Stable Memory-Efficient Nonsmooth Newton Method for Control-State Constrained Optimal Control Problems

Matthias Gerdts [*] Julia Sternberg[†]

July 13, 2010

Abstract

We investigate a stable and memory-efficient nonsmooth Newton method for the numerical solution of optimal control problems subject to mixed control-state constraints. The necessary conditions are stated in terms of a local minimum principle. By use of the Fischer-Burmeister function the local minimum principle is transformed into an equivalent nonlinear and nonsmooth equation in appropriate Banach spaces. This nonlinear and nonsmooth equation is solved by a nonsmooth Newton method. We present how this method can be realized by a multiple shooting approach, so that fast growing and fast decaying modes are decoupled during the integration, which results in a stable version of the nonsmooth Newton method. Each iteration of the Newton method contains three alternative sweeps through a time horizon, with a specific information dependence between different sweeps, so that the straightforward implementation of the algorithm would require a huge amount of memory to store all intermediate variables. To reduce this memory requirement we discuss in the present paper checkpointing techniques and prove some new theoretical results concerning them. Numerical experiences are discussed at the end of the article.

Key words: Optimal control, nonsmooth Newton method, control-state constraints, nested checkpointing technique

1 Introduction

We consider the following optimal control problem subject to mixed control-state constraints:

$$(OCP)\qquad \begin{array}{ll} \text{Minimize} & \displaystyle\int_0^1 f_0(x(t),u(t))dt \\ \text{w.r.t.} & x \in W^{1,\infty}([0,1],\mathbb{R}^{n_x}), u \in L^{\infty}([0,1],\mathbb{R}^{n_u}), \\ \text{s.t.} & x'(t) = f(x(t),u(t)) \text{ a.e. in } [0,1], \\ & \psi(x(0),x(1)) = 0, \\ & c(x(t),u(t)) \leq 0 \text{ a.e. in } [0,1]. \end{array}$$

[*]Institute of Mathematics, University of Würzburg, Am Hubland, 97074 Würzburg, Germany, gerdts@mathematik.uni-wuerzburg.de

[†]Department of Mathematics, University of Hamburg, Bundesstrasse 55, 20146 Hamburg, Germany, julia.sternberg@uni-hamburg.de

Without loss of generality the discussion is restricted to autonomous problems on the fixed time interval $[0,1]$. The functions $f_0 : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$, $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$, $\psi : \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \to \mathbb{R}^{n_\psi}$, $c : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_c}$, are supposed to be at least twice continuously differentiable w.r.t. to all arguments. As usual, the Banach space $L^\infty([0,1],\mathbb{R}^n)$ consists of all measurable functions $h : [0,1] \to \mathbb{R}^n$ with

$$\|h\|_\infty := \operatorname*{ess\ sup}_{0\le t\le 1} \|h(t)\| < \infty,$$

where $\|\cdot\|$ denotes the Euclidean norm on $\mathbb{R}^n$. The Banach space $W^{1,\infty}([0,1],\mathbb{R}^n)$ consists of all absolutely continuous functions $h : [0,1] \to \mathbb{R}^n$ with

$$\|h\|_{1,\infty} := \max\{\|h\|_\infty, \|h'\|_\infty\} < \infty.$$

Several approaches towards the numerical solution of OCP have been investigated in the literature. The so-called direct discretization method is based on a discretization of the infinite dimensional optimal control problem and leads to a finite dimensional nonlinear program, cf., e.g., Gerdts [6]. The latter can be solved numerically by suitable programming methods such as, e.g., sequential quadratic programming. The direct discretization method turns out to be very robust in practice. Nevertheless, the computational effort grows at a nonlinear rate with the number of grid points used for discretization.

The so-called indirect method for optimal control problems attempts to satisfy the necessary conditions that are provided by the well-known minimum principle numerically, cf., e.g. Oberle and Grimm [19]. The exploitation of the minimum principle leads to a nonlinear multi-point boundary value problem that has to be solved. Although the indirect method usually leads to the most accurate solutions, it suffers from the drawback that it requires a good initial guess in order to converge. One crucial task is to estimate the sequence of active and inactive intervals of the control-state constraint.

Our intention is to employ an alternative method – the nonsmooth Newton method. This method was investigated in finite dimensions amongst others by Qi [20] and Qi and Sun [21]. Extensions to infinite spaces can be found in Kummer [13, 14], Chen et al. [2], and Ulbrich [23, 24]. The method for the problem class OCP was introduced in Gerdts [7] and it was further analyzed in Gerdts [8]. Section 2 summarizes the nonsmooth Newton method for the problem class OCP. All results concerning convergence properties of the nonsmooth Newton method are taken from [8] with proofs omitted. Therefore, Section 2 and parts of Section 3 have to be understood merely as a review of the main results in [8]. Nevertheless, as these results are necessary to understand the advanced techniques introduced in Section 3 we decided to include them. The main intention of this paper is to combine the nonsmooth Newton method from [8] with an efficient and stable multiple shooting method based on marching techniques for the solution of the occurring boundary value problems in each step of the algorithm. Detailed explanation for marching techniques can be found in e.g. book from Ascher et al. [1]. For better understanding of the resulting algorithm, we give a short introduction of marching method, which is closely related to this book. Unlike the simple technique used in [8], the combination between the nonsmooth Newton method and stable multiple shooting method based on marching techniques is particularly suitable for boundary value problems with fast growing and fast decaying modes.

A brief outline of the nonsmooth Newton method is as follows. The reformulation of the necessary conditions for OCP leads to the nonsmooth equation

$$\mathcal{F}(z) = 0, \qquad \mathcal{F} : Z \to Y,$$

where Z and Y are appropriate Banach spaces. Application of the nonsmooth Newton method generates sequences $\{z^k\}$ and $\{d^k\}$ related by the iteration

$$z^{k+1} = S_k(z^k + d^k), \qquad k = 0, 1, 2, \ldots.$$

Herein, the search direction d^k is the solution of the linear operator equation $V_k(d^k) = -\mathcal{F}(z^k)$ with the linear operator V_k chosen from an appropriately defined generalized Jacobian $\partial_* \mathcal{F}(z^k)$. The smoothing step S_k is needed to map the search direction into the correct space and to enforce semismoothness.

Section 3 addresses the linear operator equation for the search direction which corresponds to a linear boundary value problem for the state and adjoint variables. A stable multiple shooting method based on marching techniques is used to solve the boundary value problem. Moreover, Section 4 presents a checkpointing approach, which can be applied for the memory-reduced implementation of the stable multiple shooting method. In this section we introduce and prove a new representation of minimal evaluation cost. This gives us a better understanding of checkpointing techniques. In Section 5 a numerical example is presented, which illustrates advantages of the stable nonsmooth Newton method. Here we apply this method to an optimal control problem with dichotomy property. Finally, in Section 6 we give some conclusions.

2 Nonsmooth Newton Method

In the following we give a brief outline of the nonsmooth Newton method. A detailed description is given in [8] by the first author.

The (augmented) Hamilton function $H : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_c} \to \mathbb{R}$ is defined by

$$H(x, u, \lambda, \eta) := f_0(x, u) + \lambda^\top f(x, u) + \eta^\top c(x, u).$$

We summarize the well-known minimum principle for OCP, compare Malanowski [16], Theorem 4.3. Let (x_*, u_*) be a (weak) local minimum of OCP and, in addition to the smoothness assumptions made above, let the following assumptions be satisfied at (x_*, u_*):

(i) Linear independence: There exist $\alpha > 0$ and $\beta > 0$ such that

$$\|c'_{I_\alpha(t),u}[t]^\top \zeta\| \geq \beta \|\zeta\|$$

for all ζ of appropriate dimension. Herein, the index set I_α is defined by $I_\alpha(t) := \{i \in \{1, \ldots, n_c\} \mid c_i[t] \geq -\alpha\}$.

(ii) Controllability: For every $q \in \mathbb{R}^{n_\psi}$ there exists a solution (x, u, σ) of the linear system

$$\begin{aligned} x'(t) - f'_x[t]x(t) - f'_u[t]u(t) &= 0, \\ \psi'_{x_0}x(0) + \psi'_{x_1}x(1) &= q, \\ c'_x[t]x(t) + c'_u[t]u(t) + S_\alpha(t)\sigma(t) &= 0, \end{aligned}$$

where $S_\alpha(t) := \operatorname{diag}(c_{i,\alpha}(t))$ and $c_{i,\alpha}(t) := \min\{c_i[t] + \alpha, 0\}$.

Then, there exist Lagrange multipliers $\lambda_* \in W^{1,\infty}([0,1],\mathbb{R}^{n_x})$, $\eta_* \in L^\infty([0,1],\mathbb{R}^{n_c})$, and $\sigma_* \in \mathbb{R}^{n_\psi}$ with

$$\begin{aligned}
x_*'(t) - f(x_*(t),u_*(t)) &= 0, && (1)\\
\lambda_*'(t) + H_x'(x_*(t),u_*(t),\lambda_*(t),\eta_*(t))^\top &= 0, && (2)\\
\psi(x_*(0),x_*(1)) &= 0, && (3)\\
\lambda_*(0) + \psi_{x_0}'(x_*(0),x_*(1))^\top\sigma_* &= 0, && (4)\\
\lambda_*(1) - \psi_{x_1}'(x_*(0),x_*(1))^\top\sigma_* &= 0, && (5)\\
H_u'(x_*(t),u_*(t),\lambda_*(t),\eta_*(t))^\top &= 0. && (6)
\end{aligned}$$

Furthermore, the complementarity conditions hold a.e. in $[0,1]$:

$$\eta_*(t) \geq 0, \qquad c(x_*(t),u_*(t)) \leq 0, \qquad \eta_*(t)^\top c(x_*(t),u_*(t)) = 0. \tag{7}$$

Let the Banach spaces

$$\begin{aligned}
Z &= W^{1,\infty}([0,1],\mathbb{R}^{n_x}) \times L^\infty([0,1],\mathbb{R}^{n_u}) \times L^\infty([0,1],\mathbb{R}^{n_x}) \times L^\infty([0,1],\mathbb{R}^{n_c}) \times \mathbb{R}^{n_\psi},\\
Y_1 &= L^\infty([0,1],\mathbb{R}^{n_x}) \times L^\infty([0,1],\mathbb{R}^{n_x}) \times \mathbb{R}^{n_\psi} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \times L^\infty([0,1],\mathbb{R}^{n_u}),\\
Y_2 &= L^\infty([0,1],\mathbb{R}^{n_c})
\end{aligned}$$

be equipped with the maximum norm for product spaces and $z_* = (x_*,u_*,\lambda_*,\eta_*,\sigma_*)$. By componentwise application of the convex and locally Lipschitz continuous Fischer-Burmeister function $\varphi : \mathbb{R}^2 \to \mathbb{R}$, see Fischer [4], defined by

$$\varphi(a,b) := \sqrt{a^2+b^2} - a - b, \tag{8}$$

to the complementarity conditions, the necessary conditions (1)-(7) are equivalent with the nonlinear equation

$$\mathcal{F}(z_*) = \begin{pmatrix} \mathcal{F}_1(z_*) \\ \mathcal{F}_2(z_*) \end{pmatrix} = 0, \tag{9}$$

where $\mathcal{F}_1 : Z \to Y_1$ and $\mathcal{F}_2 : Z \to Y_2$ denote the smooth and the nonsmooth part of $\mathcal{F} : Z \to Y := Y_1 \times Y_2$, respectively:

$$\mathcal{F}_1(z)(\cdot) := \begin{pmatrix} x'(\cdot) - f(x(\cdot),u(\cdot)) \\ \lambda'(\cdot) + H_x'(x(\cdot),u(\cdot),\lambda(\cdot),\eta(\cdot))^\top \\ \psi(x(0),x(1)) \\ \lambda(0) + \psi_{x_0}'(x(0),x(1))^\top\sigma \\ \lambda(1) - \psi_{x_1}'(x(0),x(1))^\top\sigma \\ H_u'(x(\cdot),u(\cdot),\lambda(\cdot),\eta(\cdot))^\top \end{pmatrix}, \quad \mathcal{F}_2(z)(\cdot) := \omega(z(\cdot)), \tag{10}$$

where $\omega = (\omega_1,\ldots,\omega_{n_c})^\top : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_c} \times \mathbb{R}^{n_\psi} \to \mathbb{R}^{n_c}$ and

$$\omega_i(\bar{x},\bar{u},\bar{\lambda},\bar{\eta},\bar{\sigma}) := \varphi(-c_i(\bar{x},\bar{u}),\bar{\eta}_i), \qquad i = 1,\ldots,n_c. \tag{11}$$

Notice, that the Fischer-Burmeister function has the nice property that $\varphi(a,b) = 0$ holds if and only if $a,b \geq 0$ and $ab = 0$. Hence, the complementarity conditions (7) are equivalent with the equality

$$\varphi(-c_i(x_*(t),u_*(t)),\eta_{i,*}(t)) = 0, \qquad i = 1,\ldots,n_c,$$

that has to hold almost everywhere in $[0, 1]$.

The standard approach to solve (9) numerically would be to apply the classical Newton method. Unfortunately, the derivative $\mathcal{F}'(z^k)$ does not exist since the component $\mathcal{F}_2$ is not differentiable due to the non-differentiability of φ at the origin. Rather than working with the derivative of φ, which does not exist at the origin, we will work with Clarke's generalized Jacobian of φ:

$$\partial\varphi(a,b) = \begin{cases} \left\{\left(\frac{a}{\sqrt{a^2+b^2}} - 1, \frac{b}{\sqrt{a^2+b^2}} - 1\right)\right\}, & \text{if } (a,b) \neq (0,0), \\ \left\{(s,r) \mid (s+1)^2 + (r+1)^2 \leq 1\right\}, & \text{if } (a,b) = (0,0). \end{cases}$$

Notice, that $\partial\varphi(a,b)$ is a nonempty, convex and compact set. Motivated by the chain rule in finite dimensions we define the point to set mapping $\partial_*\mathcal{F} : Z \rightrightarrows \mathcal{L}(Z,Y)$ according to

$$\partial_*\mathcal{F}(z^k)(z) := \left\{ \begin{pmatrix} \mathcal{F}_1'(z^k)(z) \\ -S\left(c_x'[\cdot]x + c_u'[\cdot]u\right) + R\eta \end{pmatrix} \;\middle|\; \begin{array}{l} S = diag(s_1, \ldots, s_{n_c}), \\ R = diag(r_1, \ldots, r_{n_c}), \\ (s_i, r_i) \in \partial\varphi[\cdot] \text{ a.e.}, \\ s_i(\cdot), r_i(\cdot) \text{ measurable} \end{array} \right\}$$

and use this set as a generalized Jacobian. The same idea was introduced earlier in Ulbrich [23], Def. 3.35, p. 47. Notice, that the first component $\mathcal{F}_1$ of $\mathcal{F}$ in (10) is continuously Fréchet-differentiable with

$$\mathcal{F}_1'(z^k)(z) = \begin{pmatrix} x'(\cdot) - f_x'[\cdot]x(\cdot) - f_u'[\cdot]u(\cdot) \\ \lambda'(\cdot) + H_{xx}''[\cdot]^\top x(\cdot) + H_{xu}''[\cdot]^\top u(\cdot) + H_{x\lambda}''[\cdot]^\top \lambda(\cdot) + H_{x\eta}''[\cdot]^\top \eta(\cdot) \\ \psi_{x_0}' x(0) + \psi_{x_1}' x(1) \\ \lambda(0) + \psi_{x_0x_0}''(\sigma^k, x(0)) + \psi_{x_0x_1}''(\sigma^k, x(1)) + \left(\psi_{x_0}'\right)^\top \sigma \\ \lambda(1) - \psi_{x_1x_0}''(\sigma^k, x(0)) - \psi_{x_1x_1}''(\sigma^k, x(1)) - \left(\psi_{x_1}'\right)^\top \sigma \\ H_{ux}''[\cdot]^\top x(\cdot) + H_{uu}''[\cdot]^\top u(\cdot) + H_{u\lambda}''[\cdot]^\top \lambda(\cdot) + H_{u\eta}''[\cdot]^\top \eta(\cdot) \end{pmatrix},$$

provided that the functions f_0, f, c, ψ are twice continuously differentiable w.r.t. all arguments. All functions are evaluated at $z^k = (x^k, u^k, \lambda^k, \eta^k, \sigma^k) \in Z$.

Replacing the non-existing Jacobian $\mathcal{F}'$ in the classical Newton method by the generalized Jacobian $\partial_*\mathcal{F}(z^k)$ leads to the following algorithm, which in addition makes use of a smoothing operator S_k, that is necessary to achieve fast local convergence.

Algorithm 2.1 Local Nonsmooth Newton Method

0. Choose z^0.

1. If some stopping criterion is satisfied, stop.

2. Choose an arbitrary $V_k \in \partial_*\mathcal{F}(z^k)$ and compute the search direction d^k from the linear equation

$$V_k(d^k) = -\mathcal{F}(z^k).$$

3. Set $z^{k+1} = S_k(z^k + d^k)$, $k = k+1$, and go to 1.

Gerdts [8], Theorems 2.5, 4.5, and 4.6, analyzed the local and global convergence properties of the nonsmooth Newton method. However, it is an open problem how to construct the smoothing operator S_k in general. In certain settings, it is known, how to construct S_k using a projection operator, compare Section 4.1 in Ulbrich [23]. Due to the lack of a convenient smoothing operator in our general setting, we cannot give a strict proof for a locally superlinear convergence rate of the nonsmooth Newton method. However, despite of this theoretical shortcoming, we fortunately observe fast local convergence behaviour in our numerical investigations. Sufficient conditions for the existence of the inverse operator V_k^{-1} and the uniform boundedness were established in Theorem 3.6 in Gerdts [8] and are summarized for the reader's convenience in Theorem 3.1 below. In the sequel we particularly focus on an efficient method to solve the operator equation $V_k(d) = -\mathcal{F}(z^k)$ in step (2) of Algorithm 2.1.

3 Computation of the Search Direction

For brevity we neglect the arguments whenever possible. In the sequel, every function is evaluated at the current iterate $z^k = (x^k, u^k, \lambda^k, \eta^k, \sigma^k)$. Let $d = (x, u, \lambda, \eta, \sigma)$ denote the solution of the linear operator equation $V_k(d) = -\mathcal{F}(z^k)$ in step (2) of Algorithm 2.1. Our aim is to derive for the components $y = (x, \lambda, \sigma)$ of d the linear boundary value problem

$$y'(t) = A(t)y(t) + q(t), \qquad \text{a.e. in } [0,1], \tag{12}$$

$$B_0 y(0) + B_1 y(1) = \beta. \tag{13}$$

The dynamic components of the linear operator equation $V_k(d) = -\mathcal{F}(z^k)$ in step (2) of Algorithm 2.1 read as

$$\begin{pmatrix} x' \\ \lambda' \\ \sigma' \end{pmatrix} - \begin{pmatrix} f'_x & 0 & 0 \\ -{H''_{xx}}^\top & -{H''_{x\lambda}}^\top & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ \lambda \\ \sigma \end{pmatrix} - \begin{pmatrix} f'_u & 0 \\ -{H''_{xu}}^\top & -{H''_{x\eta}}^\top \\ 0 & 0 \end{pmatrix} \begin{pmatrix} u \\ \eta \end{pmatrix}$$
$$= - \begin{pmatrix} (x^k)' - f \\ (\lambda^k)' + {H'_x}^\top \\ 0 \end{pmatrix} \tag{14}$$

and

$$\mathcal{A} \begin{pmatrix} u \\ \eta \end{pmatrix} + \begin{pmatrix} {H''_{ux}}^\top & {H''_{u\lambda}}^\top \\ -Sc'_x & 0 \end{pmatrix} \begin{pmatrix} x \\ \lambda \end{pmatrix} = - \begin{pmatrix} {H'_u}^\top \\ \omega(z^k(\cdot)) \end{pmatrix}, \tag{15}$$

where

$$\mathcal{A} := \begin{pmatrix} {H''_{uu}}^\top & {c'_u}^\top \\ -Sc'_u & R \end{pmatrix}. \tag{16}$$

Notice, that we added in (14) the differential equation $\sigma' = 0$ for the constant σ. Moreover, the remaining equations of the operator equation $V_k(d) = -\mathcal{F}(z^k)$ yield the boundary condition (13), where

$$B_0 = \begin{pmatrix} \psi'_{x_0} & 0 & 0 \\ ({\psi'_{x_0}}^\top \sigma^k)'_{x_0} & I & {\psi'_{x_0}}^\top \\ -({\psi'_{x_1}}^\top \sigma^k)'_{x_0} & 0 & -{\psi'_{x_1}}^\top \end{pmatrix}, \quad B_1 = \begin{pmatrix} \psi'_{x_1} & 0 & 0 \\ ({\psi'_{x_0}}^\top \sigma^k)'_{x_1} & 0 & 0 \\ -({\psi'_{x_1}}^\top \sigma^k)'_{x_1} & I & 0 \end{pmatrix},$$

and

$$\beta = -\begin{pmatrix} \psi(x^k(0), x^k(1)) \\ \lambda^k(0) + {\psi'_{x_0}}^\top \sigma^k \\ \lambda^k(1) - {\psi'_{x_1}}^\top \sigma^k \end{pmatrix}.$$

Under the conditions of Theorem 3.2 in Gerdts [8] the inverse operator $\mathcal{A}^{-1}$ exists and it is bounded. Then equation (15) can be solved for u and η according to

$$\begin{pmatrix} u \\ \eta \end{pmatrix} = -\mathcal{A}^{-1}\left[\begin{pmatrix} {H''_{ux}}^\top & {H''_{u\lambda}}^\top \\ -Sc'_x & 0 \end{pmatrix}\begin{pmatrix} x \\ \lambda \end{pmatrix} + \begin{pmatrix} {H'_u}^\top \\ \omega(z^k(\cdot)) \end{pmatrix}\right]. \tag{17}$$

Introducing this expression into the differential equation (14) yields the differential equation (12) with

$$A(t) = \left[\begin{pmatrix} f'_x & 0 & 0 \\ -{H''_{xx}}^\top & -{H''_{x\lambda}}^\top & 0 \\ 0 & 0 & 0 \end{pmatrix} - \begin{pmatrix} f'_u & 0 \\ -{H''_{xu}}^\top & -{H''_{x\eta}}^\top \\ 0 & 0 \end{pmatrix}\mathcal{A}^{-1}\begin{pmatrix} {H''_{ux}}^\top & {H''_{u\lambda}}^\top & 0 \\ -Sc'_x & 0 & 0 \end{pmatrix}\right]$$

and

$$q(t) = -\left[\begin{pmatrix} (x^k)' - f \\ (\lambda^k)' + {H'_x}^\top \\ 0 \end{pmatrix} + \begin{pmatrix} f'_u & 0 \\ -{H''_{xu}}^\top & -{H''_{x\eta}}^\top \\ 0 & 0 \end{pmatrix}\mathcal{A}^{-1}\begin{pmatrix} {H'_u}^\top \\ \omega(z^k(\cdot)) \end{pmatrix}\right].$$

Theorem 3.6 in Gerdts [8] provides sufficient conditions for the solvability of the operator equation $V_k(d) = -\mathcal{F}$ and the linear boundary value problem (12)-(13), respectively.

Theorem 3.1 *Let the following assumptions be satisfied with uniform constants for all z in some neighborhood $U_\Delta(z_*)$, $\Delta > 0$, of a zero z_* of $\mathcal{F}$. All functions below are evaluated at z. Define the index sets*

$$\begin{aligned} I_>(t) &:= \{i \in \{1,\dots,n_c\} \mid c_i[t] = 0,\ \eta_i(t) > 0\}, \\ J_\gamma(t) &:= \{i \in \{1,\dots,n_c\} \mid |c_i[t]| \le \gamma\eta_i(t),\ \eta_i(t) \ge 0\}, \qquad \gamma > 0. \end{aligned}$$

(i) Let there exist a constant C such that a.e. in $[0,1]$ it holds $\|A(t)\| \le C$.

(ii) Let there exist $\kappa > 0$ such that for all $\zeta \in \mathbb{R}^{2n_x+n_\psi}$ it holds

$$\|\left(B_0\Phi(0) + B_1\Phi(1)\right)\zeta\| \ge \kappa\|\zeta\|,$$

where Φ is a fundamental solution with $\Phi'(t) = A(t)\Phi(t)$, $\Phi(0) = I$.

(iii) Let there exist constants C_1, C_2, C_3 such that a.e. in $[0,1]$ it holds

$$\|H''_{uu}[t]\| \le C_1, \quad \|c'_u[t]^\top\| \le C_2, \quad \|c'_u[t]\| \le C_3.$$

(iv) (Coercivity) Let there exist a constant $\alpha > 0$ such that a.e. in $[0,1]$ it holds

$$d^\top H''_{uu}[t]d \ge \alpha\|d\|^2 \quad \text{for all} \quad d \in \mathbb{R}^{n_u} \ :\ c'_{I_>(t),u}[t]d = 0.$$

(v) (Linear independence) Let there exist constants $\gamma > 0$ and $\delta > 0$ such that a.e. in $[0,1]$ it holds

$$\|c'_{J_\gamma(t),u}[t]^\top\zeta\| \ge \delta\|\zeta\| \quad \text{for all } \zeta \text{ of appropriate dimension.}$$

Then, the generalized Jacobian $\partial_* F(z)$ is nonsingular and there exists a constant $K > 0$ such that $\|V^{-1}\|_{\mathcal{L}(Y,Z)} \leq K$ for every $V \in \partial_* F(z)$. In particular, the solution $d = (x, u, \lambda, \eta, \sigma)$ of the operator equation $V_k(d) = -\mathcal{F}(z^k)$ in step (2) of Algorithm 2.1 is given by the solution $y = (x, \lambda, \sigma)$ of the linear boundary value problem (12)-(13) and (u, η) from (17).

Remark 3.2 *Conditions (iii)-(v) guarantee the existence of the inverse operator $\mathcal{A}^{-1}$ and its uniform boundedness, see Gerdts [8], Theorem 3.2. If the operator $\mathcal{A}$ is not invertible, the situation becomes more involved. In this case, equation (15) imposes algebraic constraints and the equations (14) and (15) form a differential algebraic equation (DAE) with an index of at least two. Actually, the case when $\mathcal{A}$ is invertible corresponds to the index one case.*

Moreover, assume that the system matrix $A(t)$ in (12) contains both fast growing and fast decaying modes.

In contrast to applications in [8], where the BVP (12)-(13) is solved using the forward Euler's method coupled with the single shooting, we assume here a situation more involved: We assume that in (12)-(13) underlying IVPs are instable in such a manner that they contain both fast growing and fast decaying modes. From this reason, it is not possible to solve the BVP (12)-(13) using such straight forward techniques as single shooting method. In situations like this, it is reasonable to apply appropriate algorithms, that maintain dichotomy properties of the BVP. Such a numerical method has to be capable to accurately split the time integration of IVPs, so that fast decaying and fast growing modes can be separately integrated forward in time and backward in time, respectively. Multiple shooting method combined with a decoupling is one of the methods that make this splitting possible. Furthermore, the realization of this method can be arranged in such a way, that a memory efficient implementation using checkpointing is possible. A detailed explanation of this procedure is given in Section 4 of this paper. In the present section we restrict ourselves on the explanation of the decoupled multiple shooting algorithm.

The description of the multiple shooting method and decoupling techniques follows the book of Ascher, Mattheij and Russel [1]. We subdivide the interval $[0, 1]$ by a mesh $0 = t_0 < t_1 < ... < t_\ell = 1$ and consider the initial value problems for $0 \leq i \leq \ell - 1$,

$$\begin{aligned} y_i'(t) &= A(t)y_i(t) + q(t), \qquad t_i < t < t_{i+1}, &(18)\\ y_i(t_i) &= \mathbf{F}_i s_i. &(19)\end{aligned}$$

Herein, $\mathbf{F}_i \in \mathbb{R}^{n_y \times n_y}$, $i = 0, \ldots, \ell - 1$, are nonsingular matrices to be determined later on. According to the superposition principle, for each $0 \leq i \leq \ell - 1$ the general solution of the differential equation (18) is given by

$$y_i(t) = Y_i(t)s_i + v_i(t), \qquad t_i \leq t \leq t_{i+1}, \tag{20}$$

where $s_i \in \mathbb{R}^{n_y}$ is a parameter vector, v_i is a particular solution of

$$v_i'(t) = A(t)v_i(t) + q(t), \qquad v_i(t_i) = 0, \qquad t_i < t \leq t_{i+1}, \tag{21}$$

and Y_i is a fundamental solution solving the homogeneous equation

$$Y_i'(t) = A(t)Y_i(t), \qquad Y_i(t_i) = \mathbf{F}_i, \qquad t_i < t \leq t_{i+1}. \tag{22}$$

The unknowns to be found are the parameters

$$s^\top = (s_0^\top, s_1^\top, ..., s_{\ell-1}^\top). \tag{23}$$

They are to be determined, so that the solution is continuous over the entire interval $[0, 1]$. Moreover, the boundary conditions (13) are to be satisfied. Thus, the desired solution is defined by

$$y(t) := y_i(t), \qquad t_i \le t \le t_{i+1}, \qquad 0 \le i \le \ell - 1, \tag{24}$$

where the requirements on s are

$$y_i(t_{i+1}) - y_{i+1}(t_{i+1}) = 0, \quad i = 0, \ldots, \ell - 2, \quad B_0 y_0(0) + B_1 y_{\ell-1}(1) = \beta.$$

Using (20) we obtain

$$Y_i(t_{i+1}) s_i + v_i(t_{i+1}) - \mathbf{F}_{i+1} s_{i+1} = 0, \qquad 0 \le i \le \ell - 2, \tag{25}$$

and

$$B_0 \mathbf{F}_0 s_0 + B_1 \left(Y_{\ell-1}(1) s_{\ell-1} + v_{\ell-1}(1) \right) = \beta. \tag{26}$$

The question which arises now is how to determine appropriate $\mathbf{F}_i$ in (19). Most common choice for $\mathbf{F}_i$ is $\mathbf{F}_i = I$. In the dichotomy case assumed here, so that the matrix $A(t)$ contains both fast growing and fast decaying modes, it is useful to apply marching schemes, that form the initial matrix $\mathbf{F}_{i+1}$ for the $(i+1)$th subinterval after the IVP (18) - (19) for the ith subinterval has been solved. Such schemes produce a stable multiple shooting algorithm. The initial matrix $\mathbf{F}_{i+1}$ can be determined e.g. by the QU-decomposition of $Y_i(t_{i+1})$ in the following way

$$Y_i(t_{i+1}) = \mathbf{F}_{i+1} G_i, \qquad 0 \le i \le \ell - 1, \tag{27}$$

with $\mathbf{F}_{i+1}$ orthogonal and G_i upper triangular matrices. Using Lemma 4.53 in [1] we can conclude that if $\mathbf{F}_0$ is chosen, so that its first k columns represent initial values of the most dominant modes, then, the span of the first k columns of $\mathbf{F}_{i+1}$, computed by the relation (27) for each i, $0 \le i \le \ell - 1$, represents the directions of their integrated form up to $t = t_{i+1}$. This means that the upper left $k \times k$ block of $\hat{G}_i = G_i \ldots G_0$ has a norm of the same order of magnitude as the increment of these dominant modes. Similar observations can be applied 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.

Utilizing this fact we can successfully solve the linear system (25)-(26) combining compactification and decoupling, where the decoupling corresponds to the splitting of growing and decaying modes. With (27), the linear system (25)-(26) is given by

$$\begin{bmatrix} -G_0 & I & & & \\ & -G_1 & I & & \\ & & \ddots & \ddots & \\ & & & -G_{l-2} & I \\ B_0 \mathbf{F}_0 & & & & B_1 \mathbf{F}_\ell G_{\ell-1} \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ \vdots \\ s_{\ell-2} \\ s_{\ell-1} \end{bmatrix} = \begin{bmatrix} \tilde{\beta}_0 \\ \tilde{\beta}_1 \\ \vdots \\ \tilde{\beta}_{\ell-2} \\ \tilde{\beta}_{\ell-1} \end{bmatrix}, \tag{28}$$

where $\tilde{\beta}_i = \mathbf{F}_{i+1}^\top v_i(t_{i+1})$, $i = 0, \ldots, \ell-2$, and $\tilde{\beta}_{\ell-1} = \beta - B_1 v_{\ell-1}(1)$. In order to solve equation (28), we express s_i as

$$s_i = \Phi_i s_0 + \xi_i, \qquad i = 0, \ldots, \ell - 1, \tag{29}$$

with suitable $\Phi_i \in \mathbb{R}^{n_y \times n_y}$ and $\xi_i \in \mathbb{R}^{n_y}$. From

$$s_{i+1} = G_i s_i + \tilde{\beta}_i = G_i \Phi_i s_0 + G_i \xi_i + \tilde{\beta}_i = \Phi_{i+1} s_0 + \xi_{i+1}$$

we obtain the recursions

$$\Phi_{i+1} = G_i\Phi_i, \qquad \Phi_0 = I, \qquad \xi_{i+1} = G_i\xi_i + \tilde{\beta}_i, \qquad \xi_0 = 0, \qquad i = 0, \dots, \ell - 1.$$

Using (29) it suffices to determine s_0 from the reduced linear equation

$$(B_0\mathbf{F}_0\Phi_0 + B_1\mathbf{F}_\ell\Phi_\ell)\, s_0 = \tilde{\beta}_{\ell-1} - B_0\mathbf{F}_0\xi_0 - B_1\mathbf{F}_\ell G_{\ell-1}\xi_{\ell-1}$$

in order to solve the linear system (28).

Thus, the stable approach to solve the linear boundary value problem (12) - (13) using the compactification results in the following conceptual algorithm:

Algorithm 3.3 Stable multiple shooting

Choose the mesh $\pi : 0 = t_0 < t_1 < ... < t_{\ell-1} < t_\ell = 1$.
Initialize $y_i^0 = s_i^0$, $i = 0, ..., \ell - 1$.
Primal Initialization:
$\mathbf{F}_0 = I$, $\xi_0^2 = 0$, $\Phi_0^{22} = I_{n-k}$.
For $i = 0, 1, ..., \ell - 1$ **DO**
Primal Step $(\mathbf{i+1})$**:**
Solve $Y_i' = AY_i, \qquad t_i < t < t_{i+1}, \qquad Y_i(t_i) = \mathbf{F}_i$.
Solve $v_i' = Av_i + q, \qquad t_i < t < t_{i+1}, \qquad v_i(t_i) = 0$.
Find $\mathbf{F}_{i+1}$ *and* G_i *from* $Y_i(t_{i+1}) = \mathbf{F}_{i+1}G_i$, $G_i = \begin{pmatrix} D_i & C_i \\ 0 & E_i \end{pmatrix}$.
Find $\tilde{\beta}_i = \begin{pmatrix} \tilde{\beta}_i^1 \\ \tilde{\beta}_i^2 \end{pmatrix} = \mathbf{F}_{i+1}^\top v_i(t_{i+1})$.
Find ξ_{i+1}^2 *and* Φ_{i+1}^{22} *from* $\xi_{i+1}^2 = E_i\xi_i^2 + \tilde{\beta}_i^2$, $\Phi_{i+1}^{22} = E_i\Phi_i^{22}$.
Adjoint Initialization:
$\xi_\ell^1 = 0$, $\Phi_\ell^{11} = I_k$, $\Phi_\ell^{12} = 0$, $\Phi_i^{21} = 0$ *for all* i.
For $i = \ell - 1, \ell - 2, ..., 0$ **DO**
Adjoint Step $(\mathbf{i+1})$**:**
$\xi_i^1 = D_i^{-1}(\xi_{i+1}^1 - C_i\xi_i^2 - \tilde{\beta}_i^1)$.
$\Phi_i^{11} = D_i^{-1}\Phi_{i+1}^{11}$.
$\Phi_i^{12} = D_i^{-1}(\Phi_{i+1}^{12} - C_i\Phi_i^{22})$.
Final Initialization:
$M := B_0\mathbf{F}_0\Phi_0 + B_1\mathbf{F}_\ell\Phi_\ell$.
$\mu := \beta - B_1v_{\ell-1}(1) - B_0\mathbf{F}_0\xi_0 - B_1\mathbf{F}_\ell G_{\ell-1}\xi_{\ell-1}$.
Solve $M\tau = \mu$ *for* τ.
For $i = 0, 1, ..., \ell - 1$ **DO**
Final Step $(\mathbf{i+1})$**:**
$y_i = \mathbf{F}_i(\Phi_i\tau + \xi_i)$.
Solve (17) to determine u *and* η *on* $[t_i, t_{i+1}]$

Algorithm 3.3 consists of three sweeps through the time window $[0, 1]$, i.e. the primal, the adjoint and the final sweep. Once the parameter τ in the **Final Initialization** is determined, the variables y_i, u and η can be computed in any order, provided the matrices $\mathbf{F}_i$, Φ_i and the vector ξ_i are available.

The simplest strategy is to implement Algorithm 3.3 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 this basic algorithm, where all intermediate values are stored, is of order $\mathcal{O}(\ell\, n_y^2)$, where ℓ gives the number of time steps between 0 and 1; n_y is the dimension of the vector y. If ℓ and n_y are significant (i.e. resulting from the spatial discretization of PDEs) this approach can be realized only when there is a sufficiently large amount of memory available. If this is not the case, then we can apply checkpointing techniques. These techniques, as developed in [10, 9, 22], allow that not all intermediate states have to be stored but only a small subset of them is kept in memory as checkpoints. In the next part of this paper we introduce an appropriate checkpointing scheme and explain how Algorithm 3.3 can efficiently be implemented using this scheme.

4 Checkpointing Scheme

Consider a multiple sweep evolution $\mathcal{E}_{3\times\ell}$ which contains three alternative sweeps. Each sweep consists of ℓ consecutive time steps. An example of such an evolution is shown in Fig. 1. Nodes denote different intermediate states. Time steps are shown as combination of arrows, that point to a specified intermediate state. Directions of these arrows denote the information flow. Primal and adjoint sweeps are characterized by one specified direction, i.e. the direction of

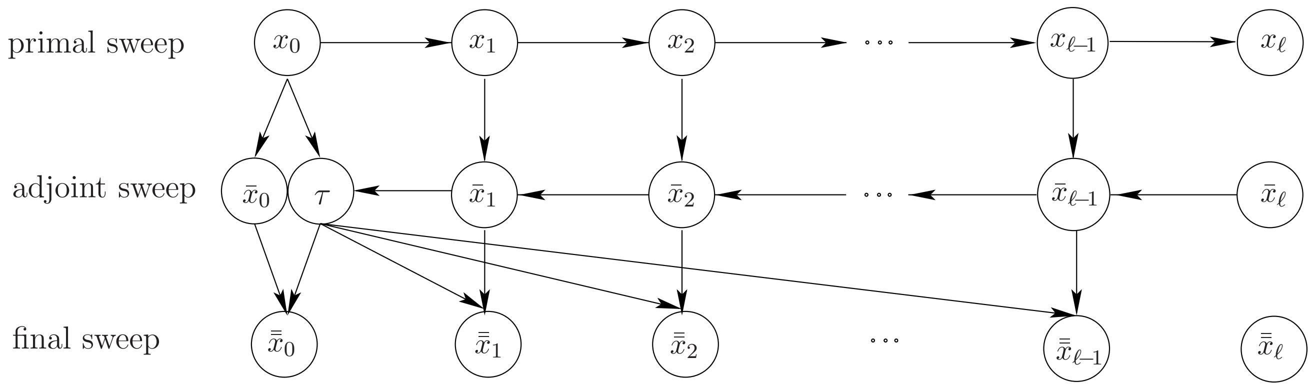

Figure 1: Multiple sweep evolution $\mathcal{E}_{3\times\ell}$

horizontal arrows within a single sweep. It shows the corresponding information flow between neighboring intermediate states in these sweeps. Note that the information flow within the primal and adjoint sweeps has a constant direction. An additional information flow exists between nodes being intermediate states of different successive sweeps. This is shown by vertical and slanted lines in Fig. 1. Final sweep does not contain any horizontal arrows, which identifies that each intermediate state of this sweep does not depend on its neighbors, but on an appropriate intermediate states of the adjoint sweep and the parameter τ. These relationships are shown also by vertical and slanted lines. To simplify further notations we add an artificial final intermediate state $\bar{\bar{x}}_\ell$, which however does not involved in any calculations.

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

$$x_i = F_i(x_{i-1}), \quad \bar{x}_{i-1} = \bar{F}_i(x_{i-1}, \bar{x}_i), \quad \bar{\bar{x}}_{i-1} = \bar{\bar{F}}_i(\bar{x}_{i-1}, \tau), \quad 1 \le i \le \ell. \tag{30}$$

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

$$t = TIME(F_i), \quad \bar{t} = TIME(\bar{F}_i), \quad \bar{\bar{t}} = TIME(\bar{\bar{F}}_i), \quad 1 \le i \le \ell. \tag{31}$$

In this paper we assume uniform evaluation costs within a corresponding sweep. To establish a relationship between the abstract multiple sweep evolution $\mathcal{E}_{3\times\ell}$ in Fig. 1 and Algorithm 3.3, we define

$$\begin{array}{rcl} x_i & = & \{\mathbf{F}_i, \xi_i^2, \Phi_i^{22}\}\,,\ i = 0, ..., \ell, \\ \bar{x}_i & = & \{\xi_i^1, \Phi_i^{11}, \Phi_i^{12}\}\,,\ i = 0, ..., \ell, \\ \bar{\bar{x}}_i & = & \{y_i\}\,,\ i = 0, ..., \ell, \\ F_i & = & \textbf{Primal Step i},\ i = 1, ..., \ell, \\ \bar{F}_i & = & \textbf{Adjoint Step i},\ i = 1, ..., \ell, \\ \bar{\bar{F}}_i & = & \textbf{Final Step i},\ i = 1, ..., \ell. \end{array} \tag{32}$$

Obviously, the relations (30) accomplished by the definitions (32) agree with the scenario presented in Algorithm 3.3.

If the straight forward implementation of Algorithm 3.3 requiring storage of all intermediate states of primal and adjoint sweeps fails due to the restricted memory capacity, we can apply the following checkpointing technique: Assume, there is memory available to simultaneously store s primal intermediate states x_i. We call them s checkpoints. Firstly, we describe the reversal of a forward simulation $F = \{F_1, ..., F_\ell\}$ using s checkpoints (see also [9, 11]), i.e. implementing primal and adjoint sweeps of the evolution $\mathcal{E}_{3\times\ell}$ in Fig. 1, such that s different primal states x_i can be simultaneously kept in memory.

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 primal or adjoint step is to evaluate, which primal state is to store or restore from a corresponding checkpoint. A certain combination of actions represents a **simple reversal schedule**. More formally, use the following definition.

Definition 4.1 *(***Simple Reversal Schedule S***). Consider a double sweep transversal $\mathcal{E}_{2\times\ell}$ consisting of a primal and an adjoint sweep, so that each of these sweeps comprises ℓ steps $F_i,\ 1 \le i \le \ell$ and $\bar{F}_i,\ 1 \le i \le \ell$, respectively. Let $s \in \mathbb{N}$ checkpoints be available each of which can accommodate data of a single primal state. ℓ denotes the number of the currently final primal step, i denotes the current state and j denotes the number of stored checkpoints. Initially, $i = i_0 \ge 0,\ j = j_0 \ge 0$ and $i < \ell \le \infty$. Then, a simple reversal schedule* **S** *consists of a finite sequence of the following actions:*

$$\begin{array}{rcl} A & = & \textit{If } i < \ell - 1, \textit{ carry out the primal step } F_{i+1} \textit{ and set } i = i + 1. \\ W_k & = & \textit{Write data of the current primal state into the checkpoint} \\ & & j_0 + k. \\ R_k & = & \textit{Read out the primal state from the checkpoint } j_0 + k \textit{ and} \\ & & \textit{set the counter } i \textit{ to its number.} \\ U & = & \textit{If } 0 \le \ell - 1 = i, \textit{ carry out the adjoint step } \bar{F}_{i+1} \textit{ and set} \\ & & \ell = \ell - 1, \textit{ else terminate.} \end{array}$$

We normally assume that all simple reversal schedules terminate successfully in that $\ell = 0$ 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 simple reversal schedule **S** begins with the action R_0, so that the initial primal state is read out from the checkpoint j_0. If an action A is consecutively carried out up to k times, it is denoted A^k. Note that some results from [9] are already incorporated in Definition 4.1. In particular, a simple reversal schedule **S** is defined in such a manner that each adjoint step $\bar{F}_i,\ 1 \le i \le \ell$, is evaluated exactly once.

Fig. 2 illustrates an example of a simple reversal schedule $\mathbf{S}$ with $\ell = 10$ and $s = 3$ for $i_0 = 0$ and $j_0 = 0$. The temporal complexity for primal and adjoint steps is given by $t = 1$ and $\bar{t} = 2$, respectively. Primal steps F_i, $1 \leq i \leq 10$, are plotted along the vertical y-axis, whereas

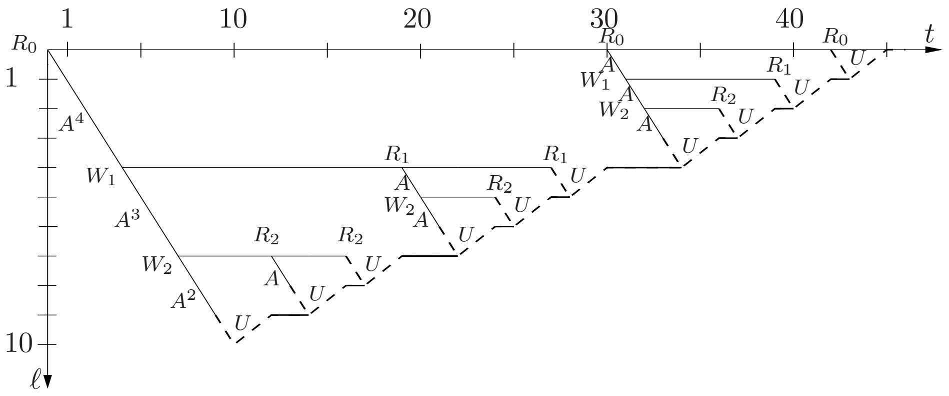

Figure 2: Simple Reversal schedule $\mathbf{S}$ with $\ell = 10$ and $s = 3$

time required for the evaluation of single actions is plotted along the horizontal x-axis. Time is measured in t-units. Horizontal lines in Fig. 2 represent checkpoints. Since the initial checkpoint contains data of the primal state x_0 and lies on the computational axis, this axis represents a checkpoint itself.

The execution of the simple reversal schedule $\mathbf{S}$ in Fig. 2 can be described as follows. Firstly, the action R_0 is performed, i.e. the initial primal state x_0 is read out from the first checkpoint $j_0 = 0$. After that, the four primal steps F_1, F_2, F_3, and F_4 are evaluated consecutively, which can be described by the action A^4. Then, the primal state x_4 is stored into the second checkpoint by the action W_1. The following three primal steps F_5, F_6, and F_7 are carried out by the action A^3. The primal state x_7 is stored into the third checkpoint by the action W_2. The following two primal 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 primal state x_7 is read out from the third checkpoint by the action R_2. Using this data, the primal 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 simple reversal schedule $\mathbf{S}$ in Fig. 2 can be executed using 45 t-units. The overall number of primal steps, evaluated throughout the execution of this schedule, is 15. This number identifies the temporal complexity of the simple reversal schedule and is denoted by $\mathbf{T}(\mathbf{S})$. Simple reversal schedules, that yield minimal number of primal steps throughout their execution are called **binomial** reversal schedules and, depending on specified parameters ℓ and s, are denoted by $\mathbf{S}_{bin}(\ell, s)$. The simple reversal schedule $\mathbf{S}$ in Fig. 2 is a binomial one.

Minimal number of primal steps which represents the temporal complexity, achieved by the **binomial** reversal schedule $\mathbf{S}_{bin}(\ell, s)$, is defined by

$$\mathbf{T}_{bin}(\ell, s) = \min_{\mathbf{S}} \left\{ \mathbf{T}(\mathbf{S}), \mathbf{S} - \text{ simple reversal schedule, admissible for } \ell \text{ and } s \right\}. \tag{33}$$

$\mathbf{T}_{bin}(\ell, s)$ satisfies the recursion

$$\mathbf{T}_{bin}(\ell, s) = \min_{0 < \check{\ell} < \ell} \{ \check{\ell} + \mathbf{T}_{bin}(\ell - \check{\ell}, s - 1) + \mathbf{T}_{bin}(\check{\ell}, s) \}, \tag{34}$$

which implies

$$\mathbf{T}_{bin}(\ell, s) = r\ell - \beta(s+1, r-1), \tag{35}$$

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

$$\beta(s, r-1) \quad < \quad \ell \quad \leq \quad \beta(s, r) = \binom{s+r}{s} = \frac{(s+r)!}{s!\, r!}. \tag{36}$$

The explicit expression (35) for $\mathbf{T}_{bin}(\ell, s)$ was proved into different ways in [9, 11, 22]. In the present paper we give a new representation and a new proof of the minimal evaluation cost $\mathbf{T}_{bin}(\ell, s)$, which affords better insights into properties of binomial reversal schedules.

Proposition 4.2 *The minimal evaluation cost $\mathbf{T}_{bin}(\ell, s)$ can be computed by*

$$\mathbf{T}_{bin}(\ell, s) = \sum_{i=1}^{\ell} r(s, i), \tag{37}$$

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

$$\beta(s, r(s,i) - 1) \quad < \quad i \quad \leq \quad \beta(s, r(s, i)). \tag{38}$$

Proof: Relation (37) can be proved by induction.

Trivial Case I ($s = 1$).

To achieve the minimal number $\mathbf{T}_{bin}(\ell, 1)$ of primal steps evaluations, we can explicitly derive the corresponding simple reversal schedule: The only checkpoint available must be set to the initial primal state x_0. For the first Reverse action U in Definition 4.1, $(l-1)$ primal steps F_j, $1 \leq j \leq \ell - 1$ have to be evaluated. To perform the next Reverse action U, $(l-2)$ primal steps F_j, $1 \leq j \leq \ell - 2$ have to be evaluated and so on. Hence, for the overall reversal we obtain

$$\mathbf{T}_{bin}(\ell, 1) = (\ell - 1) + (\ell - 2) + \cdots + 1 + 0 = \sum_{i=1}^{\ell}(i-1) = \sum_{i=1}^{\ell} r(1, i),$$

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

Trivial Case II ($1 < \ell \leq s + 1$).

For this choice of ℓ and s we obtain

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

The s checkpoints must be set to all primal states except for the last two states. For the first reverse step a minimum of $(\ell - 1)$ primal steps F_j, $1 \leq j \leq \ell - 1$ is needed. To perform other reverse steps, no primal steps are necessary. Therefore, the minimal number of primal steps is equal to

$$\mathbf{T}_{bin}(\ell, s) = \ell - 1 = \underbrace{0 + 1 + 1 + \ldots + 1}_{\ell} = \sum_{i=1}^{\ell} r(s, i),$$

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

Induction step over ℓ.

The numbers s and ℓ are given. Assume that the assertion (37) is true for all $(s, \tilde{\ell})$ with $\tilde{\ell} \leq \ell$. Now, it will be shown that the property (37) is valid for the pair $(s, \ell+1)$, i.e.

$$\mathbf{T}_{bin}(\ell+1, s) = \sum_{i=1}^{\ell+1} r(s,i). \tag{39}$$

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

Therefore, in order to prove that (39) is valid, it is enough to show that $\mathbf{T}_{bin}(\ell+1, s) - \mathbf{T}_{bin}(\ell, s) = r(s, \ell+1)$, with $\mathbf{T}_{bin}(\ell+1, s)$ and $\mathbf{T}_{bin}(\ell, s)$ evaluated using (35). Thus, applying (35), we obtain

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

Here, consider two different cases:

First case: $r(s, \ell+1) = r(s, \ell)$. Therefore,

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

which proves the assertion of Proposition 4.2 for this case.

Second case: $r(s, \ell+1) = r(s,\ell)+1$. Note that if $r(s,\ell+1) = r(s,\ell)+1$, relation (36) implies that $\ell = \beta(s, r(s,\ell))$. Therefore,

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

This proves the assertion of Proposition 4.2 for the second case. Here, we use the relation

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

This relation can be straightforwardly proved applying the definition of the function β.

$\square$

Utilizing checkpointing techniques, described above, an efficient implementation of Algorithm 3.3 proceeds as follows:

Algorithm 4.3

a. Apply a binomial reversal schedule $\mathbf{S}_{bin}(\ell, s)$ *with* s *checkpoints to reverse the primal sweep containing* ℓ *primal steps* $F_1, ..., F_\ell$ *and calculate adjoint states* $\bar{x}_0, ..., \bar{x}_{\ell-1}$.

b. Determine the parameter τ.

c. *Apply the binomial reversal schedule* $\mathbf{S}_{bin}(\ell, s)$ *one more time to evaluate adjoint states* $\bar{x}_i,\ i = 0, ..., \ell - 1$. *Once the adjoint state* $\bar{x}_i$ *is computed, apply the final step* $\bar{\bar{F}}_{i+1}$ *to evaluate final intermediate states* $\bar{\bar{x}}_i,\ i = 0, ..., \ell - 1$.

In the next section of this paper we demonstrate on some numerical example how the combination of Algorithm 3.3 and Algorithm 4.3 works.

5 Numerical Results

The following academic example of an optimal control problem is constructed over the linear boundary value problem in Example 4.11 in Ascher, Mattheij and Russel [1]: Minimize

$$\int_0^T u(t)^2 - \psi(t)x(t)^2 - 2(\exp(t) - \psi(t))x(t)dt \tag{40}$$

subject to

$$x' \;=\; \psi(t)x(t) - \psi(t)\exp(t) + u(t), \qquad x(T) = \exp(T), \tag{41}$$

and

$$-u(t) + \exp(t) \le 0,$$

where $\psi(t) = 20\sin(t) + 20t\cos(t)$, $T = 2.0$. The Hamilton function reads as

$$H(x, u, \lambda, \eta) = u(t)^2 - \psi(t)x(t)^2 - 2(\exp(t) - \psi(t))x(t) + \lambda\left(\psi(t)x(t) - \psi(t)\exp(t) + u(t)\right) + \eta(t)\left(-u(t) +\right.$$

With $z = (x, u, \lambda, \eta, \sigma)$ the function F in (9) is given by

$$F(z) = \left(\begin{array}{l} x'(t) - \psi(t)x(t) + \psi(t)\exp(t) - u(t) \\ \lambda'(t) - 2\psi(t)x(t) - 2(\exp(t) - \psi(t)) + \lambda\psi(t) \\ \sigma'(t) \\ x(T) - \exp(T) \\ \lambda(0) \\ \lambda(T) - \sigma(0) \\ 2u(t) + \lambda(t) - \eta(t) \\ \varphi\left(-\left(-u(t) + \exp(t)\right), \eta\right) \end{array}\right).$$

In each iteration of the stable nonsmooth Newton method we have to solve the linear boundary value problem for $\delta x, \delta\lambda, \delta\sigma$:

$$\begin{pmatrix} \delta x'(t) \\ \delta\lambda'(t) \\ \delta\sigma'(t) \end{pmatrix} = A(t)\begin{pmatrix} \delta x(t) \\ \delta\lambda(t) \\ \delta\sigma(t) \end{pmatrix} + q(t), \qquad \text{a.e. in } [0, T],$$

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix}\begin{pmatrix} \delta x(0) \\ \delta\lambda(0) \\ \delta\sigma(0) \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}\begin{pmatrix} \delta x(T) \\ \delta\lambda(T) \\ \delta\sigma(T) \end{pmatrix} = \begin{pmatrix} x(T) - \exp(T) \\ \lambda(0) \\ \lambda(T) - \sigma(0) \end{pmatrix},$$

with

$$A(t) = \begin{pmatrix} \psi(t) & 0 & 0 \\ 2\psi(t) & -\psi(t) & 0 \\ 0 & 0 & 0 \end{pmatrix} - \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}\mathcal{A}^{-1}\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

$$q(t) = -\begin{pmatrix} (x^k)' - f \\ (\lambda^k)' + H_x'^\top \\ 0 \end{pmatrix} - \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \mathcal{A}^{-1} \begin{pmatrix} H_u'^\top \\ \omega(z^k(\cdot)) \end{pmatrix} \qquad \text{and} \qquad \mathcal{A} = \begin{pmatrix} 2 & -1 \\ s & r \end{pmatrix}.$$

Therefore, the underlying ordinary differential equation has modes growing like $e^{20t\sin t}$ and $e^{-20t\sin t}$, i.e. possess a dichotomy property. Both, the single shooting as well as the multiple shooting method without decoupling techniques fail to converge for this boundary value problem. Table 1 shows the output of the nonsmooth Newton method based on multiple shooting without decoupling techniques. By application of the single shooting method the situation turns out to be even more dramatic.

The optimal control problem (40) - (41) is solved by the nonsmooth Newton method with the combination of multiple shooting and decoupling techniques described in Algorithm 3.3 with $\ell = 100$ equidistant subintervals. The occurring initial value problems (21) and (22) are solved by the explicit Euler's method. For this purpose each of ℓ subintervals is divided into 30 subsubintervals. Applying the stable nonsmooth Newton method from Algorithm 3.3 the analytic solution of the optimal control problem (40) - (41)

$$\begin{pmatrix} x(t) \\ \lambda(t) \\ u(t) \\ \eta(t) \end{pmatrix} = \begin{pmatrix} \exp(t) \\ 2\exp(t) - 2 \\ \exp(t) \\ 4\exp(t) - 2 \end{pmatrix}$$

is achieved after 6 iterations with the absolute error $\|dz^6\| < 10e-5$. Table 2 shows the output of the stable nonsmooth Newton method. The iterations show the rapid quadratic convergence at the end of the iteration sequence.

Fig. 3 illustrates the iterates of the stable nonsmooth Newton method. As the initial guess we choose $(x^0(t), \lambda^0(t), u^0(t), \sigma^0(t))^\top = (0,0,0,0)^\top$ a.e. $t \in [0,T]$.

To investigate how checkpointing techniques from Section 4 work we apply Algorithm 4.3 to solve the OCP (40) - (41). The advantage of the checkpointing is that not all intermediate states have to be simultaneously stored but only a subset of them. The size of the subset depends on memory available. This strategy yields that some intermediate steps have to be computed many times. This naturally increases the overall run-time requirement. To investigate its growth behavior we apply Algorithm 4.3 to perform a single stable nonsmooth Newton iteration throughout the solution of the OCP (40) - (41) by Algorithm 2.1 for different parameters ℓ and s: $\ell = 200, 500$ and 1000; the number s of available checkpoints varies from one to 50. Fig. 4 illustrates the obtained run-time results.

The number s of available checkpoints is plotted along the horizontal x - axis, while the run-time ratios are plotted along the vertical y - axis. They are computed by the formula:

$$\frac{\text{run-time of checkpointing approach with } s \text{ checkpoints}}{\text{run-time of straight forward approach: } s = \ell}.$$

From Fig. 4 we can see that the memory reduction of one order of magnitude results in the run-time increase only up to three times. More sophisticated results concerning a run-time - memory trade-off can be obtained if state equations of the optimal control problem are represented by a spatial discretization of partial differential equations. The consideration of such types of control problems is the subject of the future work.

6 Conclusion

In the present paper we described how the nonsmooth Newton method can be realized for the solution of optimal control problems subject to mixed control-state constraints in the dichotomy case, i.e. dynamic components contain both fast growing and fast decaying modes. The realization of this method could be arranged in such a way, that a memory efficient implementation using checkpointing was possible. To this end, we explained how an abstract checkpointing scheme can be utilized for the memory efficient implementation of the nonsmooth Newton method. Checkpointing techniques are especially of interest in cases if state equations are represented by discretized partial differential equations. The huge memory amount required by the straight forward implementation of the nonsmooth Newton method in this case can be reduced to an acceptable level by utilizing binomial reversal schedules. In addition we got a better insight into binomial reversal schedules by giving a new representation of minimal reversal cost. Stable nonsmooth Newton method as well as checkpointing techniques were tested on an optimal control problem with the dichotomy property. On this example it was clearly justified that the choice of stable algorithms based on a decoupling is more advantageous. Furthermore, it was illustrated that using binomial reversal schedules it is possible to significantly reduce the memory requirement, while the run-time requirement increases moderately. Applications of the stable nonsmooth Newton method to solve optimal control problems based on partial differential equations is the task to be tackled in future.

7 Acknowledgment

The second author is grateful to Professor Andreas Griewank for bringing her to study subjects proposed in the present paper. For his continued encouragement and support during the progress of research.

References

[1] U. Ascher, R. Mattheij and R. Russel, *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*, SIAM, Philadelphia, 1995.

[2] X. Chen, Z. Nashed, and L. Qi, *Smoothing methods and semismooth methods for nondifferentiable operator equations*, SIAM Journal on Numerical Analysis, 38 (2000), pp. 1200–1216.

[3] F. H. Clarke, *Optimization and Nonsmooth Analysis*, John Wiley & Sons, New York, 1983.

[4] A. Fischer, *A special Newton-type optimization method*, Optimization, 24 (1992), pp. 269–284.

[5] ——, *Solution of monotone complementarity problems with locally Lipschitzian functions*, Mathematical Programming, 76 (1997), pp. 513–532.

[6] M. Gerdts, *Direct Shooting Method for the Numerical Solution of Higher Index DAE Optimal Control Problems*, Journal of Optimization Theory and Applications, 117 (2003), pp. 267–294.

[7] ——, *A nonsmooth newton's method for control-state constrained optimal control problems*, vol. 30 of Proceedings 5th Vienna Symposium on Mathematical Modelling, ARGESIM Reports, ISBN 3-901608-30-3, Vienna, 2006.

[8] ——, *Global convergence of a nonsmooth Newton method for control-state constrained optimal control problems*, SIAM Journal on Optimization, 19 (1), pp. 326-350, 2008. A corrected version can be found on http://www.mathematik.uni-wuerzburg.de/~gerdts/erratum_SIAM_19_1_2008_326-350_full.pdf.

[9] A. Griewank, *Evaluating Derivatives. Principle and Techniques of Algorithmic Differentiation*, SIAM, Philadelphia, 2000.

[10] A. Griewank, *Achieving Logarithmic Growth of Temporal and Spatial Complexity in Reverse Automatic Differentiation*, Optimization Methods and Software, 1:35 – 54, 1992.

[11] 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.

[12] H. Jiang, *Global convergence analysis of the generalized Newton and Gauss-Newton methods of the Fischer-Burmeister equation for the complementarity problem*, Mathematics of Operations Research, 24 (1999), pp. 529–543.

[13] B. Kummer, *Newton method for non-differentiable functions*, in Advances in Mathematical Optimization, J. Guddat et al., ed., Berlin, 1988, Akademie-Verlag, pp. 171–194.

[14] ——, *Newton method based on generalized derivatives for nonsmooth functions: convergence analysis*, in Advances in Optimization, W. Oettli and D. Pallaschke, eds., Berlin, 1991, Springer, pp. 171–194.

[15] L. A. Ljusternik and W. I. Sobolew, *Elemente der Funktionalanalysis*, Verlag Harri Deutsch, Zürich-Frankfurt/Main-Thun, 1976.

[16] K. Malanowski, *On normality of Lagrange multipliers for state constrained optimal control problems*, Optimization, 52 (2003), pp. 75–91.

[17] H. Maurer and D. Augustin, *Sensitivity Analysis and Real-Time Control of Parametric Optimal Control Problems Using Boundary Value Methods*, in Online Optimization of Large Scale Systems, M. Grötschel, S. O. Krumke, and J. Rambau, eds., Springer, 2001, pp. 17–55.

[18] L. W. Neustadt, *Optimization: A Theory of Necessary Conditions*, Princeton, New Jersey, 1976.

[19] H. J. Oberle and W. Grimm, *Bndsco – A Program for the Numerical Solution of Optimal Control Problems*, Tech. Report Reihe B, Bericht 36, Hamburger Beiträge zur Angewandten Mathematik, Department of Mathematics, University of Hamburg, http://www.math.uni-hamburg.de/home/oberle/software.html, 2001.

[20] L. Qi, *Convergence analysis of some algorithms for solving nonsmooth equations*, Mathematics of Operations Research, 18 (1993), pp. 227–244.

[21] L. Qi and J. Sun, *A nonsmooth version of Newton's method*, Mathematical Programming, 58 (1993), pp. 353–367.

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

[23] M. Ulbrich, *Nonsmooth Newton-like Methods for Variational Inequalities and Constrained Optimization Problems in Function Spaces*, Habilitation, Technical University of Munich, Munich, 2002.

[24] M. Ulbrich, *Semismooth newton methods for operator equations in function spaces*, SIAM Journal on Optimization, 13 (2003), pp. 805–841.

[25] D. Werner, *Funktionalanalysis*, Springer, Berlin-Heidelberg-New York, 1995.

ITER k	$\|dz^k\|$
1	$1.685671e+14$
2	$1.871935e+14$
3	$1.220427e+15$
$\vdots$	$\vdots$

Table 1: Output of the nonsmooth Newton method based on multiple shooting without decoupling techniques with $\ell = 100$ equidistant shooting subintervals

ITER k	$\|dz^k\|$
1	$1.610586e+01$
2	$7.940036e-01$
3	$9.307721e-01$
4	$2.732964e-01$
5	$6.551486e-03$
6	$1.688786e-05$

Table 2: Output of the stable nonsmooth Newton method with $\ell = 100$ equidistant shooting subintervals

List of figures

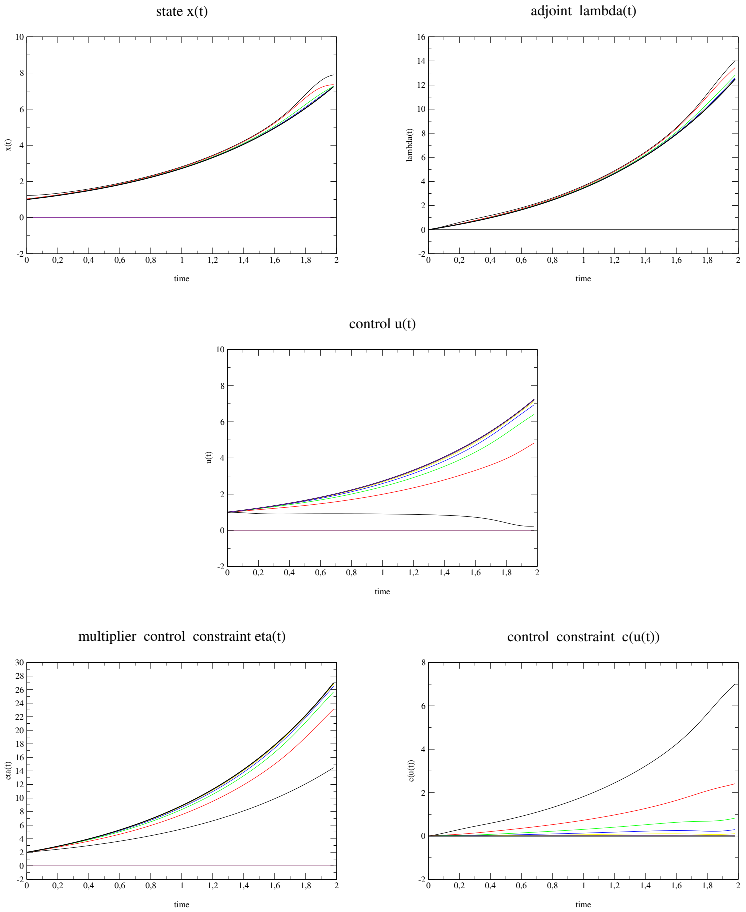

Figure 3: Numerical solution of the OCP (40) - (41) applying nonsmooth Newton method for $\ell = 100$ multiple shooting subintervals: Intermediate iterates (thin lines) and converged solution (thick lines)

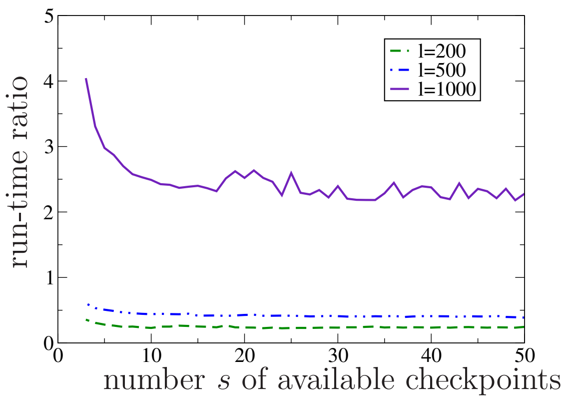

Figure 4: Run-time ratios obtained from the implementation of Algorithm 2.1 for the solution of the OCP (40) - (41) using binomial checkpointing according to Algorithm 4.3