Memory-efficient Implementation of Riccati Approach for Time-Dependent Optimal Control Problems in ODEs

Julia Sternberg[*1] and **Andreas Griewank**[**2]

[1] Technische Universität Dresden, Institut für Wissenschaftliches Rechnen, Germany
[2] Humboldt-Universität zu Berlin, Institut für Mathematik, Germany

We consider a time-dependent optimal control problem, where the state evolution is described by an ODE. There is a variety of methods for the treatment of such problems. We prefer to view them as boundary value problems and apply to them the Riccati approach for non-linear BVPs with separated boundary conditions. There are many relationships between multiple shooting techniques, the Riccati approach and the Pantoja method, which describes a computationally efficient stage-wise construction of the Newton direction for the discrete-time optimal control problem. We present an efficient implementation of this approach. Furthermore, the well-known checkpointing approach is extended to a 'nested checkpointing' for multiple transversals. Some heuristics are introduced for an efficient construction of nested reversal schedules. We discuss their benefits and compare their results to the optimal schedules computed by exhaustive search techniques.

Copyright line will be provided by the publisher

1 Time-Dependent Optimal Control Problems in ODEs

Consider the following unconstrained primal control problem

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

where the system is described by

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

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 minimizes (1). In order to characterize an optimal control function $\mathbf{u}(t)$ for the minimization problem (1) and (2) we consider the following adjoint state equation

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

where the *Hamiltonian* function H is given by $H(\mathbf{x}(t), \mathbf{u}(t), \bar{\mathbf{x}}(t), t) = \bar{\mathbf{x}}^T(t) f(\mathbf{x}(t), \mathbf{u}(t), t)$. At each point along the solution path the Hamiltonian function must be minimal with respect to the control value $\mathbf{u}(t)$. Therefore, we have the *First Order Necessary Optimality Condition*

$$\left(\frac{\partial H}{\partial \mathbf{u}}\right)^T = 0, \qquad 0 \le t \le T, \tag{4}$$

for the optimal control problem (1) - (2).

A large variety of numerical methods for solving optimal control problems has been proposed and used in various fields of applications. One fairly popular concept is to juxtapose approaches that first discretize and then optimize with those that first optimize and then discretize. Methods of the first type are sometimes called *direct* as they treat the discretized control problem immediately as a finite dimensional nonlinear program, which can be handed over to increasingly sophisticated and robust NLP codes. In the alternative *indirect* approach one firstly derives optimality conditions in a suitable function space setting and then discretizes the resulting boundary value problem with algebraic side constraints. Here we show how the memory requirement can be kept within reasonable bounds for the indirect approach nevertheless.

The BVP (2) - (3) is in general non-linear with separated (BC). We use a quasilinearization scheme to solve it iteratively. First, we linearize (2) - (3), and (4); after that we solve the resulting linear BVP using the Riccati approach. This approach leads to the *Quasilinearization scheme for solving optimal control problems using the Riccati method* (see [3]). Each iteration of the quasilinearization scheme consists of three sweeps through the time window $[0,T]$, which are referred to as *original*, *adjoint* and *final* sweep. In Fig. 1 the dimensions of the data objects flowing between these three sweeps are shown. The horizontal arrows represent informational flow between the three sweeps that are represented by slanted lines. Two cameras pointing at the original and 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.

* e-mail: jstern@math.tu-dresden.de, Phone: ++49 351 463 34082, Fax: +49 351 463 37096
** e-mail: griewank@mathematik.hu-berlin.de, Phone: +49 30 2093 5820, Fax: +49 30 2093 5859

Copyright line will be provided by the publisher

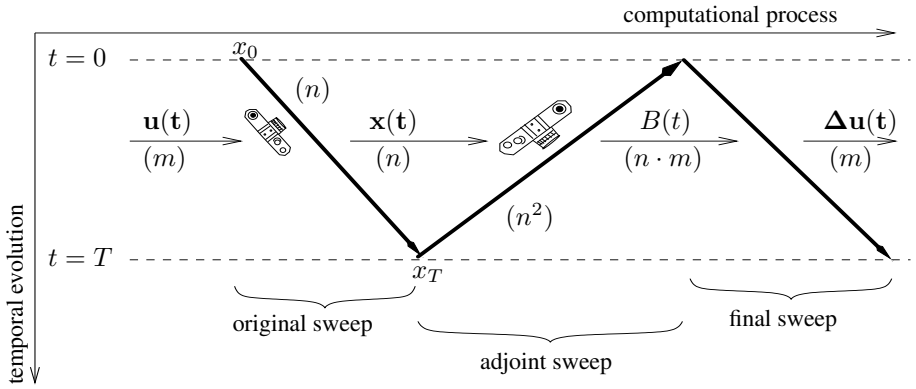

Fig. 1 Information flow for Riccati/Pantoja computation of Newton step

2 Memory-efficient Implementation of Quasilinearization Scheme

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. However, 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. 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. The schedules are designed to minimize the overall execution time given a certain total amount of storage for the checkpoints. While we have not found a closed form solution for this discrete optimization problem we have developed a cheap heuristic for constructing nested reversals that are quite close to optimality. Moreover, the dependence on l can be arranged polylogarithmically [1] 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. The proposed scheduling schemes were applied to laser surface hardening of steel [2] and other practical optimal control problems. Figure 2 illustrates resulted run-time ratios for the optimal control problem of laser surface hardening of steel with a nonuniform mesh with $n = 91$ degrees of freedom. x-axis gives the number of time-

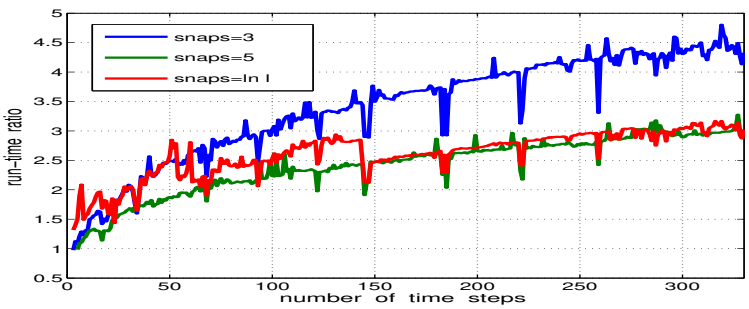

Fig. 2 Run-time & Memory Trade-off

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. In Figure 2 we can observe that run-times, resulted 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. For the details see [3].

References

[1] A. Griewank. Achieving Logarithmic Growth of Temporal and Spatial Complexity in Reverse Automatic Differentiation. Optimization Methods and Software, 1:35 – 54, 1992.
[2] 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.
[3] J. Sternberg. Reduction of storage requirement by checkpointing for time-dependent optimal control problems. Dissertation, in preparation.

Copyright line will be provided by the publisher