. Naturally, the run-time requirement can increase. Figure 5.2 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.

Blue line shows run-time ratios if only three checkpoints are available. 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 shows run-time ratios if the number of checkpoints available is evaluated by the natural logarithm of time-steps. In Figure 5.2 we can observe, that for a reasonable number of time-steps, run-times ratios are nearly one. Therefore, for this example the utilisation of nested checkpointing techniques does not result in an increase of run-time at all.

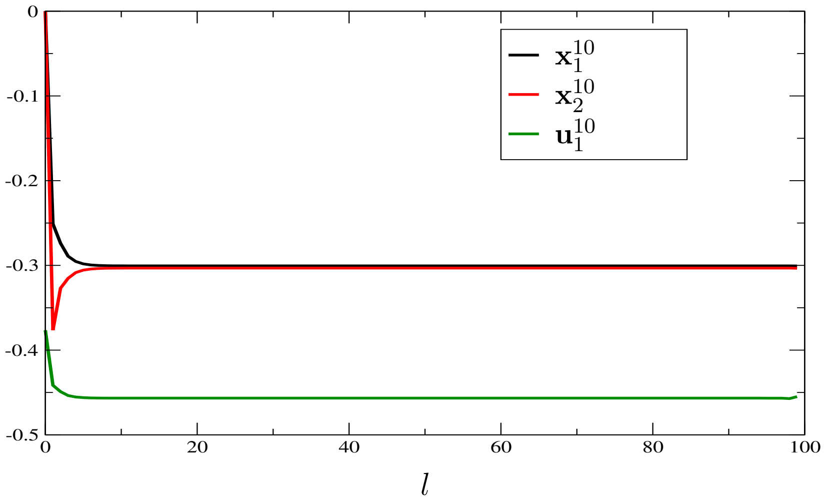

Figure 5.1: Resulting trajectory $\mathbf{x}^{10}$ and optimal control $\mathbf{u}^{10}$

Run-time behaviours for other examples from the set of optimal control problems given in [27] are similar to Figure 5.2. Therefore, in the following the presentation of all examples from this set is omitted and an optimal control problem of laser surface hardening of steel is considered.

5.2 Laser Surface Hardening of Steel

5.2.1 Introduction

We consider a control problem which describes the laser surface hardening of steel. Therefore, the following section closely follows the description in [52].

The mode of operation of this process is depicted in Fig. 5.3. A laser beam

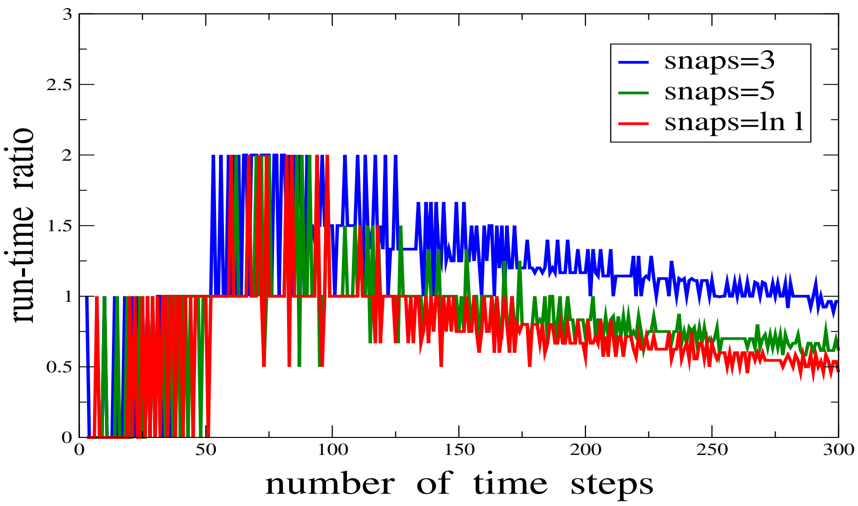

Figure 5.2: Run-time & Memory Trade-off

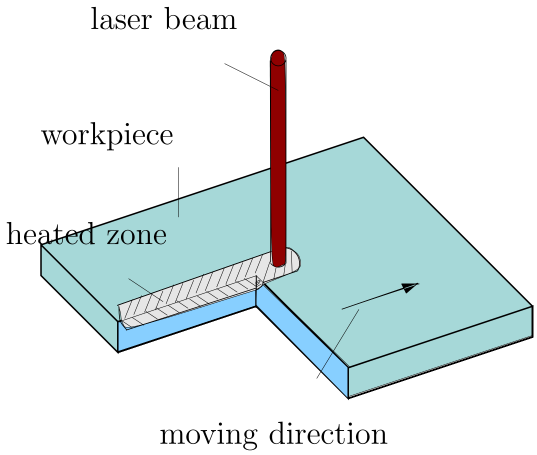

Figure 5.3: Sketch of a laser hardening process

moves along the surface of a workpiece, creating a heated zone around its trace. The heating process is accompanied by a phase transition, in which the high temperature phase in steel, called austenite, is produced.

A simple model to describe the evolution of austenite is given by the following rate law

$$a(0) = 0, \tag{5.7}$$

$$a_t = \frac{1}{\tau(\theta)}[a_{eq}(\theta) - a]_+, \tag{5.8}$$

where a is the volume fraction of austenite, θ the temperature, τ a time constant, and $[x]_+ = \max\{x, 0\}$ is the positive part function. The equilibrium volume fraction a_{eq} is monotone and satisfies $a_{eq}(x) = 0$ for $x < A_s$ and $a_{eq}(x) = 1$ for $x > A_f$, where A_s and A_f are the threshold temperatures for the beginning and the end of transformation. Hence, the austenite volume fraction increases during heating until it reaches some value $\tilde{a} \leq 1$. During cooling we have $a_t = 0$, and the value $\tilde{a}$ is kept.

Since one usually tries to keep the moving velocity of the laser beam constant, the most important control parameter is the laser energy. Whenever the temperature in the heated zone exceeds the melting temperature of steel, the work-piece quality is destroyed. Therefore, the goal of surface hardening is to achieve a desired hardening zone, in our case 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 $J(\mathbf{u})$ defined by

$$J(\mathbf{u}) = \frac{\beta_1}{2}\int_\Omega (a(x,T) - a_d(x))^2 dx + \frac{\beta_2}{2}\int_0^T \int_\Omega [\theta - \theta_m]_+^2 dx\, dt + \frac{\beta_3}{2}\int_0^T \mathbf{u}^2 dt, \tag{5.9}$$

where $\mathbf{u}$ is the laser energy and β_i, $i = 1, 2, 3$, are positive constants. The second term in (5.9) penalises temperatures above the melting temperature θ_m.

5.2.2 Problem Statement and Physical Data

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 following system of state equations (5.10) - (5.14) consists of a semi-linear heat equation coupled with the initial-value problem for the phase transitions:

$$
\begin{aligned}
&(5.10)\quad a_t = f(\theta, a) &=& \frac{1}{\tau(\theta)}(a_{eq}(\theta) - a)\mathcal{H}(a_{eq}(\theta) - a), && in\ Q,\\
&(5.11)\quad a(0) &=& 0, && in\ \Omega,\\
&(5.12)\quad \rho c_\rho \theta_t - k\triangle\theta &=& -\rho L a_t + \mathbf{u}\alpha, && in\ Q,\\
&(5.13)\quad \frac{\partial\theta}{\partial\nu} &=& 0, && on\ \Sigma,\\
&(5.14)\quad \theta(0) &=& \theta_0, && in\ \Omega.
\end{aligned}
$$

Since the main cooling effect is the self-cooling of the workpiece, we have assumed homogeneous Neumann conditions on the boundary. The term $-\rho L a_t$ describes the consumption of latent heat due to the phase transition. The term $\mathbf{u}(t)\alpha(x,t)$ is the volumetric heat source due to laser radiation, where the laser energy $\mathbf{u}(t)$ 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. Moreover, we assume

(A1) $a_{eq}(x) \in [0,1]$ for all $x \in \mathbb{R}$, $\|a_{eq}\|_{C^2(\mathbb{R})} \leq c$;

(A2) $0 < \underline{\tau} \leq \tau(x) \leq \overline{\tau}$ for all $x \in \mathbb{R}$, $\|\tau\|_{C^2(\mathbb{R})} \leq c$;

(A3) $\mathcal{H} \in C^{2,1}(\mathbb{R})$, monotone approximation of the Heaviside function satisfying $\mathcal{H}(x) = 0$ for $x \leq 0$;

(A4) $\theta_0 \in H^1(\Omega)$;

(A5) $\alpha \in L^\infty(0,T;\ L^\infty(\Omega))$;

(A6) $\mathbf{u} \in L^2(0,T)$;

(A7) $a_d \in L^2(\Omega)$, $0 \leq a_d \leq 1$ a.e. in Ω.

Since $\mathcal{H}$ is a regularised Heaviside function, the term $x\mathcal{H}(x)$ is a regularisation of the positive part function $[x]_+$.

We study the following state and control constrained optimal control problem with the cost functional $J(\mathbf{u})$ defined in (5.9):

$$(5.15)\qquad \min J(\mathbf{u}),\ s.t.\ (\theta, a, \mathbf{u})\ solves\ (5.10) - (5.14).$$

The physical parameters for the heat equations are given by

$$(5.16)\quad \rho c_\rho = 4.9\frac{J}{cm^3K}, \qquad k = 0.64\frac{J}{cmKs}, \qquad and \qquad \rho L = 627.9\frac{J}{cm^3}.$$

	$\theta = 729.9$	$\theta = 730$	$\theta = 830$	$\theta = 840$	$\theta = 900$	$\theta = 900.1$
$a_{eq}(\theta)$	0	0	0.91	1	1	1
$\tau(\theta)$	1	1	0.2	0.18	0.05	0.05

Table 5.2: Pointwise data for a_{eq} and τ

The equilibrium volume fraction a_{eq} and the function τ are cubic spline functions interpolating the pointwise data presented in Table 5.2. Thus, (A1) and (A2) are satisfied. For the monotone regularisation of the Heaviside function, we take

$$\mathcal{H} = \begin{cases} 1, & for\ s \geq \delta, \\ \frac{35}{32} * \tilde{s} - \frac{35}{32} * \tilde{s}^3 + \frac{21}{32} * \tilde{s}^5 - \frac{5}{32} * \tilde{s}^7 + \frac{1}{2}, & for\ \delta > s \geq 0, \\ 0, & for\ s < 0, \end{cases} \tag{5.17}$$

where $\tilde{s} = \frac{2s}{\delta} - 1$ and $\delta = 0.15$. In particular, (A3) holds. In contrast to the regularisation of the Heaviside function in [52], we preferred the regularisation up to the seventh order to prevent unsmoothnesses in the resulting control function $\mathbf{u}$.

The initial condition for the θ-variable is the room temperature, i.e. $\theta_0 = 20$, and we choose $\theta_m = 1400$ the melting temperature of steel. Thus, (A4) is satisfied.

We take a $2.8kW$ laser, and the laser radiation profile $\alpha = \alpha(x, y, t)$ is given by

$$\alpha(x, y, t) = \frac{4kA}{\pi D^2} exp\left(-\frac{2(x - vt)^2}{D^2}\right) exp(ky), \tag{5.18}$$

where $D = 0.47$cm, $k = 60$ 1/cm, $A = 0.3$, and $v = 1$ cm/s. Notice that α satisfies (A5).

5.2.3 Numerical Simulation

Let $\mathbf{u}$ be arbitrary. Throughout, we denote (θ, a) the unique solution to the state equations (5.10) - (5.14).

For given $l \in \mathbb{N}$, let

$$0 = t_0 < t_1 < \ldots < t_l = T$$

be an equidistant grid in the interval $[0, T]$ with time step $\Delta t = T/l$. Since we have no chance to get exact solution for a chosen laser energy at some given time instances, we compute a discrete solution of (5.10) - (5.14) on a fine grid. For that purpose, we introduce piecewise linear finite elements $\{\varphi_1, ..., \varphi_n\} \subset H^1(\Omega)$ and denote by $x_1, ..., x_n \in \bar{\Omega}$ the finite-element (FE) nodes, so that $\varphi_j(x_i) = \delta_{ij}$ for $1 \leq i, j \leq n$. The FE solution $\{(\theta_n^j, a_n^j)\}_{j=0}^l$ to (5.10) - (5.14) is obtained by the following scheme. Find (θ_n^0, a_n^0) from

$$(5.19) \quad a_n^0 = 0 \qquad and \qquad (\theta_n^0, \varphi_i)_{L^2(\Omega)} = (\theta_0, \varphi_i)_{L^2(\Omega)}, \qquad for\ i = 1, ..., n.$$

Then, for $j = 1, ..., l$, solve

$$\begin{aligned} \rho c_\rho \left(\bar{\partial}_\tau \theta_n^j, \varphi_i\right)_{L^2(\Omega)} + k \left(\nabla \theta_n^j, \nabla \varphi_i\right)_{L^2(\Omega)} &= (\mathbf{u}(t_j)\alpha(t_j) - \\ (5.20) \qquad\qquad &- \rho L f\left(\theta_n^{j-1}(x_i), a_n^{j-1}(x_i)\right), \varphi_i\Big)_{L^2(\Omega)}, \\ \text{for } i &= 1, ..., n, \end{aligned}$$

where $\bar{\partial}_\tau \theta_n^j = \left(\theta_n^j - \theta_n^{j-1}\right)/\Delta t$, and use θ_n^{j-1} to compute a_n^j from

$$(5.21) \qquad \bar{\partial}_\tau a_n^j(x_i) = f\left(\theta_n^{j-1}(x_i), a_n^j(x_i)\right), \qquad for\ i = 1, ..., n.$$

In this case, the system of ordinary differential equations has dimension $2n$. The computational domain $\Omega = [0, 5] \times [-1, 0]$ is the cross-section of the plate depicted in Figure 5.3.

Choosing $T = 5.25$ and the laser energy $\mathbf{u} = 600$, we compute the finite-element (FE) solution of (5.19) - (5.21), where we use multigrid methods to solve the linear system (5.20) at each time level. The FE triangulation of Ω is done by a nonuniform mesh with $n = 231$ degrees of freedom. For the time grids, we take $l = 200$. Therefore, $\Delta t = 5.25/200 = 0.02625$.

The laser beam moves along the x-axis. Figure 5.4 shows the evolution of temperature θ and austenite volume fraction a at the point $(x, y) = (2.5, 0)$ on the surface. During the heating stage, austenite starts to grow until it reaches a volume fraction 100%. Owing to the Heaviside-function in (5.10), it does not decrease with decreasing temperature. Figure 5.5 shows the FE solution for the temperature θ at $t = T$. Figure 5.9 shows the austenite volume fraction at the end-time T. The hardening depth increases until it reaches 1mm. At the end of the plate, heat can only diffuse in one direction. Hence, we obtain the typical thickening strip at the end, which is also observed in experiments.

5.2.4 Optimal Control

The optimal control is unknown. Therefore, the optimal control problem (5.15) is solved using Pantoja Algorithm 3.8 from Section 3.5.4. Nested

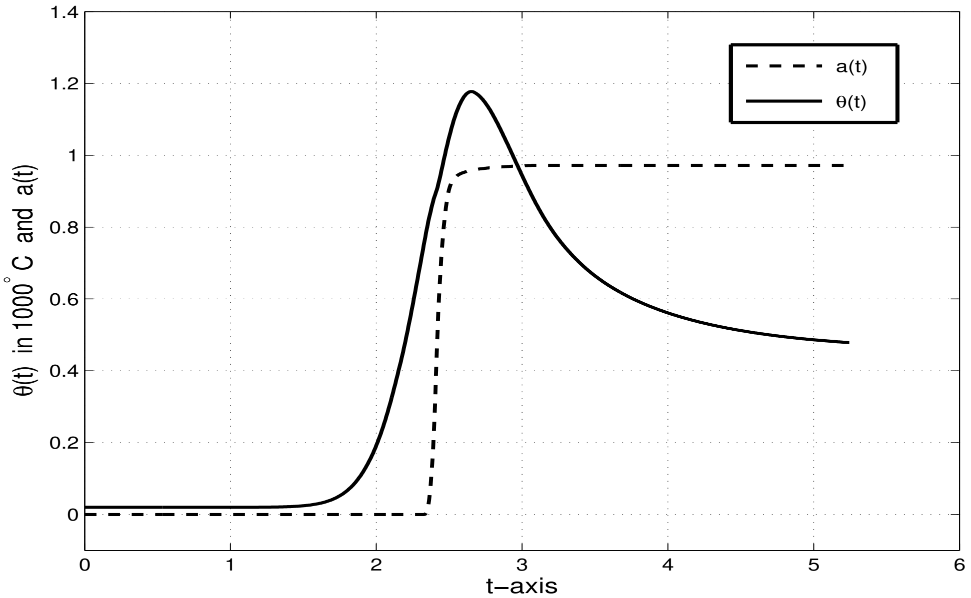

Figure 5.4: The time evolution of the temperature and austenite at $(x, y) = (2.5, 0)$

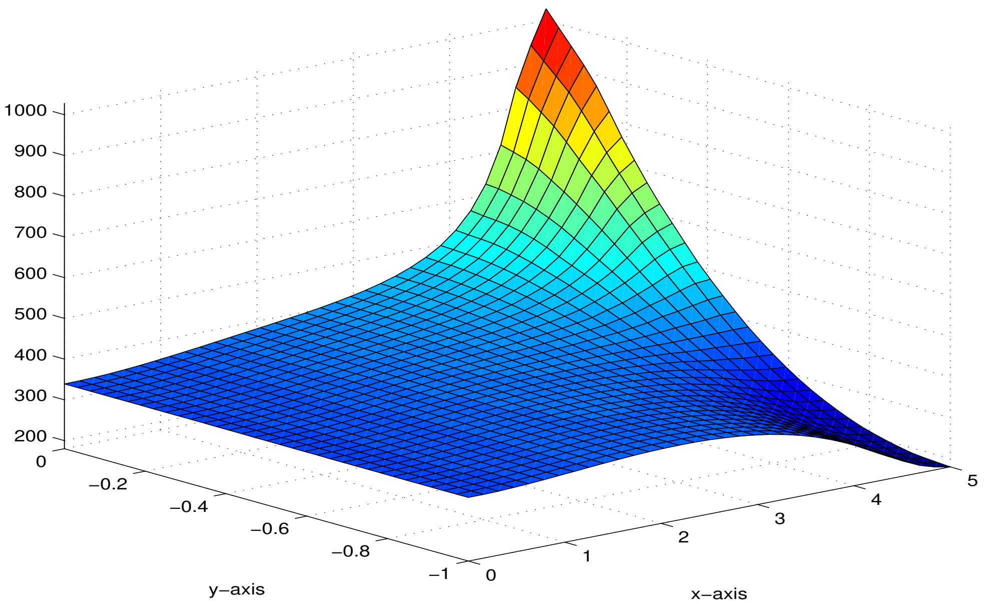

Figure 5.5: FE snapshot for the temperature at $t = T$

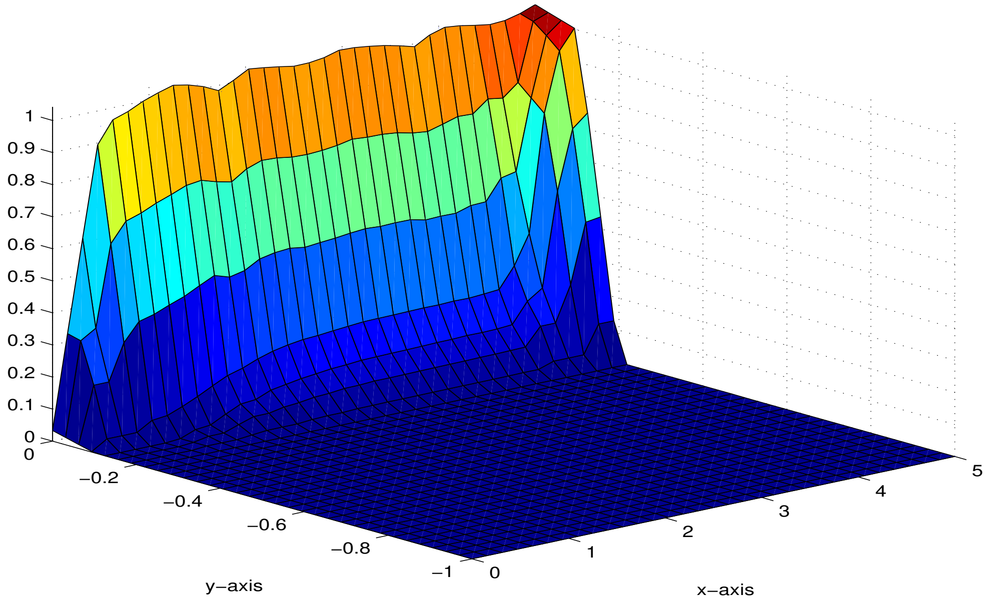

Figure 5.6: The volume fraction of austenite at $t = T$

reversal schedules are utilised for the reduction of memory requirement during the implementation of Algorithm 3.8 applied to the optimal control problem (5.15).

First, we discretise the problem (5.15) as it was shown in Section 5.2.3 and write this in the standard form as follows

$$\min J(\mathbf{u}, a^l, \theta^l, v^l) = \frac{\beta_1}{2} \sum_{i=1}^{n} (a_i^l - a_{d_i})^2 \int_\Omega \varphi_i dx + v^l, \tag{5.22}$$

so that the state variables a, θ, and v are given resulting from the following state equations

$$a^j = a^{j-1} + \Delta t f\left(\theta^{j-1}, a^j\right), \tag{5.23}$$

$$C\theta^j = A\left(\frac{\rho c_\rho}{\Delta t}\theta^{j-1} - \rho L f\left(\theta^{j-1}, a^{j-1}\right) + \mathbf{u}^j \alpha^j\right), \tag{5.24}$$

$$v^j = v^{j-1} + \frac{\Delta t \beta_2}{2} \sum_{i=1}^{n} [\theta_i^{j-1} - \theta_{m_i}]_+^2 \int_\Omega \varphi_i dx + \frac{\Delta t \beta_3}{2} \left(\mathbf{u}^j\right)^2. \tag{5.25}$$

Here, $j = 1, ..., l$,

$$C = \left(\frac{\rho c_\rho}{\Delta t} A + kB\right), \qquad (a^j)^\top = (a_1^j, ..., a_n^j), \qquad (\theta^j)^\top = (\theta_1^j, ..., \theta_n^j),$$

$v^j \in \mathbb{R}$, and $\mathbf{u}^j \in \mathbb{R}$. A and B are matrices to be evaluated in the following way

$$A = (A_{ij})_{i,j=1}^n = \left(\int_\Omega \varphi_i \varphi_j\right)_{i,j=1}^n, \qquad B = (B_{ij})_{i,j=1}^n = \left(\int_\Omega \nabla\varphi_i \nabla\varphi_j\right)_{i,j=1}^n.$$

Initial values a^0, θ^0, and v^0 are defined by

$$a^0 = 0, \qquad \theta^0 A = \theta_0 A, \qquad v^0 = 0. \tag{5.26}$$

The discrete Hamiltonian function $H^j(a^j, \theta^j, v^j, \bar{a}^j, \bar{\theta}^j, \bar{v}^j, \mathbf{u}^j)$, $0 \le j \le l$, for the discrete optimal control problem (5.22) - (5.26) is defined by

$$H^j(a^j, \theta^j, v^j, \bar{a}^j, \bar{\theta}^j, \bar{v}^j, \mathbf{u}^j) = \bar{a}^{j^\top} a^j + \bar{\theta}^{j^\top} \theta^j + \bar{v}^j v^j, \tag{5.27}$$

with a^j, θ^j, and v^j being given by the state equations (5.23) - (5.26). Therefore, the adjoint problem associated with (5.22) - (5.26) has the following form

$$\bar{a}^{j-1} = \bar{a}^{j^\top} \frac{\partial a^j}{\partial a^{j-1}} + \bar{\theta}^{j^\top} \frac{\partial \theta^j}{\partial a^{j-1}}, \tag{5.28}$$

$$\bar{\theta}^{j-1} = \bar{a}^{j^\top} \frac{\partial a^j}{\partial \theta^{j-1}} + \bar{\theta}^{j^\top} \frac{\partial \theta^j}{\partial \theta^{j-1}} + \bar{v}^j \frac{\partial v^j}{\partial \theta^{j-1}}, \tag{5.29}$$

$$\bar{v}^{j-1} = \bar{v}^j, \tag{5.30}$$

for $j = l, ..., 1$. The partial derivatives $\frac{\partial a^j}{\partial a^{j-1}}$ and $\frac{\partial a^j}{\partial \theta^{j-1}}$ are to be find using the Implicit Function Theorem (see also Section 3.5.3). The final values $\bar{a}^l$, $\bar{\theta}^l$, and $\bar{v}^l$ are defined by

$$\bar{a}^l = \left(\frac{\partial J(\mathbf{u}, a^l, \theta^l, v^l)}{\partial a^l}\right)^\top, \qquad \bar{\theta}^l = 0, \qquad \bar{v}^l = 1. \tag{5.31}$$

Furthermore, the **First Order Necessary Optimality Conditions** for the discrete optimal control problem (5.22) - (5.26) are given by

$$\frac{\partial H^j}{\partial \mathbf{u}^j} = \bar{\theta}^{j^\top} P^j + \bar{v}^j \Delta t \beta_3 \mathbf{u}^j = 0, \qquad 0 \le j \le l, \tag{5.32}$$

with $P^j \in \mathbb{R}^n$ being a vector, defined by $CP^j = A\alpha^j$. Then, the **Second Order Sufficient Conditions** for the discrete optimal control problem (5.22) - (5.26) are given by

$$\frac{\partial H^j}{\partial \mathbf{u}^j \partial \mathbf{u}^j} = \bar{v}^j \Delta t \beta_3 \ge 0, \qquad 0 \le j \le l. \tag{5.33}$$

Since the **Strict Legendre-Clebsch Conditions** 3.1 are satisfied, due to (5.33), and the **Coercivity Conditions** 3.2 are satisfied, due to the construction of Pantoja Algorithm 3.8 with Riccati-type decoupling, Theorem 3.1 can be applied. This theorem provides the existence of a local solution for the optimal control problem (5.22) - (5.26).

The equations (5.23) - (5.26), (5.28) - (5.31), and (5.32) build the discrete Euler-Lagrange equations for the optimal control problem (5.22) - (5.26). Therefore, the task to find an optimal control for the problem (5.22) - (5.26) is reduced to find appropriate $(a^j, \theta^j, v^j, \bar{a}^j, \bar{\theta}^j, \bar{v}^j, \mathbf{u}^j)$, for $j = 0, ..., l$, satisfying (5.23) - (5.26), (5.28) - (5.31), and (5.32). To make this, we apply Procedure 5.1.

Procedure 5.1 Newton's Method for Optimal Control Problem of Laser Surface Hardening of Steel
1. Choose an initial control $\mathbf{u}^0 = \mathbf{u}_{exp}$ *and set* $i = 0$.
2. Compute $\delta\mathbf{u}^i_{init}$ *using Pantoja Algorithm 3.8.*
3. Select the step size λ^i *by the Backtracking Procedure 5.2*
4. Determine the resulting control correction $\delta\mathbf{u}^i_{backtr} = \lambda^i \delta\mathbf{u}^i_{init}$.
5. **If** *the relative norm*

$$\Lambda(i) = \frac{\|\delta\mathbf{u}^i_{backtr}\|_{L^2(0,T)}}{\|\delta\mathbf{u}^i_{backtr} + \mathbf{u}^i\|_{L^2(0,T)}} \tag{5.34}$$

is greater than $\epsilon_{rel} > 0$ *and* i *has not reached the iteration upper bound, then, set* $i = i + 1$, $\mathbf{u}^i = \mathbf{u}^{i-1} + \delta\mathbf{u}^{i-1}_{backtr}$, *and go back to Step* 2.
Otherwise *Stop.*

We take a backtracking line search based on the sufficient decrease condition

$$J\left(\mathbf{u}^i(\lambda^i)\right) - J(\mathbf{u}^i) \leq c\lambda^i \nabla J(\mathbf{u}^i)^\top \delta\mathbf{u}^i_{init}, \tag{5.35}$$

where $\mathbf{u}^i$ is the actual iterate for the control, $c = 0.0001$,

$$\mathbf{u}^i(\lambda^i) = \mathbf{u}^i + \lambda^i \delta\mathbf{u}^i_{init}, \quad \text{for} \quad \lambda^i \geq 0. \tag{5.36}$$

We start the line search algorithm by choosing the initial step size $\lambda^i = 1$ and then we use the backtracking procedure from [28]. The backtracking line search terminates by finding the first λ^i for which (5.35) is satisfied. Then, $\delta\mathbf{u}^i_{backtr}$ is defined by

$$\delta\mathbf{u}^i_{backtr} = \lambda^i \delta\mathbf{u}^i_{init}. \tag{5.37}$$

We terminate Procedure 5.1 provided (5.34) is smaller than ϵ_{rel}.

We now specify how our line-search algorithm will choose λ^i. The description closely follows the description in [28]. As have stated, we start with $\lambda^i = 1$, and then, if $\mathbf{u}^i + \delta\mathbf{u}^i_{init}$ is not acceptable, backtrack (reduce λ^i) until an acceptable $\mathbf{u}^i + \lambda^i\delta\mathbf{u}^i_{init}$ is found. The framework of such an algorithm is given below.

Procedure 5.2 Backtracking Line-search Procedure
Given $c \in (0, \frac{1}{2})$ $\quad 0 < lower < upper < 1.$
$\lambda^i = 1.$
While $J\left(\mathbf{u}^i(\lambda^i)\right) > J(\mathbf{u}^i) + c\lambda^i\nabla J(\mathbf{u}^i)^\top\delta\mathbf{u}^i_{init},$, **do**
$\lambda^i = \rho\lambda^i$ *for some* $\rho \in [lower, upper]$.
ρ *is chosen anew each time by the line search.*
Return λ^i.

In practice, c is quite small, so that hardly more than a decrease in function value is required. Our algorithm uses $c = 10^{-4}$. Now, just the strategy for reducing λ^i (choosing ρ) remains. Let us define

$$\hat{J}(\lambda) = J(\mathbf{u}^i + \lambda^i\delta\mathbf{u}^i_{init}), \tag{5.38}$$

the one-dimensional restriction of J to the line through $\mathbf{u}^i$ in the direction $\delta\mathbf{u}^i_{init}$. If we need to backtrack, we will use our most current information about $\hat{J}$ to model it, and then, take the value of λ that minimises this model as our next value of λ^i in Procedure 5.2.

Initially, we have two peaces of information about $\hat{J}(\lambda)$.

$$\hat{J}(0) = J(\mathbf{u}^i), \qquad and \qquad \hat{J}'(0) = \nabla J(\mathbf{u}^i)^\top\delta\mathbf{u}^i_{init}. \tag{5.39}$$

After calculating $J(\mathbf{u}^i + \delta\mathbf{u}^i_{init})$, we also know that

$$\hat{J}(1) = J(\mathbf{u}^i + \delta\mathbf{u}^i_{init}), \tag{5.40}$$

so if $J(\mathbf{u}^i + \delta\mathbf{u}^i_{init})$ does not satisfy (5.35) (i.e. $\hat{J}(1) > \hat{J}(0) + c\hat{J}'(0)$), we model $\hat{J}(\lambda)$ by the one-dimensional quadratic satisfying (5.39) and (5.40),

$$\hat{m}_q(\lambda) = (\hat{J}(1) - \hat{J}(0) - \hat{J}'(0))\lambda^2 + \hat{J}'(0)\lambda + \hat{J}(0), \tag{5.41}$$

and calculate the point

$$\hat{\lambda} = \frac{-\hat{J}'(0)}{2(\hat{J}(1) - \hat{J}(0) - \hat{J}'(0))}, \tag{5.42}$$

for which $\hat{m}_q'(\lambda) = 0$. Now,

$$\hat{m}_q''(\lambda) = 2(\hat{J}(1) - \hat{J}(0) - \hat{J}'(0)) > 0, \tag{5.43}$$

since $\hat{J}(1) > \hat{J}(0) + \hat{J}'(0)$. Thus, $\hat{\lambda}$ minimises $\hat{m}_q(\lambda)$. Also $\hat{\lambda} > 0$, because $\hat{J}'(0) < 0$. Therefore, we take $\hat{\lambda}$ as our new value of λ^i in Procedure 5.2 (see Figure 5.7). Note that since $\hat{J}(1) > \hat{J}(0) + c\hat{J}'(0)$, we have

$$\hat{\lambda} < \frac{1}{2(1-c)}. \tag{5.44}$$

In fact, if $\hat{J}(1) \geq \hat{J}(0)$, then, $\lambda \leq \frac{1}{2}$. Thus, (5.42) gives a useful implicit upper bound of *upper* $\approx \frac{1}{2}$ on the first value of ρ in Procedure 5.2. On the other hand, if $\hat{J}(1)$ is much larger than $\hat{J}(0)$, $\hat{\lambda}$ can be very small. We probably do not want to decrease λ^i too much based on this information, since probably it indicates that $\hat{J}(\lambda)$ is poorly modelled by a quadratic in this region, so we impose the lower bound of *lower* $= \frac{1}{10}$ in Procedure 5.2. This means that at the first backtrack at each iteration, if $\hat{\lambda} < 0.1$, then, we next try $\lambda^i = \frac{1}{10}$.

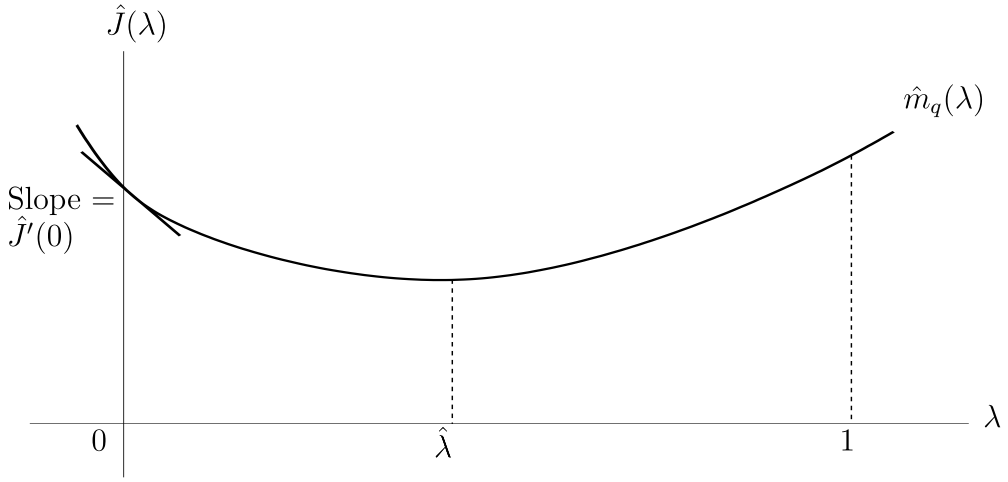

Figure 5.7: Backtracking at the first iteration, using a quadratic model

Now, suppose $\hat{J}(\lambda^i) = J(\mathbf{u}^i + \lambda^i \delta \mathbf{u}^i_{init})$ does not satisfy (5.35). In this case we need to backtrack again. Although we could use a quadratic model as we did on the first backtrack, we now have four pieces of information about $\hat{J}(\lambda^i)$. So, at this and any subsequent backtrack during the current iteration, we use a cubic model of $\hat{J}$, fit $\hat{m}_{cu}(\lambda)$ to $\hat{J}(0)$, $\hat{J}'(0)$, and the last two values of $\hat{J}(\lambda)$, and subject to the same sort of upper and lower limits as before, set λ^i to the value of λ at which $\hat{m}_{cu}(\lambda)$ has its local minimiser (see Figure 5.8).

The reason for using the cubic is that it can more accurately model situations where J has negative curvature, which are likely when (5.35) has failed for two positive values of λ. Furthermore, such a cubic has a unique local minimiser, as illustrated in Figure 5.8.

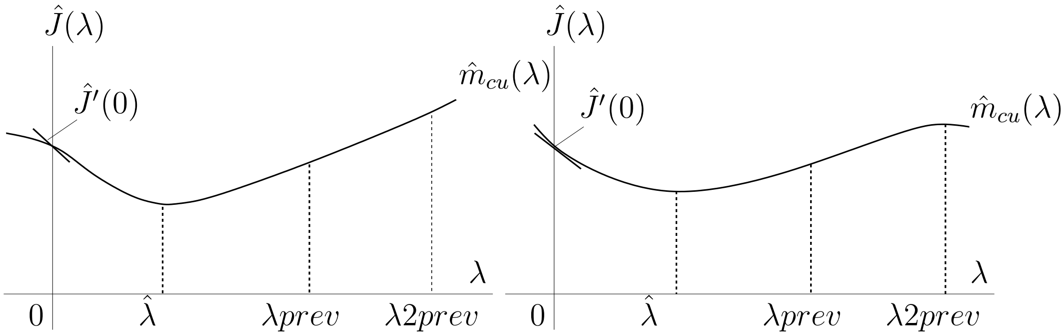

Figure 5.8: Cubic backtrack - the two possibilities

The calculation of λ proceeds as follows. Let $\lambda prev$ and $\lambda 2prev$ be the last two previous values of λ^i. Then, the cubic that fits $\hat{J}(0)$, $\hat{J}'(0)$, $\hat{J}(\lambda prev)$, and $\hat{J}(\lambda 2prev)$ is

$$\hat{m}_{cu}(\lambda) = a\lambda^3 + b\lambda^2 + \hat{J}'(0)\lambda + \hat{J}(0), \tag{5.45}$$

where

(5.46)

$$\begin{bmatrix} a \\ b \end{bmatrix} = \frac{1}{\lambda prev - \lambda 2prev} \times \begin{bmatrix} \frac{1}{\lambda prev^2} & \frac{-1}{\lambda 2prev^2} \\ \frac{-\lambda 2prev}{\lambda prev^2} & \frac{\lambda prev}{\lambda 2prev^2} \end{bmatrix} \begin{bmatrix} \hat{J}(\lambda prev) - \hat{J}(0) - \hat{J}'(0)\lambda prev \\ J(\lambda 2prev) - \hat{J}(0) - \hat{J}'(0)\lambda 2prev \end{bmatrix}.$$

Its local minimiser point $\hat{\lambda}$ is

$$\frac{-b + \sqrt{b^2 - 3a\hat{J}'(0)}}{3a}. \tag{5.47}$$

It can be shown that if $\hat{J}(\lambda prev) \geq \hat{J}(0)$, then, $\hat{\lambda} < \frac{2}{3}\lambda prev$, but this reduction can be achieved in practice and is considered too small. Therefore, the upper bound $upper = 0.5$ is imposed, which means that if $\hat{\lambda} > \frac{1}{2}\lambda prev$, we set the new $\lambda^i = \frac{1}{2}\lambda prev$. Also, since $\hat{\lambda}$ can be an arbitrary small fraction of $\lambda prev$, the lower bound $lower = \frac{1}{10}$ is used again (i.e., if $\hat{\lambda} < \frac{1}{10}\lambda prev$, we set the new $\lambda^i = \frac{1}{10}\lambda prev$).

5.2.5 Test Runs

In the following, we present test runs for the optimal control problem (5.22) - (5.26) using Pantoja Algorithm 3.8 for implicit state equations. This is because the state variable a is discretised using the implicit Euler scheme (accordingly to (5.23)). Here, we use a modified version of Pantoja Algorithm (see [26]). In each step of Pantoja Algorithm 3.8 we replace the matrix C, which in this example is a scalar, with its absolute value. Pantoja Algorithm with this modification produces a descent direction in most cases. In these examples the FE triangulation of Ω is done by a nonuniform mesh with $n = 147$ degrees of freedom. The following three examples differ in the choice of the desired hardening profile a_d. The weights are chosen as $\beta_1 = 20000$, $\beta_2 = 10000$, and $\beta_3 = 0.001$.

Example 5.1 *The desired volume fraction of austenite is shown in Figure 5.9. We choose $T = 5.25$ and want to achieve a constant hardening depth of $1mm$. As the first iterate for the control we take $\mathbf{u}^0 = 400$. Algorithm 3.8 takes 35 iterations and stops with $\Lambda(35) < 10^{-6}$.*

In Figure 5.10 and Figure 5.11 the differences $a^0(T) - a_d$ and $a^{35}(T) - a_d$, respectively, are shown. Using the optimal control, we obtain a significant reduction of the residuum. Figure 5.12 and Figure 5.13 show the initial a^0 and, respectively, the terminal a^{35} volume fraction of austenite at the time $t = T$. A comparison of Figure 5.12 with Figure 5.13 shows that the goal of a uniform hardening depth has been nearly achieved. The resulting optimal control $\mathbf{u}^{35}$ is depicted in Figure 5.14. 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.

Figure 5.15 gives two plots for the initial gradient ∇J^0 and the final gradient ∇J^{35} of the cost function as a function on time t. On these graphs we can see how the gradient ∇J^0 is reduced after 35 iterations.

Table 5.3 and Table 5.4 present data obtained during the run of the program for Example 5.1. The first column of these tables give 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 5.3 and Table 5.4. 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* $\|a^{35}(T) - a_d\|_{L^2(\Omega)}$ *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* $\Lambda(i)$ *evaluated by (5.34) are given in the sixth, the seventh, and the eighth columns of Table 5.3 and Table 5.4. We observe that* $\|\theta^{35}\|_{L^\infty(Q)} = 986.605683$*,* $\|a^0(T) - a_d\|_{L^2(\Omega)} = 0.158144$*, and* $\|a^{35}(T) - a_d\|_{L^2(\Omega)} = 0.045634$*. The norm* $\|\nabla J(\mathbf{u}^i)\|_2$ *of the gradient is reduced from* 2.197499 *to* 0.0*. Moreover, the cost functional* $J(\mathbf{u}^0) = 670.096247$ *is reduced to* $J(\mathbf{u}^{35}) = 355.043510$*.*

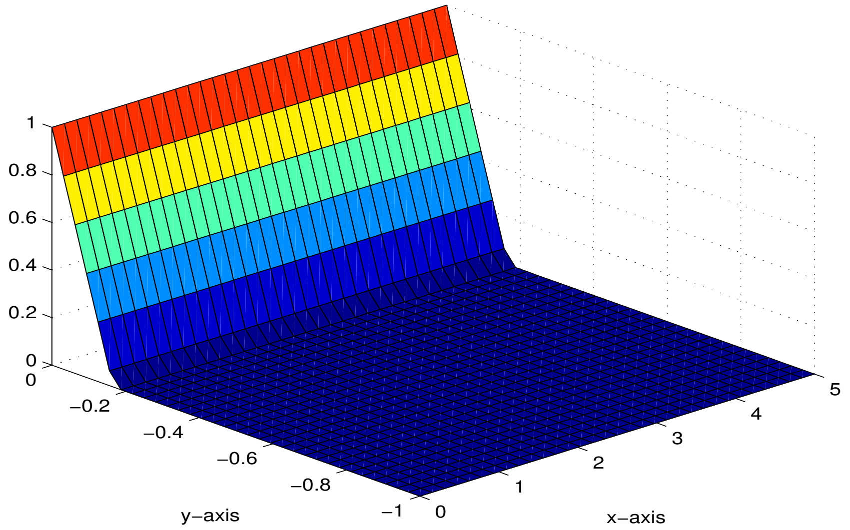

Figure 5.9: Example 5.1: Desired state a_d

Table 5.3: Program data for Example 5.1

i	λ^i	$J(\mathbf{u}^i)$	$\lVert\nabla J(\mathbf{u}^i)\rVert_2$	$\nabla J(\mathbf{u}^i)\delta\mathbf{u}^i$	$\lVert a^i(T) - a_d\rVert_{L^2(\Omega)}$	$\lVert\theta^i\rVert_{L^\infty(Q)}$	$\Lambda(i)$
0	0.000000	670.096247	2.197499	−0.000000	0.158144	1148.594458	0.000000
1	0.001130	665.493535	2.198253	−4275.090081	0.157792	1148.527231	0.019460
2	0.005000	650.324921	2.194708	−3257.726444	0.156671	1148.370625	0.032751
3	0.000034	647.678880	2.194745	−82044.859253	0.156744	1148.433767	0.014622
4	0.000001	647.678880	2.166298	782.845599	0.155249	1147.414625	0.000561
5	0.000000	647.678880	2.136994	2524.100850	0.153778	1146.409009	0.000554
6	0.100000	589.317913	1.864956	−543.282251	0.140428	1141.066647	0.088242
7	0.134720	576.179109	1.824791	−409.596505	0.144286	1134.496470	0.088891
8	0.100000	534.398496	1.378520	−668.869910	0.123036	1201.254915	0.236620
9	0.100000	522.596740	1.229784	−209.076937	0.119931	1185.524519	0.094729
10	0.100000	507.601359	1.066479	−192.808499	0.114103	1178.850090	0.130864
11	1.000000	458.536134	0.463360	−212.394660	0.069952	1219.525080	0.391770
12	0.000005	458.536134	0.460430	375.676115	0.069810	1219.512293	0.000117
13	0.050000	454.151963	0.400742	−303.119052	0.064523	1203.078728	0.181681
14	0.010000	449.265299	0.394659	−694.610663	0.063295	1134.568968	0.074352
15	0.100000	434.308670	0.354590	−195.703647	0.060600	1098.145344	0.085859

Table 5.4: Program data for Example 5.1

i	λ^i	$J(\mathbf{u}^i)$	$\|\nabla J(\mathbf{u}^i)\|_2$	$\nabla J(\mathbf{u}^i)\delta\mathbf{u}^i$	$\|a^i(T) - a_d\|_{L^2(\Omega)}$	$\|\theta^i\|_{L^\infty(Q)}$	$\Lambda(i)$
16	0.068913	425.930733	0.336206	−183.298145	0.062027	1070.120815	0.064029
17	0.106795	418.530501	0.296163	−87.992032	0.060585	1056.262926	0.094609
18	0.100000	412.269121	0.269185	−91.874846	0.059917	1048.402661	0.070154
19	1.000000	377.876166	0.109762	−62.712936	0.048161	1075.204147	0.254459
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
25	0.466521	358.256298	0.070310	−12.930429	0.049959	992.202982	0.072710
26	0.168467	357.800026	0.057865	−6.642662	0.049911	1003.807791	0.033639
27	0.339261	357.084832	0.041801	−4.338000	0.048627	995.732876	0.043046
28	0.297777	356.634981	0.030828	−3.579584	0.047516	995.080975	0.033259
29	1.000000	356.294331	0.027788	−1.311356	0.045011	1007.091796	0.045189
30	1.000000	355.305487	0.019172	−2.436999	0.045926	987.141029	0.052059
31	1.000000	355.106094	0.004853	−0.396550	0.045660	986.614861	0.019979
32	1.000000	355.047319	0.001508	−0.103916	0.045595	986.605644	0.008947
33	1.000000	355.043523	0.000163	−0.008023	0.045628	986.605683	0.003201
34	1.000000	355.043510	0.000006	−0.000185	0.045634	986.605683	0.000582
35	1.000000	355.043510	0.000000	−0.000000	0.045634	986.605683	0.000000

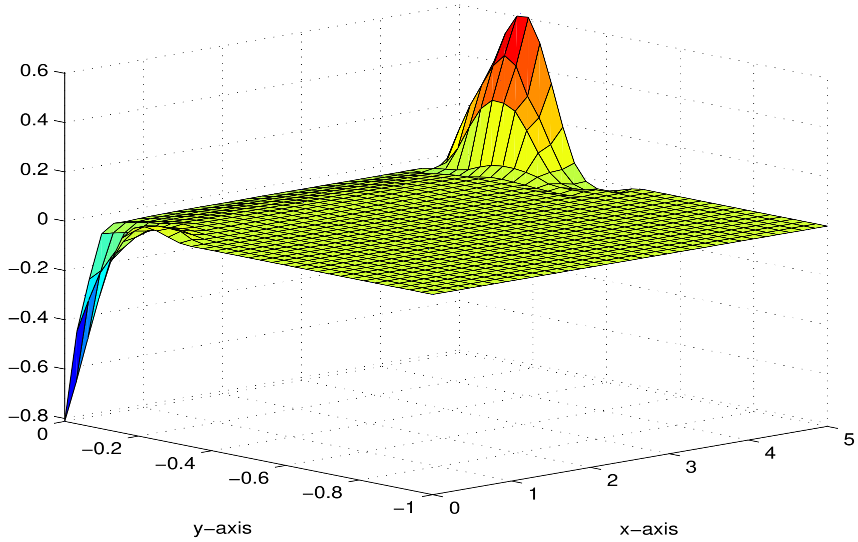

Figure 5.10: Example 5.1: Difference $a^0(T) - a_d$ for the first iterate a^0

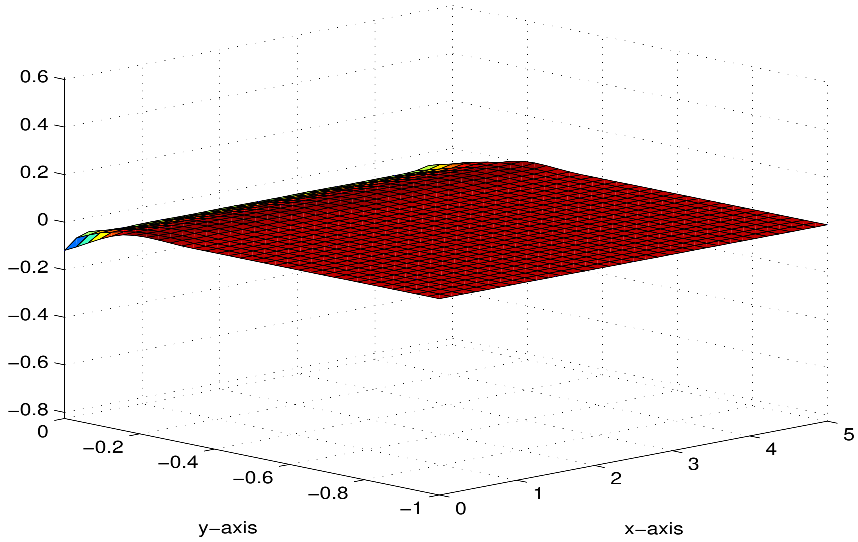

Figure 5.11: Example 5.1: Difference $a^{35}(T) - a_d$ for the optimal state a^{35}

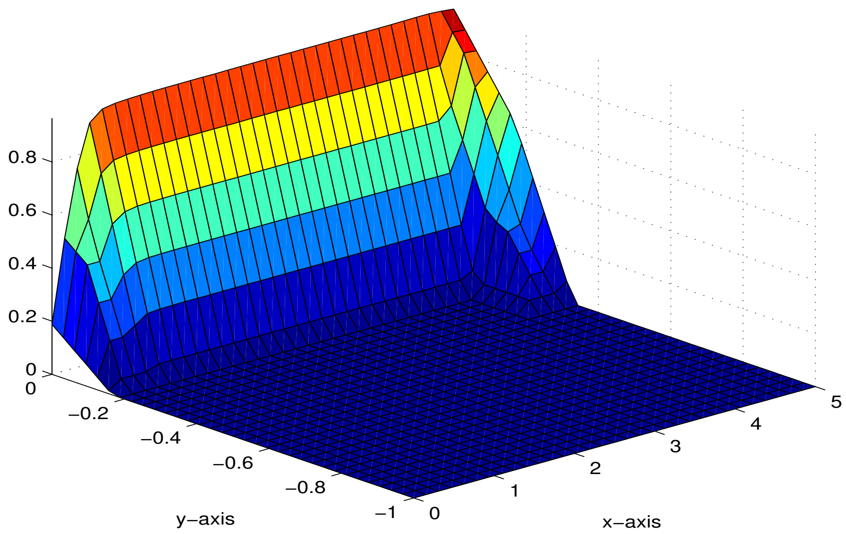

Figure 5.12: Example 5.1: FE snapshot for the austenite at time $t = T$ for the first iterate a^0

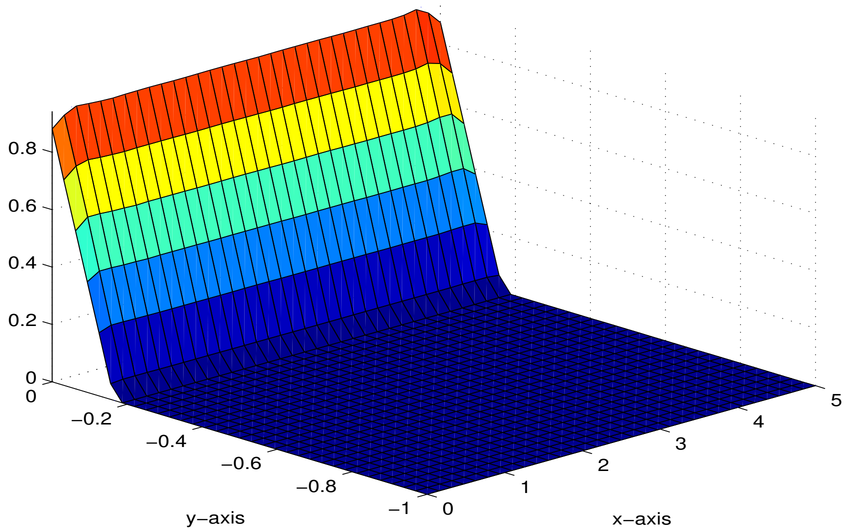

Figure 5.13: Example 5.1: FE snapshot for the austenite at time $t = T$ for the optimal state a^{35}

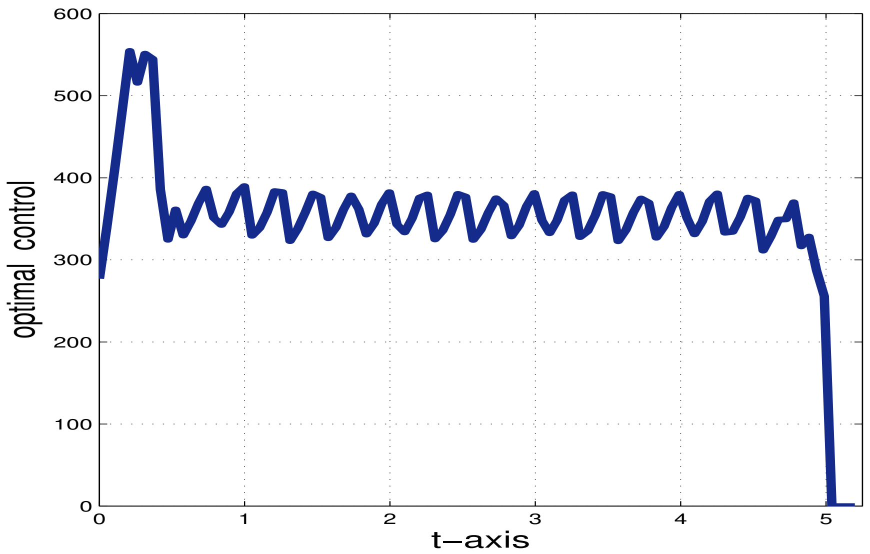

Figure 5.14: Example 5.1: Optimal control $\mathbf{u}^{35}$

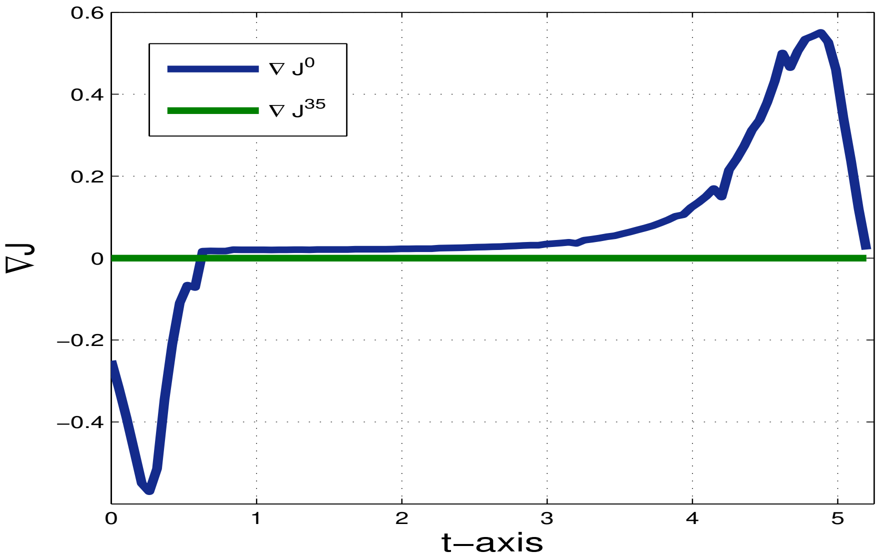

Figure 5.15: Example 5.1: Gradients ∇J^0 and ∇J^{35} of the cost function

In the following simulations, we consider two different step profiles for the hardening depth.

Example 5.2 *The desired volume fraction of austenite is shown in Figure 5.16. We choose* $T = 5.25$*. As the first iterate for the control we take* $\mathbf{u}^0 = 450$*. Algorithm 3.8 takes* 58 *iterations and stops with* $\Lambda(58) < 10^{-6}$*. In Figure 5.17 and Figure 5.18 the differences* $a^0(T) - a_d$ *and* $a^{58}(T) - a_d$*, respectively, are shown. Using the optimal control, we obtain a significant reduction of the residuum. Figure 5.19 and Figure 5.20 show the initial* a^0 *and, respectively, the terminal* a^{58} *volume fraction of austenite at the time* $t = T$*. A comparison of Figure 5.19 with Figure 5.20 shows that the desired profile from Figure 5.16 has been nearly achieved. The resulting optimal control* $\mathbf{u}^{58}$ *is depicted in Figure 5.21.*

Figure 5.22 gives two plots for the initial gradient ∇J^0 *and the final gradient* ∇J^{58} *of the cost function as a function on time* t*. On these graphs we can see how the gradient* ∇J^0 *is reduced after* 58 *iterations.*

Table 5.5 and Table 5.6 present data obtained during the run of the program for Example 5.2. We observe that $\|\theta^{58}\|_{L^\infty(Q)} = 1399.014910$*,* $\|a^0(T) - a_d\|_{L^2(\Omega)} = 0.331357$*, and* $\|a^{58}(T) - a_d\|_{L^2(\Omega)} = 0.045257$*. The norm* $\|\nabla J(\mathbf{u}^i)\|_2$ *of the gradient is reduced from* 3.895398 *to* 0.0*. Moreover, the cost functional* $J(\mathbf{u}^0) = 1629.537138$ *is reduced to* $J(\mathbf{u}^{58}) = 410.031481$*.*

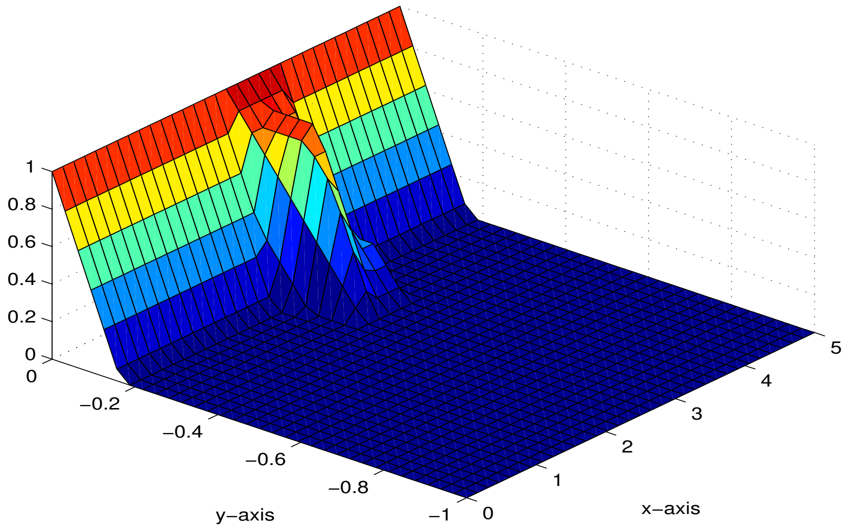

Figure 5.16: Example 5.2: Desired state a_d

Table 5.5: Program data for Example 5.2

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	1629.537138	3.895398	−0.000000	0.331357	1255.882115	0.000000
1	0.003754	1611.671801	3.871172	−2590.303579	0.327943	1291.832216	0.098111
2	0.005000	1368.869791	3.605227	−67646.182231	0.287059	1312.638740	0.130090
3	0.100000	1203.725365	4.033390	−1283.800122	0.259034	1267.876424	0.085950
4	0.010000	1153.733054	4.164837	−6975.730010	0.247527	1347.985982	0.111501
5	0.050000	1052.338603	2.988814	−13632.728437	0.183954	1394.464686	0.633601
6	0.100000	980.090279	2.773759	−1182.119552	0.173256	1315.735037	0.140583
7	0.012769	970.127689	2.747543	−962.269517	0.172171	1315.630818	0.045326
8	0.100000	896.648554	2.530053	−768.552512	0.159823	1318.990094	0.050768
9	0.100000	795.981714	2.102666	−1198.337449	0.134128	1326.482744	0.146982
10	0.100000	747.158865	1.896453	−519.101833	0.124824	1324.461668	0.048985
11	1.000000	546.879725	0.733054	−418.413214	0.077323	1379.555825	0.417800
12	0.050000	539.423873	0.712609	−248.127630	0.074175	1379.961341	0.095983
13	1.000000	497.237133	0.253185	−117.743915	0.045122	1394.481288	0.201156
14	0.000124	497.198788	0.253282	−659.488155	0.044944	1394.479871	0.008128
15	0.050000	490.629260	0.251498	−359.532943	0.049364	1394.131692	0.118292

Table 5.6: Program data for Example 5.2

i	λ^i	$J(\mathbf{u}^i)$	$\lVert\nabla J(\mathbf{u}^i)\rVert_2$	$\nabla J(\mathbf{u}^i)\delta\mathbf{u}^i$	$\lVert a^i(T) - a_d\rVert_{L^2(\Omega)}$	$\lVert\theta^i\rVert_{L^\infty(Q)}$	$\Lambda(i)$
16	0.100000	488.372211	0.218454	−142.106405	0.044211	1393.373479	0.152611
17	0.022133	483.796772	0.183435	−343.512993	0.041484	1392.930484	0.093815
18	0.100000	475.694410	0.168858	−93.754179	0.039885	1392.441895	0.056390
19	0.050000	474.054286	0.168797	−64.636595	0.038212	1391.990872	0.057116
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
48	0.010000	410.249982	0.012402	−0.562673	0.045592	1399.014911	0.000310
49	0.497145	410.087017	0.008992	−0.544514	0.045372	1399.944289	0.015315
50	0.010000	410.086064	0.008903	−0.093094	0.045370	1399.979688	0.000057
51	0.010000	410.085131	0.008232	−0.091470	0.045368	1399.014728	0.000057
52	1.000000	410.034724	0.002175	−0.085665	0.045254	1399.977618	0.005446
53	0.010000	410.034630	0.001775	−0.009154	0.045254	1399.012945	0.000022
54	1.000000	410.031644	0.000529	−0.004615	0.045257	1399.012952	0.001351
55	1.000000	410.031488	0.000167	−0.000041	0.045257	1399.015443	0.000137
56	1.000000	410.031481	0.000009	−0.000000	0.045257	1399.014939	0.000004
57	1.000000	410.031481	0.000000	−0.000000	0.045257	1399.014910	0.000000
58	1.000000	410.031481	0.000000	−0.000000	0.045257	1399.014910	0.000000

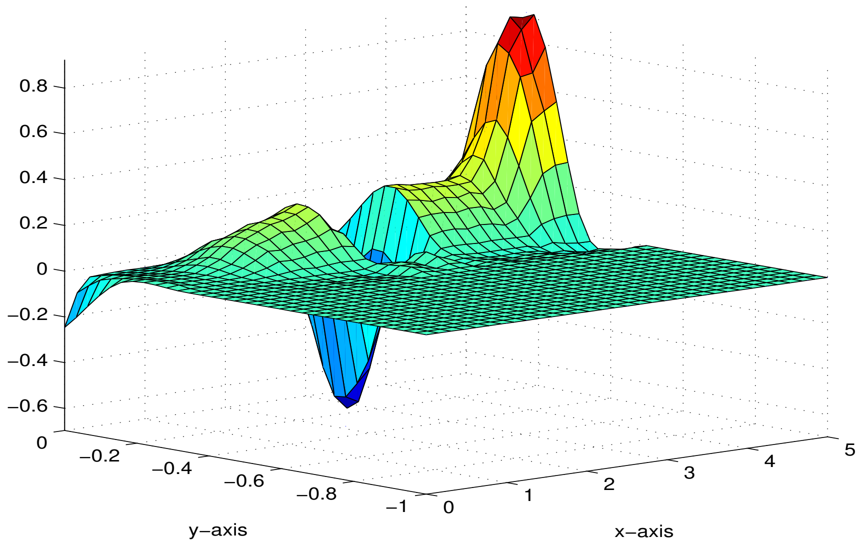

Figure 5.17: Example 5.2: Difference $a^0(T) - a_d$ for the first iterate a^0

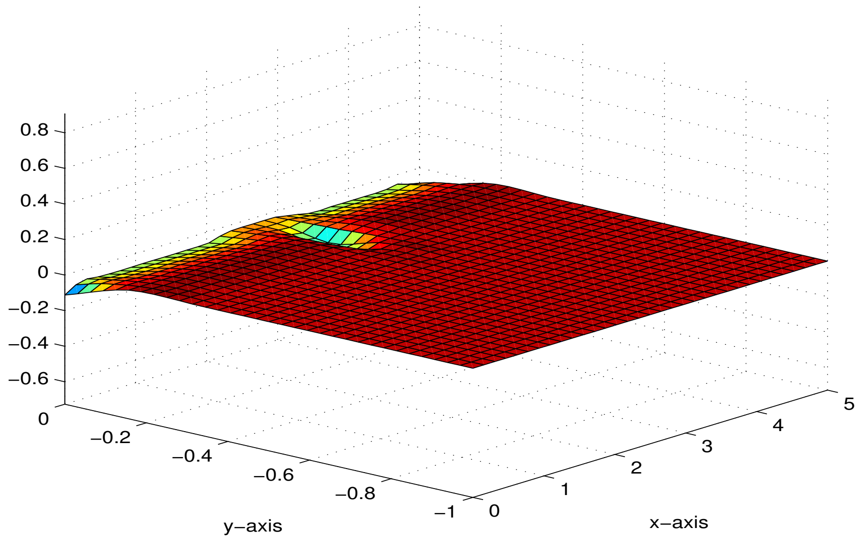

Figure 5.18: Example 5.2: Difference $a^{58}(T) - a_d$ for the optimal state a^{58}

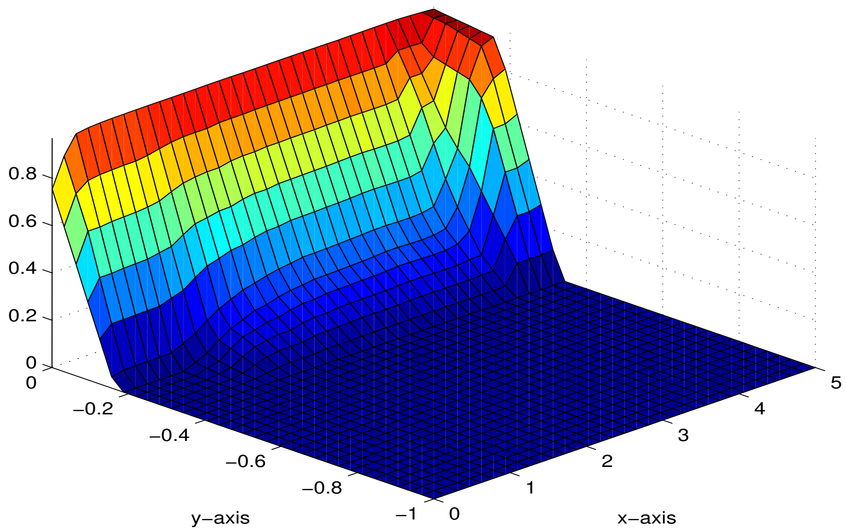

Figure 5.19: Example 5.2: FE snapshot for the austenite at time $t = T$ for the first iterate a^0

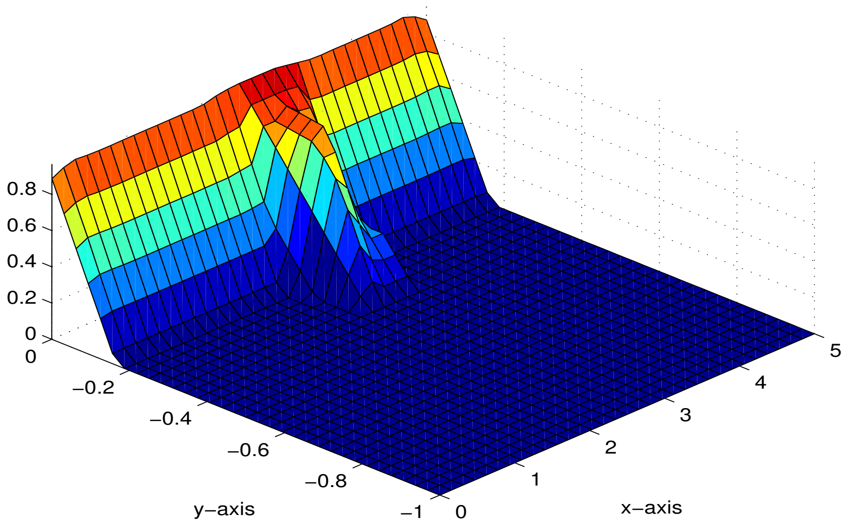

Figure 5.20: Example 5.2: FE snapshot for the austenite at time $t = T$ for the optimal state a^{58}

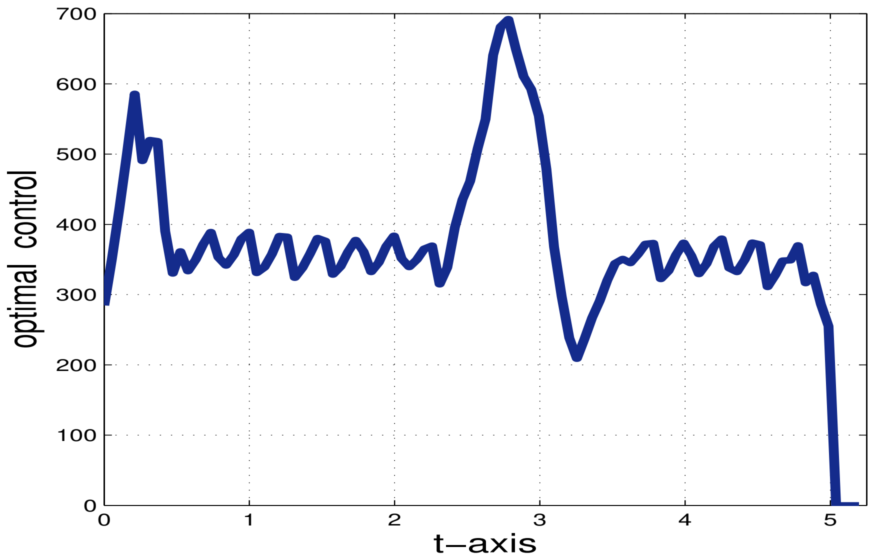

Figure 5.21: Example 5.2: Optimal control $\mathbf{u}^{58}$

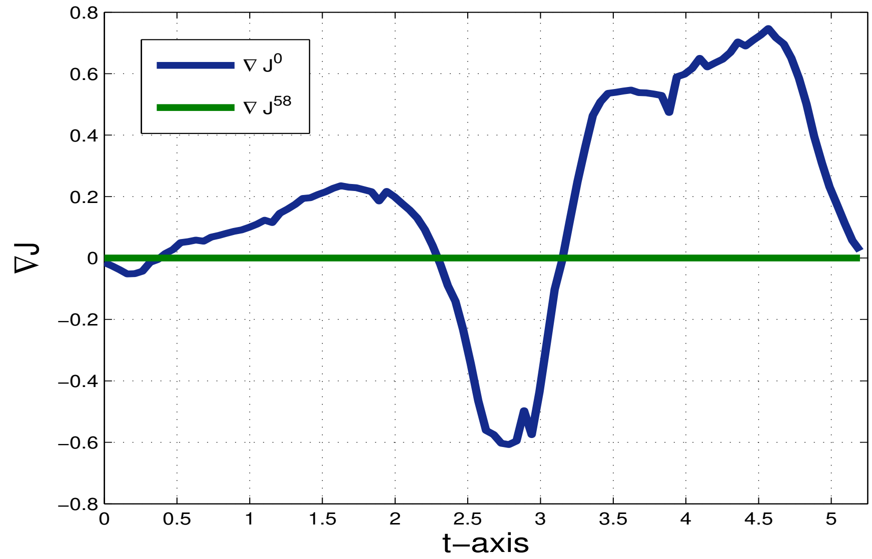

Figure 5.22: Example 5.2: Gradients ∇J^0 and ∇J^{58} of the cost function

Example 5.3 *The desired volume fraction of austenite is shown in Figure 5.23. We choose* $T = 5.25$*. As the first iterate for the control we take* $\mathbf{u}^0 = 470$*. Algorithm 3.8 takes* 58 *iterations and stops with* $\Lambda(58) < 10^{-6}$*. In Figure 5.24 and Figure 5.25 the differences* $a^0(T) - a_d$ *and* $a^{58}(T) - a_d$*, respectively, are shown. Using the optimal control, we obtain a significant reduction of the residuum. Figure 5.26 and Figure 5.27 show the initial* a^0 *and, respectively, the terminal* a^{58} *volume fraction of austenite at the time* $t = T$*. A comparison of Figure 5.26 with Figure 5.27 shows that the desired profile from Figure 5.23 has been nearly achieved. The resulting optimal control* $\mathbf{u}^{58}$ *is depicted in Figure 5.28.*

Figure 5.29 gives two plots for the initial gradient ∇J^0 *and the final gradient* ∇J^{58} *of the cost function as a function on time* t*. On these graphs we can see how the gradient* ∇J^0 *is reduced after* 58 *iterations.*

Table 5.7 and Table 5.8 present data obtained during the run of the program for Example 5.2. We observe that $\|\theta^{58}\|_{L^\infty(Q)} = 1399.341962$*,* $\|a^0(T) - a_d\|_{L^2(\Omega)} = 0.436387$*, and* $\|a^{58}(T) - a_d\|_{L^2(\Omega)} = 0.045697$*. The norm* $\|\nabla J(\mathbf{u}^i)\|_2$ *of the gradient is reduced from* 6.471205 *to* 0.0*. Moreover, the cost functional* $J(\mathbf{u}^0) = 2484.202016$ *is reduced to* $J(\mathbf{u}^{58}) = 466.998875$*.*

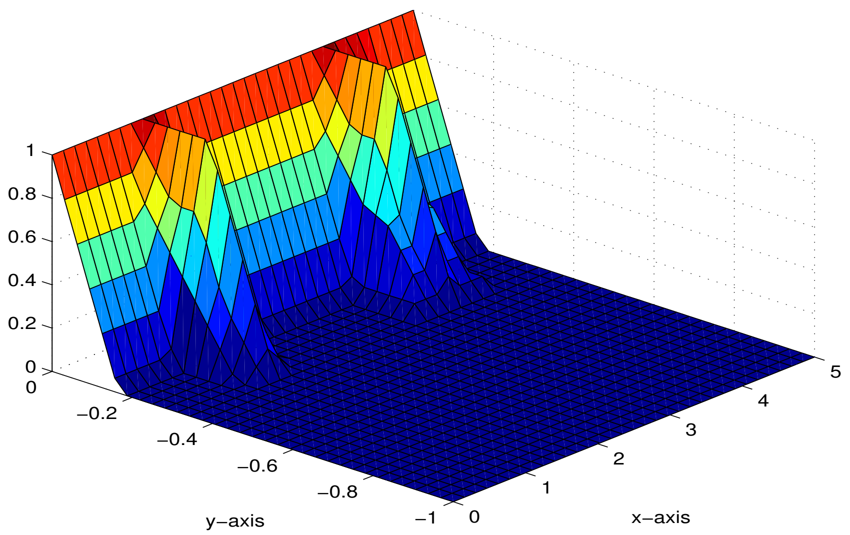

Figure 5.23: Example 5.3: Desired state a_d

Table 5.7: Program data for Example 5.3

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	2484.202016	6.471205	−0.000000	0.436387	1294.056963	0.000000
1	0.000010	2454.376184	6.574132	−2663809.536351	0.432893	1346.932265	0.066236
2	0.000008	2454.376184	6.445610	17421.636759	0.428029	1346.184868	0.001398
3	0.000002	2454.376184	6.325539	157748.319035	0.423300	1345.430728	0.001372
4	0.000100	2210.290827	6.395595	−1968777.730847	0.401929	1358.426986	0.270880
5	0.000007	2105.389365	6.630179	−28168256.947977	0.387213	1326.644143	0.308504
6	0.100000	1411.275126	5.319535	−9780.214146	0.278602	1350.500997	0.333429
7	0.010000	1406.142580	5.268190	−1096.638508	0.277218	1367.616729	0.066527
8	0.100000	1239.169729	4.655845	−1917.013380	0.246501	1378.774770	0.148495
9	0.100000	1112.492483	4.106241	−1361.789060	0.222954	1387.140576	0.080206
10	0.100000	1039.066756	3.618044	−1132.844713	0.204268	1381.173728	0.201045
11	0.001096	1038.778714	3.616225	−1230.697114	0.204368	1381.171147	0.014064
12	0.022861	1015.728303	3.536878	−1022.517997	0.200093	1381.176106	0.044817
13	0.050000	972.800479	3.316391	−867.004927	0.190759	1382.048355	0.058894
14	0.100000	911.602648	2.955902	−647.685403	0.178165	1382.095917	0.036741
15	0.100000	860.802333	2.666781	−534.167493	0.167037	1378.973914	0.032842

Table 5.8: Program data for Example 5.3

i	λ^i	$J(\mathbf{u}^i)$	$\|\nabla J(\mathbf{u}^i)\|_2$	$\nabla J(\mathbf{u}^i)\delta\mathbf{u}^i$	$\|a^i(T) - a_d\|_{L^2(\Omega)}$	$\|\theta^i\|_{L^\infty(Q)}$	$\Lambda(i)$
16	0.100000	817.847497	2.399560	−452.061138	0.156972	1370.243286	0.029916
17	1.000000	603.053380	0.833357	−390.517252	0.090633	1389.629964	0.295629
18	0.005000	602.973938	0.833060	−38.429739	0.090657	1390.197083	0.015728
19	0.010000	601.306965	0.826948	−217.242264	0.089060	1391.342871	0.028836
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
48	0.100000	468.848733	0.031382	−2.441952	0.045126	1397.277566	0.003830
49	0.100000	468.686860	0.028586	−1.721384	0.045226	1397.918490	0.004550
50	0.100000	468.520394	0.025866	−1.700835	0.045315	1398.964634	0.003213
51	0.100000	468.340442	0.101542	−1.915463	0.045474	1398.042968	0.004871
52	0.000002	467.185468	0.011354	−0.000001	0.045478	1398.035776	0.000000
53	0.158196	467.111269	0.010382	−0.727791	0.045309	1398.248514	0.008802
54	1.000000	467.006928	0.001695	−0.207760	0.045606	1399.465314	0.012530
55	1.000000	466.999902	0.000336	−0.013714	0.045656	1399.337516	0.004964
56	1.000000	466.998875	0.000099	−0.000360	0.045715	1399.345595	0.000377
57	1.000000	466.998875	0.000008	−0.000080	0.045715	1399.345611	0.000001
58	1.000000	466.998875	0.000000	−0.000000	0.045697	1399.341962	0.000000

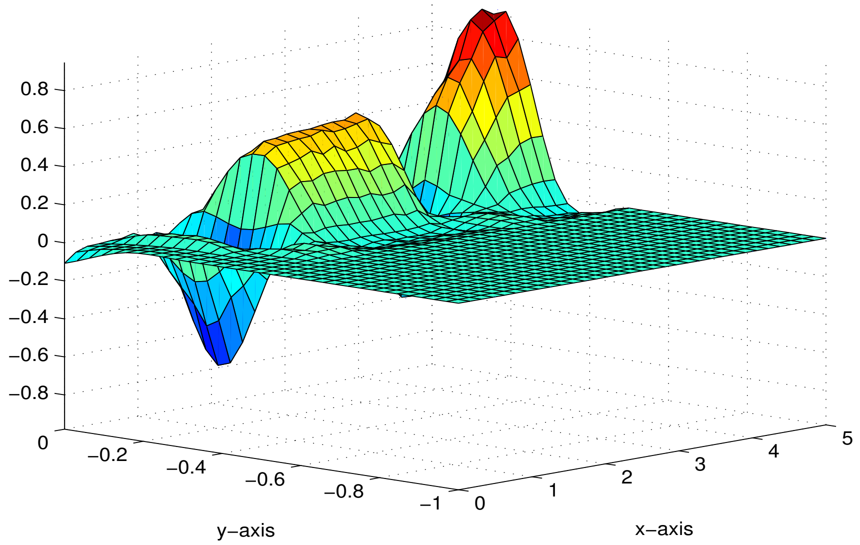

Figure 5.24: Example 5.3: Difference $a^0(T) - a_d$ for the first iterate a^0

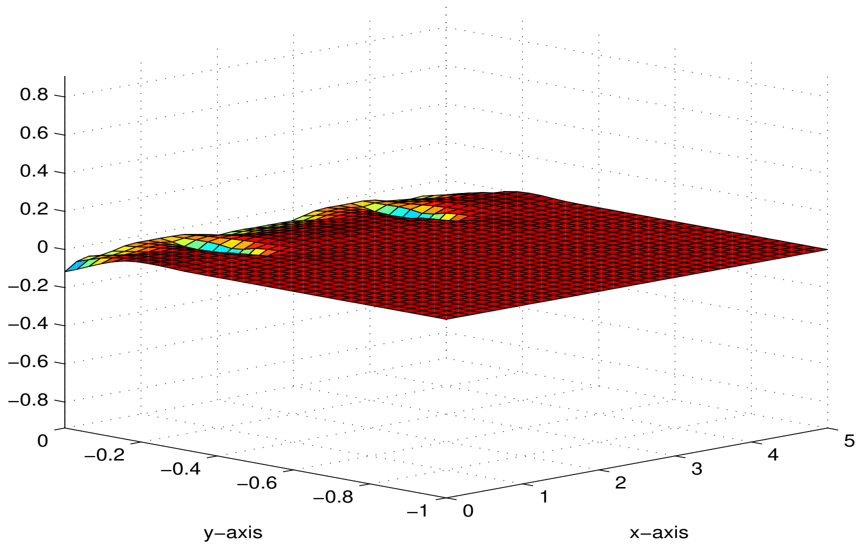

Figure 5.25: Example 5.3: Difference $a^{58}(T) - a_d$ for the optimal state a^{58}

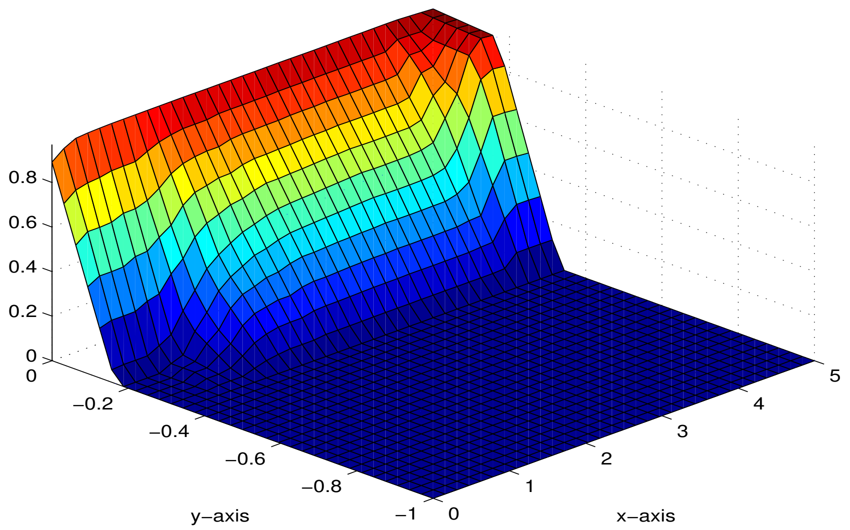

Figure 5.26: Example 5.3: FE snapshot for the austenite at time $t = T$ for the first iterate a^0

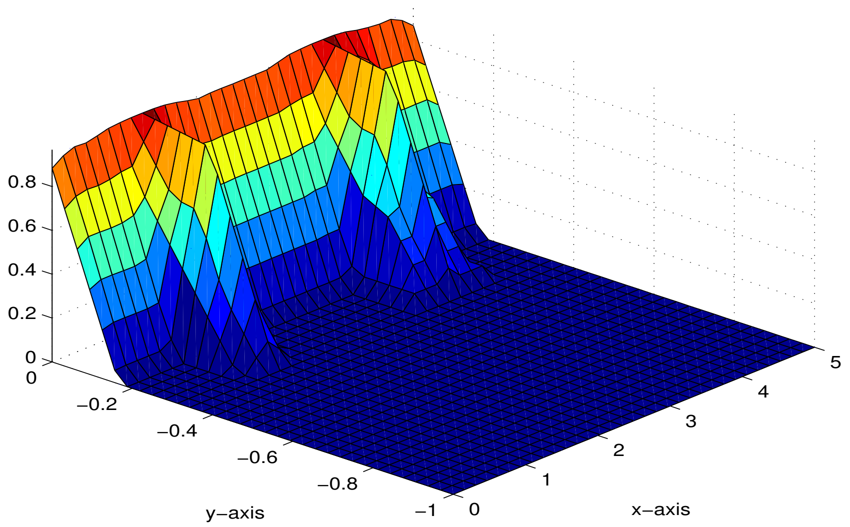

Figure 5.27: Example 5.3: FE snapshot for the austenite at time $t = T$ for the optimal state a^{58}

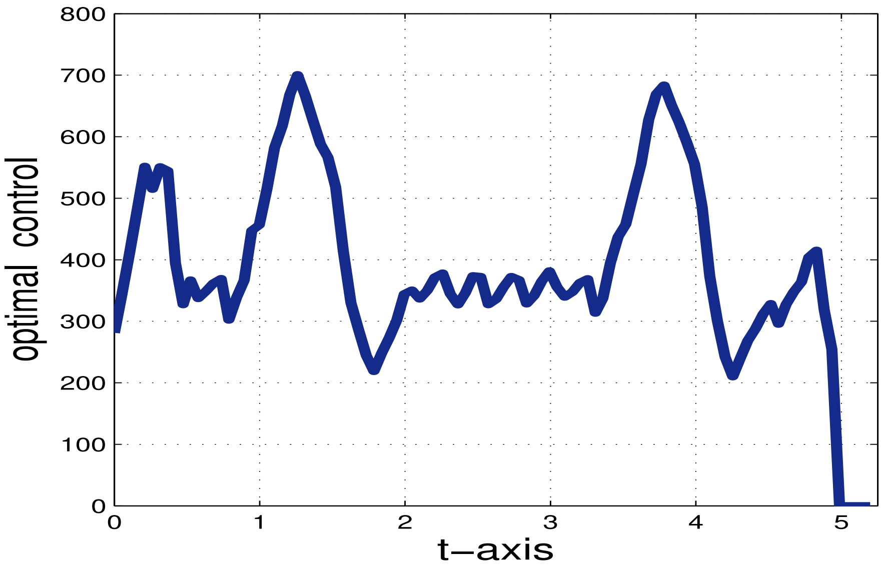

Figure 5.28: Example 5.3: Optimal control $\mathbf{u}^{58}$

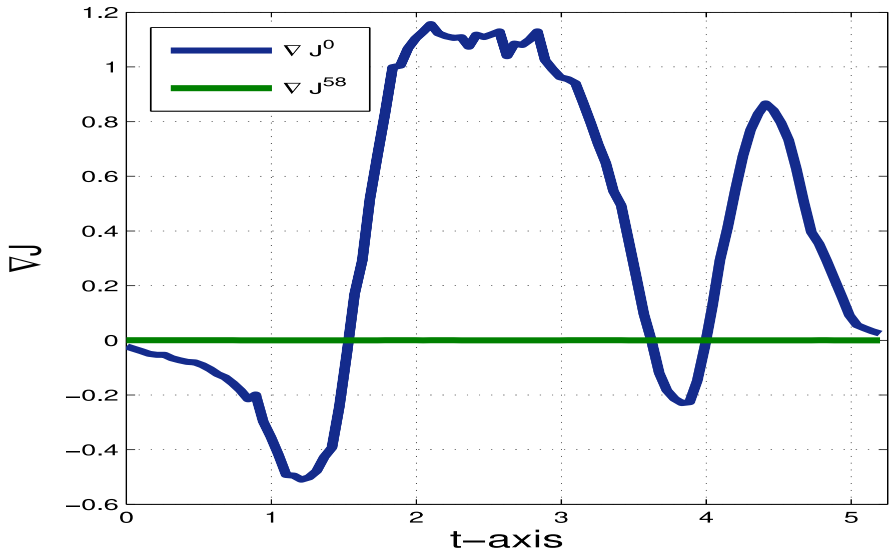

Figure 5.29: Example 5.3: Gradients ∇J^0 and ∇J^{58} of the cost function

5.2.6 Run-time & Memory Trade-off

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 5.30 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.

Blue line shows run-time ratios if only three checkpoints are available. 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 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. Zig-zags in the resulting lines can be explained by memory access effects. In Figure 5.30 we can observe that run-times, resulting from nested reversal schedules, increase in up to four times compared to the basic approach. At the same time, the memory requirement is reduced in 100 times compared to the basic approach.

Obviously, using nested reversal schedules, memory requirement can be reduced enormously, while run-time requirement increases slightly.

5.3 Conclusions

In this chapter algorithms, developed and investigated in this thesis, were applied to numerical problems. Results, achieved throughout, are quite promising.

The optimal control problem of the laser surface hardening of steel was already considered in the literature (see [52]). Optimal controls, produced by numerical methods throughout this thesis are improved, compared to the numerical results from [52]. This means, that the desired hardening profile is nearly achieved, i.e. deviations between resulting hardening profile and the desired hardening profile are reduced compared to the corresponding deviations in [52]. Furthermore, the cost function $J(\mathbf{u})$ is immense reduced.

At the same time, usage of checkpointing techniques, namely the application of nested reversal schedules, results in an enormous reduction of memory

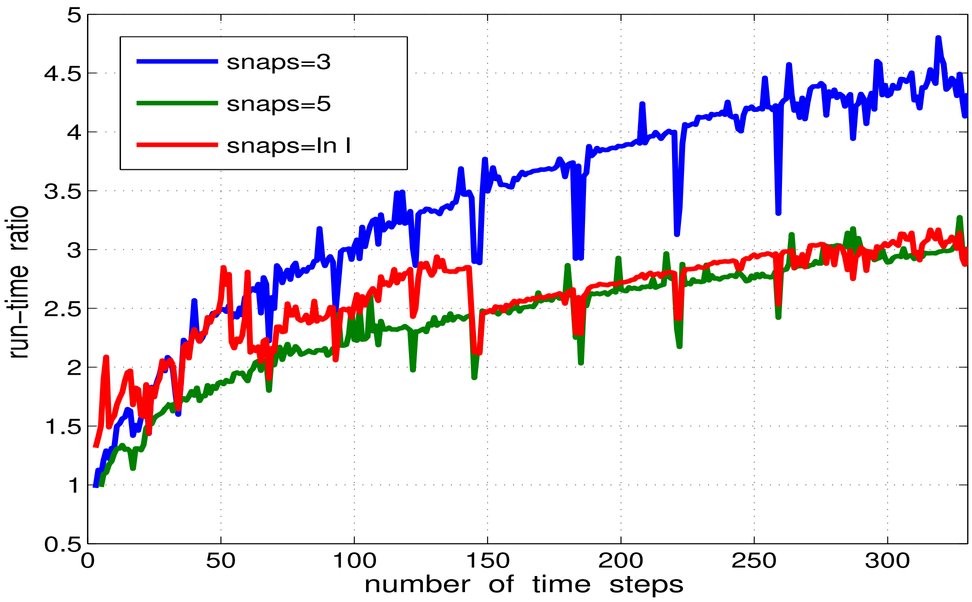

Figure 5.30: Run-time & Memory Trade-off

requirement, needed to implement triple sweep approaches. Although many time steps have to be evaluated more than once, resulting run-times are increased just slightly, compared to the basic approach, where all intermediate states are to be stored.

Chapter 6

Conclusions and Outlook

Unconstrained optimal control problems in ODEs were considered in this thesis. There are a variety of numerical methods to solve these problems. The so-called indirect approach was considered detailed in this thesis. These methods are very popular in the literature. They solve decoupled BVPs resulting from the necessary conditions for the optimal control problem. Different type of multiple shooting techniques for solving decoupled BVPs were considered in this thesis. The iterative solution of optimal control problems in ODEs or the iterative solution of non-linear BVPs with decoupled BC by some of these methods leads to a succession of double sweeps through the discretised time interval. The second (adjoint) sweep relies on information from the first (original) sweep. The key difficulty is given by the enormous amount of memory required for the implementation of these methods if the basic approach is used. For the memory reduced implementation of these methods reversal schedules were utilised. As numerical application it was used to approximate solutions of decoupled BVPs with separated BC using multiple shooting methods. These numerical methods, i.e. the compactification approach, the stabilised march, and the Riccati approach, were coded using the C++ programming language environment. Furthermore, coupled with the routine **revolve** from [38] for the memory reduced implementation of corresponding methods, these routines were utilised for various numerical tests. The achieved results were reported. Furthermore, the concept of symplecticness and Hamiltonian systems was introduced. In this regard, a suitable numerical method was presented, which can be applied to unconstrained optimal control problems. It was proved that this method is a symplectic one. Its numerical implementation will be the subject of further work in the future.

Moreover, theoretical relationships between some types of multiple shooting techniques, namely the Riccati approach, and Pantoja method were investigated. Pantoja method describes a computationally efficient stage-wise construction of the Newton direction for the discrete-time optimal control problem. The equivalence of Pantoja method and multiple shooting method of Riccati type was shown in this thesis. The routine **pantoja** constructing the Newton direction for the discrete-time optimal control problem using Pantoja method was coded. As numerical applications unconstrained optimal control problems in ODEs were considered. The achieved results were reported. It is planned to extend Pantoja method to solve constrained optimal control problems in ODEs. Moreover, Pantoja method was extended to the case where the state equations are discretised using one of implicit numerical methods.

The iterative solution of optimal control problems in ODEs by Pantoja or Riccati equivalent methods leads to a succession of triple sweeps through the discretised time interval. The second (adjoint) sweep relies on information from the first (original) sweep, and the third (final) sweep depends on both of them. This flow of information is depicted in Figure 3.14. Typically, the steps on the adjoint sweep involve more operations and require more storage than the other two. The key difficulty is given by the enormous amount of memory required for the implementation of these methods if the basic approach is used. One of goals of this thesis was to present checkpointing techniques for memory reduced implementation of these methods. The proposed nested reversal schedules drastically reduce the required spatial complexity.

In order to develop these required reversal schedules, the simple case of double sweep evolutions was revisited, so that a forward simulation F combines l time steps of the same temporal complexity. This case has been often discussed in the literature. Nevertheless, this thesis presents new results concerning the construction of optimal reversal schedules for these simulations. For this purpose, a new proof of optimality is given. This new approach yields so far unknown properties of the optimal checkpointing strategies and thus provides new insights. Moreover, these recently developed results are applied to construct nested reversal schedules and to specify their structural properties later on.

Furthermore, evolutions containing multiple sweeps were considered in this thesis. So far, no results were known regarding the memory efficient implementation of such evolutions. The case of evolutions containing three alternative sweeps, the initial, the adjoint, and the final sweep, was considered more detailed. In order to avoid storing full traces of the original and adjoint sweeps, this thesis presents new reversal schedules, the so-called nested reversal schedules, that require only the storage of selected original and adjoint intermediate states called thin and fat checkpoints. The schedules are designed to minimise the overall execution time given a certain total amount of storage for the checkpoints. The routine **optimal**, implementing optimal nested reversal schedules for evolutions containing three alternative sweeps, was coded.

While a closed form solution for this discrete optimisation problem was not found, a cheap heuristic for constructing nested reversals was developed which is quite close to optimality. Numerical results, evaluated for different number of available checkpoints and different number of time steps, were reported. The routine **heuristic**, implementing nested reversal schedules developed by this heuristic for evolutions containing three alternative sweeps, was coded as well. Furthermore, minimal evaluation cost, required to implement an optimal nested reversal schedules, can be estimated by its lower bound established in this thesis. Moreover, it was demonstrated that the dependence on l can be arranged polylogarithmically by nested checkpoint strategies. Consequently, the operations count also grows as a second power of $\ln l$, which however does not need to result in an increase of the actual run-time due to memory effects. As it has been done in the case of simple reversal schedules that involve only an original and an adjoint sweep, nested reversal schedules resulting from this thesis should be extended to scenarios with nonuniform step costs, a-priory unknown overall number of time steps, and parallel computing systems. This will be the subject for further work in the future.

The routine **nested-revolve**, implementing evolutions containing of three alternative sweeps using one of available methods, namely, optimal nested reversal strategies or the developed heuristic, was coded. Furthermore, the computer routines for triple sweep numerical methods, i.e. **red-cmpt**, **riccati**, and **pantoja**, and the reversal schedules routines, i.e. **nested-revolve**, **optimal**, and **heuristic** were summarised to a software solving unconstrained optimal control problems in ODEs using stabilised march, Riccati, or Pantoja methods coupled with nested checkpointing techniques for a memory reduced implementation.

The proposed scheduling schemes were applied to the memory reduced implementation of the optimal control problem of laser surface hardening of steel and other optimal control problems. The recently developed software was applied to the numerical solutions of these problems. For the solution of optimal control problem of laser surface hardening Pantoja method for implicit discretisation of state equations was utilised. The numerical results obtained here for the optimal control problem of laser surface hardening of steel were compared with the previous results from the literature (see [52, 53, 88, 89, 90]). The observations were reported.

List of Figures

List of Tables

Bibliography

[1] *Test Set for Initial Value Problems, http://pitagora.dm.uniba.it/ test-set//index.htm.*

[2] K. Afanasiev and M. Hinze. Adaptive control of a wake flow using proper orthogonal decomposition. *Lecture Notes in Pure and Applied Mathematics, Shape Optimization and Optimal Design, Marcel Dekker*, 216:317–332, 2001.

[3] U. Ascher, R. Mattheij, and R. Russel. *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations.* SIAM, Philadelphia, 1995.

[4] R. Becker, D. Meidner, and B. Vexler. Efficient numerical solution of parabolic optimization problems by finite element methods. *To appear in Optimization Methods and Software*, 2005.

[5] R. Bellman. *Dynamic Programming.* Princeton, N.J.: Princeton University Press, 1957.

[6] C. Bennet. Logical reversibility of computation. *IBM Journal Research and Development*, 17:525–532, 1973.

[7] M. Berggren, R. Glowinski, and J. L. Lions. A computational approach to controllability issues for flow-related models. i: Pointwise control of the viscous burgers equation. *Int. J. Comput. and Fluid Dyn.*, 7(3):237–252, 1996.

[8] M. Berz, C. Bischof, G. Corliss, and A. Griewank, editors. *Computational Differentiation: Techniques, Applications, and Tools.* SIAM Philadelphia, 1996.

[9] J.T. Betts. *Practical Methods for Optimal Control using Nonlinear Programming.* Society for Industrial and Applied Mathematics, Philadelphia, PA., 2001.

[10] P. Blanhard, R. L. Devaney, and G. R. Hall. *Differential Equations*. Brooks/Cole Publishing Company, 1998.

[11] H. G. Bock. Recent advances in parameter identification techniques for ODE. In P. Deufhard and E. Hairer, editors, *Numerical Treatment of Inverse Problems in Differential and Integral Equations*. Birkhäuser, Boston, 1983.

[12] H. G. Bock. Randwertproblemmethoden zur Parameteridentifizierung in Systemen nichtlinearer Differenzialgleichungen. *Bonner Mathematische Schriften*, 183, 1987.

[13] H. G. Bock and K. J. Plitt. A multiple shooting algorithm for direct solution of optimal control problems. In *In Proc. 9th IFAC World Congress Budapest*. Pergamon Press, 1984.

[14] F. Bonnans. 2004. Private Communication.

[15] K.E. Brenan, S.L. Campbell, and L.R. Petzold. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. SIAM, Philadelphia, 1996.

[16] W. L. Briggs, V. E. Henson, and S. F. McCormick. *A Multigrid Tutorial*. Siam, 2000.

[17] A.E. Bryson and Jr. Yu-Chi Ho. *Applied Optimal Control. Optimization, Estimation and Control*. Hemisphere Publishing Corporation, 1975.

[18] C. Büskens and H. Maurer. Real-time control of an industrial robot under control and state constraints. *Online Optimization of Large Scale Systems, Springer-Verlag*, pages 83–92, 2001.

[19] C. Büskens and H. Maurer. Sensitivity analysis and real-time control of parametric nonlinear programming problems. *Online Optimization of Large Scale Systems, Springer-Verlag*, pages 3–16, 2001.

[20] C. Büskens and H. Maurer. Sensitivity analysis and real-time control of parametric optimal control problems using nonlinear programming methods. *Online Optimization of Large Scale Systems, Springer-Verlag*, pages 57–68, 2001.

[21] G. Bärwolff and M. Hinze. Optimization of semiconductor melts. *Submitted to ZAMM*, 2004.

[22] I. Charpentier. Checkpoints schemes for adjoint codes: Application to the meteorological model Meso-NH. Technical report, Project IDOPT, LMC-IMAG, 1998.

[23] W. Cheney and D. Kincaid. *Numerical Mathematic and Computing*. Brooks/Cole Publishing Company Monterey, California, 1980.

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

[25] B. Christianson. A self-stabilizing pantoja-like indirect algorithm for optimal control. *Optimization Methods and Software*, 16:131–149, 2001.

[26] 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.

[27] T. F. Coleman and A. Liao. An efficient trust region method for unconstrained discrete-time optimal control problems. *Computation Optimization and Applications*, 4:47–66, 1995.

[28] J. E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. SIAM, Philadelphia, 1996.

[29] M. Diehl. *Real-Time Optimization for Large Scale Nonlinear Processes*. PhD thesis, University of Heidelberg, 2001.

[30] J. Dunn and D. Bertsekas. Efficient dynamic programming implementations of newton's method for unconstrained optimal control problems. *Journal of Optimization Theory and Applications*, 63:83–92, 1989.

[31] R. England and R. M. Mattheij. Discretizations with dichotomic stability for two-point boundary value problems. In U. M. Ascher and R. D. Russell, editors, *Numerical Boundary Value ODEs, Proceedings of an International Workshop, Vancouver, Canada, July 10-13, 1984*, volume 5 of *Progress in Scientific Computing*, pages 91–107, Boston/Basel/Stuttgart, 1985. Birkhäuser.

[32] Y.G. Evtushenko. Automatic differentiation viewed from optimal control. In A. Griewank and G. Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 25 – 30. SIAM Philadelphia, 1991.

[33] J. Fuhrmann and D. Hömberg. Numerical simulation of surface heat treatments. *Num. Meth. Heat and Fluid Flow*, 9:705–724, 1999.

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

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

[36] A. Griewank. A mathematical view of automatic differentiation. *Acta Numerica*, 12:321–398, 2003.

[37] A. Griewank and G. Corliss, editors. *Automatic Differentiation of Algorithms: Theory, Implementations, and Application*. SIAM Philadelphia, 1991.

[38] 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.

[39] A. Griewank and A. Walther. Revolve: An implementation of checkpointing for the reverse or adjoint mode of computational differentiation. *ACM Trans. Math. Software*, 26:19 – 45, 2000.

[40] J. Grimm, L. Pottier, and N. Rostaing-Schmidt. Optimal time and minimum space-time product for reversing a certain class of programs. In M. Berz, C. Bischof, G. Corliss, and A. Griewank, editors, *Computational Differentiation: Techniques, Applications, and Tools*, pages 95 – 106. SIAM Philadelphia, 1996.

[41] W. W. Hager. Rate of convergence for discrete approximations to unconstrained control problems. *SIAM Journal of Numerical Analysis*, 13:449–471, 1976.

[42] W. W. Hager. Runge-kutta methods in optimal control and the transformed adjoint system. *Numerical Mathematics*, 87:247–282, 2000.

[43] W. W. Hager and R. Rostamian. Optimal coatings bang-bang controls and gradient techniques. *Optimal Control Applications and Methods*, 8:1–20, 1987.

[44] E. Hairer, S.P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I. Nonstiff Problems*. Computational Mechanics, No. 14. Springer-Verlag, Berlin, 2nd revised edition, 1996.

[45] P. Heimbach, C. Hill, and R. Giering. An efficient exact adjoint of the parallel mit general circulation model, generated via automatic differentiation. *Future Generation Computer Systems*, 2003.

[46] M. Hestenes. *Calculus of Variations and Optimal Control.* Robert E. Krieger Publishing Company, Huntington, New York, 1980.

[47] V. Heuveline and A. Walther. Towards the economical computation of adjoints in PDEs using optimal online checkpointing. Technical Report MATH-WR-07-2004, TU Dresden, 2004.

[48] M. Hinze. Optimal and instantaneous control of the instationary Navier-Stokes equations. *Habilitation thesis, Fachbereich Mathematik, Technische Universität Berlin*, 2000.

[49] M. Hinze and J. Sternberg. A-revolve: An adaptive memory- and run-time-reduced procedure for calculating adjoints; with an application to the instationary Navier-Stokes system. MATH-NM-01-2003, TU Dresden, 2003. To appear in Optimization Methods and Software.

[50] 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. *Submitted to Optimal Control, Applications and Methods*, 2005.

[51] D. Hömberg and J. Sokolowski. Optimal control of laser hardening. *Adv. Math. Sci. Appl.*, 8:911–928, 1998.

[52] 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.

[53] D. Hömberg and S. Volkwein. Laser surface hardening using proper orthogonal decomposition for a three dimensional example. *University of Graz and Technical University of Graz*, Report No. 275-June, 2003.

[54] F. De Hoog and R. M. Mattheij. The role of conditioning in shooting techniques. In U. M. Ascher and R. D. Russell, editors, *Numerical Boundary Value ODEs, Proceedings of an International Workshop, Vancouver, Canada, July 10-13, 1984*, volume 5 of *Progress in Scientific Computing*, pages 21–55, Boston/Bael/Stuttgart, 1985. Birkhäuser.

[55] K. Ito and K. Kunisch. Optimal control. *Encyclopedia of Electrical and Electronics Engineering*, 15:364–379, 1999.

[56] D. H. Jacobson and D. Q. Mayne. *Differential Dynamic Programming.* New York: Elsevier, 1970.

[57] M. Jung and U. Langer. *Methode der finiten Elemente für Ingenieure. Eine Einführung in die numerischen Grundlagen und Computersimulation.* B. G. Teubner, Stuttgart/Leipzig/Wiesbaden, 2001.

[58] C. T. Kelley and E. W. Sachs. A pointwise quasi-newton method for unconstrained optimal control problems. *Numerische Mathematik*, 55:159–176, 1989.

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

[60] U. Lehmann. *Schedules for Dynamic Bidirectional Simulations on Parallel Computers.* PhD thesis, Institute of Scientific Computing, TU Dresden, 2003.

[61] G. Leitmann. *The Calculus of Variations and Optimal Control: an Introduction.* Plenum Press, New York and London, 1981.

[62] P. Van Loon. Riccati transformations: When and how to use? In U. M. Ascher and R. D. Russell, editors, *Numerical Boundary Value ODEs, Proceedings of an International Workshop, Vancouver, Canada, July 10-13, 1984*, volume 5 of *Progress in Scientific Computing*, pages 77–91, Boston/Bael/Stuttgart, 1985. Birkhäuser.

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

[64] R. M. Mattheij and F. De Hoog. On non-invertible boundary value problems. In U. M. Ascher and R. D. Russell, editors, *Numerical Boundary Value ODEs, Proceedings of an International Workshop, Vancouver, Canada, July 10-13, 1984*, volume 5 of *Progress in Scientific Computing*, pages 55–77, Boston/Bael/Stuttgart, 1985. Birkhäuser.

[65] 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.

[66] 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.

[67] S. K. Mitter. Successive approximation methods for the solution of optimal control problems. *Automatika*, 3:135–149, 1966.

[68] D. M. Murray and S. J. Jakowitz. Differential dynamic programming and newton's method for discrete optimal control problems. *Journal of Optimization Theory and Applications*, 43(3):395–414, 2001.

[69] S. G. Nash and A. Sofer. *Linear and Nonlinear Programming.* The McGraw-Hill Companies, Inc., 1996.

[70] J. Nocedal and S. Wright. *Numerical Optimization.* Springer, 1999.

[71] H.J. Oberle. BOUNDSCO, Hinweise zur Benutzung des Mehrziel-Verfahrens für die numerische Lösung von Randwertproblemen mit Schaltbedingungen. *Hamburger Beiträge zur Angewandten Mathematik*, Reihe B and Preprint 6, 1987.

[72] H.J. Oberle. Numerical computation of singular-control functions in trajectory optimization problems. *Hamburger Beiträge zur Angewandten Mathematik*, Reihe A and Preprint 12, 1988.

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

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

[75] 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.

[76] A. Quarteroni, R. Sacco, and F. Saleri. *Numerical Mathematics.* Springer, 2000.

[77] H.-G. Roos and H. Schwetlick. *Numerische Mathematik.* B. G. Teubner, Stuttgart/Leipzig, 1999.

[78] R. D. Russell. A unified view of some recent developments in the numerical solution of BVODEs. In U. M. Ascher and R. D. Russell, editors, *Numerical Boundary Value ODEs, Proceedings of an International Workshop, Vancouver, Canada, July 10-13, 1984*, volume 5 of *Progress in Scientific Computing*, pages 1–21, Boston/Bael/Stuttgart, 1985. Birkhäuser.

[79] A. A. Samarskij. *Teorija raznostnych schem.* Izdat. Nauka, Moskva, 1977.

[80] A. A. Samarskij and A. V. Gulin. *Ustojcivost' raznostnych schem.* Izdat. Nauka, Moskva, 1973.

[81] A. A. Samarskij and E. S. Nikolaev. *Metody resenija setocnych uravnenij.* Izdat. Nauka, Moskva, 1978.

[82] J. M. Sanz-Serna and M. P. Calvo. *Numerical Hamiltonian Problems.* Chapman & Hall, 1994.

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

[84] J. Sternberg and A. Griewank. Reduction of storage requirement by checkpointing for time-dependent optimal control problems in ODEs. *To appear in AD 2004 post-conference special collection*, 2004.

[85] J. Sternberg and A. Griewank. Memory-efficient implementation of riccati approach for time-dependent optimal control problems in ODEs. *To appear in GAMM 2005 - Proceedings*, 2005.

[86] J. Stoer and R. Bulirsch. *Einführung in die Numerische Mathematik.* Springer-Verlag, 1978.

[87] Y.M. Volin and G.M. Ostrovskii. Automatic computation of derivatives with use of the multilevel differentiation technique. *Computers and Mathematics with Applications*, 11:1099–1114, 1985.

[88] S. Volkwein. A globalized SQP method for the optimal control of laser surface hardening. *University of Graz and Technical University of Graz*, Report No. 272-June, 2003.

[89] S. Volkwein. Optimal control of laser surface hardening by utilizing a nonlinear primal-dual active set strategy. *University of Graz and Technical University of Graz*, Report No. 277-July, 2003.

[90] S. Volkwein. Nonlinear conjugate gradient methods for the optimal control of laser surface hardening. *Optimization Methods and Software*, 19:179–199, 2004.

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

[92] A. Walther and A. Griewank. Applying the checkpointing routine treeverse to discretizations of burgers' equation. In H.-J. Bungartz, F. Durst, and C. Zenger, editors, *High Performance Scientific and Engineering Computing*, volume 8 of *Lecture Notes in Computational Science and Engineering*, pages 13–24, Berlin/Heidelberg, 1999. Springer.

[93] A. Walther and A. Griewank. Constructing time-minimal parallel reversal schedules. Preprint IOKOMO-2001, Techn. Univ. Dresden, 2001.

[94] A. Walther and A. Griewank. New results on program reversal. In G.F. Corliss, C. Faure, A. Griewank, L. Hascoët, and U. Naumann, editors, *Automatic Differentiation: from Simulation to Optimization*, pages 237–245, New York, 2001. Springer.

[95] A. Walther and A. Griewank. Advantages of binomial checkpointing for memory-reduced adjoint calculations. In M. Feistauer et al., editor, *Numerical mathematics and advanced applications*. Springer, 2004. Proceedings of ENUMATH 2003.

[96] P. Wesseling. *An introduction to multigrid methods*. Wiley, Chichester, 1992.

[97] S. Wright. Interior point methods for optimal control of discrete-time systems. *J. Optim. Theory Appl.*, 77:161–187, 1993.

[98] 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.

Versicherung

Hiermit versichere ich, daß ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher weder im Inland noch im Ausland in gleicher oder ähnlicher Form einer anderen Prüfungsbehörde vorgelegt.

Die Promotionsordnung der Fakultät Mathematik und Naturwissenschaften der Technischen Universität Dresden vom 20. März 2000 erkenne ich hiermit an.

Die vorgelegte Dissertation habe ich am Institut für Wissenschaftliches Rechnen der Technischen Universität Dresden unter der wissenschaftlichen Betreuung von Herrn Prof. Ph.D. Andreas Griewank angefertigt.

Affirmation

I affirm that I have written this dissertation without any inadmissible help from any third person and without recourse to any other aids; all sources are clearly referenced. The dissertation has never been submitted in this or similar form before, neither in Germany nor in any foreign country.

I accept the regulations for the conferral of a doctorate of the Faculty of Mathematics and Science of Dresden University of Technology (published on March, 20th, 2000).

I have written this dissertation at the Institute of Scientific Computing, Technical University Dresden, under the scientific supervision of Prof. Ph.D. Andreas Griewank.

Dresden, den 4. August 2005