Christian H. Bischof · H. Martin Bücker
Paul Hovland · Uwe Naumann
Jean Utke
Editors

Advances in Automatic Differentiation

With 111 Figures and 37 Tables

Springer

Christian H. Bischof
H. Martin Bücker
Institute for Scientific Computing
RWTH Aachen University
52056 Aachen
Germany
bischof@sc.rwth-aachen.de
buecker@sc.rwth-aachen.de

Uwe Naumann
Software and Tools for Computational Engineering
RWTH Aachen University
52056 Aachen
Germany
naumann@stce.rwth-aachen.de

Paul Hovland
Jean Utke
Mathematics and Computer Science Division
Argonne National Laboratory
9700 S Cass Ave
Argonne, IL 60439
USA
hovland@mcs.anl.gov
utke@mcs.anl.gov

ISBN 978-3-540-68935-5 e-ISBN 978-3-540-68942-3

Lecture Notes in Computational Science and Engineering ISSN 1439-7358

Library of Congress Control Number: 2008928512

Mathematics Subject Classification (2000): 65Y99, 90C31, 68N19

Cover design: deblik, Berlin

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

spinger.com

Preface

The Fifth International Conference on Automatic Differentiation held from August 11 to 15, 2008 in Bonn, Germany, is the most recent one in a series that began in Breckenridge, USA, in 1991 and continued in Santa Fe, USA, in 1996, Nice, France, in 2000 and Chicago, USA, in 2004. The 31 papers included in these proceedings reflect the state of the art in automatic differentiation (AD) with respect to theory, applications, and tool development. Overall, 53 authors from institutions in 9 countries contributed, demonstrating the worldwide acceptance of AD technology in computational science.

Recently it was shown that the problem underlying AD is indeed NP-hard, formally proving the inherently challenging nature of this technology. So, most likely, no deterministic "silver bullet" polynomial algorithm can be devised that delivers optimum performance for general codes. In this context, the exploitation of domain-specific structural information is a driving issue in advancing practical AD tool and algorithm development. This trend is prominently reflected in many of the publications in this volume, not only in a better understanding of the interplay of AD and certain mathematical paradigms, but in particular in the use of hierarchical AD approaches that judiciously employ general AD techniques in application-specific algorithmic harnesses. In this context, the understanding of structures such as sparsity of derivatives, or generalizations of this concept like scarcity, plays a critical role, in particular for higher derivative computations.

On the tool side, understanding of program structure is the key to improving performance of AD tools. In this context, domain-specific languages, which by design encompass high-level information about a particular application context, play an increasingly larger role, and offer both challenges and opportunities for efficient AD augmentation. This is not to say that tool development for general purpose languages is a solved issue. Advanced code analysis still leads to improvements in AD-generated code, and the set of tools capable of both forward- and reverse mode AD for C and C++ continues to expand. General purpose AD tool development remains to be of critical importance for unlocking the great wealth of AD usage scenarios, as the user interface and code performance of such tools shape computational practitioners' view of AD technology.

Overall, the realization that simulation science is a key requirement to fundamental insight in science and industrial competitiveness continues to grow. Hence, issues such as nonlinear parameter fitting, data assimilation, or sensitivity analysis of computer programs are becoming de rigueur for computational practitioners to adapt their models to experimental data. Beyond the "vanilla" nonlinear least squares formulation one needs also to question in this context which parameters can at all be reliably identified by the data available in a particular application context, a question that again requires the computation of derivatives if one employs methods based on, for example, Fisher information matrix. Beyond that, experimental design then tries to construct experimental setups that, for a given computer model, deliver experimental data that have the highest yield with respect to model fitting or even model discrimination. It is worth noting that all these activities that are critical in reliably correlating computer model predictions with real experiments rely on the computation of first- and second-order derivatives of the underlying computer models and offer a rich set of opportunities for AD.

These activities are also examples of endeavors that encompass mathematical modeling, numerical techniques as well as applied computer science in a specific application context. Fortunately, computational science curricula that produce researchers mentally predisposed to this kind of interdisciplinary research continue to grow, and, from a computer science perspective, it is encouraging to see that, albeit slowly, simulation practitioners realize that there is more to computer science than "programming," a task that many code developers feel they really do not need any more help in, except perhaps in parallel programming.

Parallel programming is rising to the forefront of software developers' attention due to the fact that shortly multicore processors, which, in essence, provide the programming ease of shared-memory multiprocessors at commodity prices, will put 32-way parallel computing (or even more) on desk- and laptops everywhere. Going a step further, in the near future any substantial software system will, with great probability, need to be both parallel and distributed. Unfortunately, many computer science departments consider these issues solved, at least in theory, and do not require their students to develop practical algorithmic and software skills in that direction. In the meantime, the resulting lag in exploiting technical capabilities offers a great chance for AD, as the associativity of the chain rule of differential calculus underlying AD as well as the additional operations inserted in the AD-generated code provide opportunities for making use of available computational resources in a fashion that is transparent to the user. The resulting ease of use of parallel computers could be a very attractive feature for many users.

Lastly, we would like to thank the members of the program committee for their work in the paper review process, and the members of the Institute for Scientific Computing, in particular Oliver Fortmeier and Cristian Wente, for their help in organizing this event. The misfit and velocity maps of the Southern Ocean on the cover were provided by Matthew Mazloff and Patrick Heimbach from Massachusetts Institute of Technology and are a result of an ocean state estimation project using automatic differentiation. We are also indebted to Mike Giles from Oxford University, Wolfgang Marquardt from RWTH Aachen University, Arnold Neumeier from the

University of Vienna, Alex Pothen from Old Dominion University, and Eelco Visser from the Technical University in Delft for accepting our invitation to present us inspirations on AD possibilities in their fields of expertise. We also acknowledge the support of our sponsors, the Aachen Institute for Advanced Study in Computational Engineering Science (AICES), the Bonn-Aachen International Center for Information Technology (B-IT), and the Society for Industrial and Applied Mathematics (SIAM).

Aachen and Chicago,
April 2008

Christian Bischof
H. Martin Bücker
Paul Hovland
Uwe Naumann
Jean Utke

Program Committee AD 2008

Contents

List of Contributors

Hany S. Abdel-Khalik
North Carolina State University
Department of Nuclear Engineering
Raleigh, NC 27695–7909
USA
abdelkhalik@ncsu.edu

Roscoe A. Bartlett
Sandia National Laboratories
PO Box 5800
Albuquerque, NM 87185
USA
rabartl@sandia.gov

Houtan Bastani
Board of Governors of the Federal Reserve System Washington, DC
20551 USA
houtan.bastani@frb.gov

Bradley Bell
University of Washington
Applied Physics Laboratory
1013 NE 40th Street
Seattle, Washington 98105–6698
USA
bradbell@washington.edu

Sanjukta Bhowmick
The Pennsylvania State University
343 H IST Building
University Park PA 16802
USA
bhowmick@cse.psu.edu

Christian Bischof
RWTH Aachen University
Institute for Scientific Computing
Seffenter Weg 23
D–52074 Aachen
Germany
bischof@sc.rwth-aachen.de

H. Martin Bücker
RWTH Aachen University
Institute for Scientific Computing
Seffenter Weg 23
D–52074 Aachen
Germany
buecker@sc.rwth-aachen.de

James V. Burke
University of Washington
Department of Mathematics
Box 354350
Seattle, Washington 98195–4350
USA
burke@math.washington.edu

Isabelle Charpentier
Centre National de la Recherche Scientifique
Laboratoire de Physique et Mécanique des Matériaux
UMR CNRS 7554
Ile du Saulcy

57045 Metz Cedex 1
France
isabelle.charpentier@univ-metz.fr

Salvatore Cuomo
University of Naples Federico II
Via Cintia
Naples
Italy
salvatore.cuomo@unina.it

Luisa D'Amore
University of Naples Federico II
Via Cintia
Naples
Italy
luisa.damore@dma.unina.it

Claude Dal Cappello
Université de Metz
Laboratoire de Physique Moléculaire et des Collisions
1 Bd Arago
57078 Metz Cedex 3
France
cappello@univ-metz.fr

Atya Elsheikh
Siegen University
Department of Simulation
D–57068 Siegen
Germany
elsheikh@simtec.mb.uni-siegen.de

Shaun A. Forth
Cranfield University
Applied Mathematics & Scientific Computing Group
Engineering Systems Department
Defence College of Management and Technology
Shrivenham, Swindon SN6 8LA
UK
S.A.Forth@cranfield.ac.uk

David M. Gay
Sandia National Laboratories
PO Box 5800
Albuquerque, NM 87185
USA
dmgay@sandia.gov

Assefaw Gebremedhin
Old Dominion University
Computer Science Department and
Center for Computational Sciences
4700 Elkhorn Ave., Suite 3300
Norfolk, VA 23529
USA
assefaw@cs.odu.edu

Ralf Giering
FastOpt
Schanzenstrasse 36
D–20357 Hamburg
Germany
ralf.giering@fastopt.com

Mike Giles
Oxford University
Mathematical Institute
24–29 St Giles
Oxford OX1 3LB
UK
mike.giles@maths.ox.ac.uk

Markus Grabner
Graz University of Technology
Inffeldgasse 16a/II
8010 Graz
Austria
grabner@icg.tugraz.at

Andreas Griewank
Humboldt-Universität zu Berlin
Institute of Mathematics
Unter den Linden 6
D–10099 Berlin
Germany
griewank@math.hu-berlin.de

Tobias Gross
Graz University of Technology
Inffeldgasse 16a/II
8010 Graz
Austria
tobias.gross@student.tugraz.at

Marin D. Guenov
Cranfield University
Engineering Design Group
Aerospace Engineering Department
School of Engineering
Bedfordshire MK43 0AL
UK
M.D.Guenov@cranfield.ac.uk

Luca Guerrieri
Board of Governors of the Federal Reserve System
Washington, DC 20551
USA
luca.guerrieri@frb.gov

Niels Guertler
RWTH Aachen University
Institute for Scientific Computing
Seffenter Weg 23
D–52074 Aachen
Germany
Niels.Guertler@rwth-aachen.de

Laurent Hascoët
INRIA
Sophia-Antipolis
2004 route des lucioles – BP 93
FR–06902 Sophia Antipolis Cedex
France
Laurent.Hascoet@sophia.inria.fr

Robert J. Hoekstra
Sandia National Laboratories
PO Box 5800
Albuquerque, NM 87185
USA
rjhoeks@sandia.gov

Paul D. Hovland
Argonne National Laboratory
Mathematics and Computer Science Division
9700 S. Cass Ave.
Argonne, IL 60439
USA
hovland@mcs.anl.gov

Armen Jaworski
Warsaw University of Technology
Institute of Aeronautics and Applied Mechanics
Plac Politechniki 1
00661 Warsaw
Poland
armen@meil.pw.edu.pl

Bernhard Kainz
Graz University of Technology
Inffeldgasse 16a/II
8010 Graz
Austria
kainz@icg.tugraz.at

Thomas Kaminski
FastOpt
Schanzenstrasse 36
D–20357 Hamburg
Germany
thomas.kaminski@fastopt.com

Andreas Kowarz
Technische Universität Dresden
Institute of Scientific Computing
Mommsenstr. 13
D–01062 Dresden
Germany
Andreas.Kowarz@tu-dresden.de

Koichi Kubota
Chuo University
Kasuga 1–13–27
Bunkyo-ku,
Tokyo 112–0003
Japan
kubota@ise.chuo-u.ac.jp

René Lamour
Humboldt-Universität zu Berlin
Unter den Linden 6
D–10099 Berlin
Germany
lamour@math.hu-berlin.de

Arnaud Lejeune
Université de Metz
Laboratoire de Physique et Mécanique des Matériaux
UMR CNRS 7554, Ile du Saulcy
57045 Metz Cedex 1
France
arnaud.lejeune@univ-metz.fr

Andrew Lyons
University of Chicago
Computation Institute
5640 S. Ellis Avenue
Chicago, IL 60637
USA
lyonsam@gmail.com

Priyadarshini Malusare
Argonne National Laboratory
Mathematics and Computer Science Division
9700 S. Cass Ave.
Argonne, IL 60439
USA
malusare@mcs.anl.gov

Massimiliano Martinelli
INRIA
Sophia-Antipolis
2004 route des lucioles – BP 93
FR–06902 Sophia Antipolis Cedex
France
Massimiliano.Martinelli@sophia.inria.fr

Dagmar Monett
Humboldt-Universität zu Berlin
DFG Research Center MATHEON
Unter den Linden 6
D–10099 Berlin
Germany
monett@math.hu-berlin.de

Jens-Dominik Müller
Queen Mary, University of London
School of Engineering and Materials Science
Mile End Road
London, E1 4NS
UK
j.mueller@qmul.ac.uk

Almerico Murli
University of Naples Federico II
Via Cintia
Naples
Italy
almerico.murli@dma.unina.it

Uwe Naumann
RWTH Aachen University
LuFG Informatik 12: Software and Tools for Computational Engineering
RWTH Aachen University
D–52056 Aachen
Germany
naumann@stce.rwth-aachen.de

Mattia Padulo
Cranfield University
Engineering Design Group,
Aerospace Engineering Department

School of Engineering
Bedfordshire MK43 0AL
UK
M.Padulo@cranfield.ac.uk

Valerie Pascual
INRIA
Sophia-Antipolis
2004 route des lucioles – BP 93
FR–06902 Sophia Antipolis Cedex
France
Valerie.Pascual@sophia.inria.fr

Barak Pearlmutter
Hamilton Institute
NUI Maynooth
Co. Kildare
Ireland
barak@cs.nuim.ie

Monika Petera
RWTH Aachen University
Institute for Scientific Computing
Seffenter Weg 23
D–52074 Aachen
Germany
petera@sc.rwth-aachen.de

Eric Phipps
Sandia National Laboratories
PO Box 5800
Albuquerque, NM 87185
USA
etphipp@sandia.gov

Thomas Pock
Graz University of Technology
Inffeldgasse 16a/II
8010 Graz
Austria
pock@icg.tugraz.at

Alex Pothen
Old Dominion University
Computer Science Department and
Center for Computational Sciences
4700 Elkhorn Ave., Suite 3300
Norfolk, VA 23529
USA
pothen@cs.odu.edu

Michel Potier-Ferry
Université de Metz
Laboratoire de Physique et Mécanique des Matériaux
UMR CNRS 7554, Ile du Saulcy
57045 Metz Cedex 1
France
michel.potierferry@univ-metz.fr

Jan Riehme
University of Hertfordshire
Department of Computer Science
College Lane
Hatfield, Herts AL10 9AB
UK
riehme@stce.rwth-aachen.de

Mariarosaria Rizzardi
University of Naples Parthenope
Centro Direzionale is. C4
Naples
Italy
mariarosaria.rizzardi@uniparthenope.it

Adrian Sandu
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061
USA
sandu@cs.vt.edu

Jaewook Shin
Argonne National Laboratory
Mathematics and Computer Science Division

9700 S. Cass Ave.
Argonne, IL 60439
USA
jaewook@mcs.anl.gov

Jeffrey Mark Siskind
School of Electrical and Computer Engineering
Purdue University
465 Northwestern Avenue
West Lafayette, IN 47907–2035
USA
qobi@purdue.edu

Jörg Stiller
Technische Universität Dresden
Institute for Aerospace Engineering
D–01062 Dresden
Germany
joerg.stiller@tu-dresden.d

Tracy E. Stover
North Carolina State University
Department of Nuclear Engineering
Raleigh, NC 27695–7909
USA
testover@ncsu.edu

Philipp Stumm
Technische Universität Dresden
Fachrichtung Mathematik
Institut für Wissenschaftliches Rechnen
Zellescher Weg 12–14
D–01062 Dresden
Germany
Philipp.Stumm@tu-dresden.de

Emmanuel Tadjouddine
Aberdeen University
King's College
Department of Computing Science
Aberdeen AB24 3UE, Scotland
UK
etadjoud@csd.abdn.ac.uk

Jean Utke
Argonne National Laboratory
Mathematics and Computer Science Division,
9700 S. Cass Ave.
Argonne, IL 60439
USA
utke@mcs.anl.gov

Andre Vehreschild
RWTH Aachen University
Institute for Scientific Computing
Seffenter Weg 23
D–52074 Aachen
Germany
vehreschild@sc.rwth-aachen.de

Michael Voßbeck
FastOpt
Schanzenstrasse 36
D–20357 Hamburg
Germany
michael.vossbeck@fastopt.com

Andrea Walther
Technische Universität Dresden
Fachrichtung Mathematik
Institut für Wissenschaftliches Rechnen
Zellescher Weg 12–14
D–01062 Dresden
Germany
Andrea.Walther@tu-dresden.de

Wolfgang Wiechert
Siegen University
Department of Simulation
D–57068 Siegen
Germany
wiechert@simtec.mb.uni-siegen.de

Fabrice Zaoui
RTE
9 rue de la porte de Buc
78000 Versailles
France
fabrice.zaoui@RTE-FRANCE.com

Reverse Automatic Differentiation of Linear Multistep Methods

Adrian Sandu

Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061, USA, `sandu@cs.vt.edu`

Summary. Discrete adjoints are very popular in optimization and control since they can be constructed automatically by reverse mode automatic differentiation. In this paper we analyze the consistency and stability properties of discrete adjoints of linear multistep methods. The analysis reveals that the discrete linear multistep adjoints are, in general, inconsistent approximations of the adjoint ODE solution along the trajectory. The discrete adjoints at the initial time converge to the adjoint ODE solution with the same order as the original linear multistep method. Discrete adjoints inherit the zero-stability properties of the forward method. Numerical results confirm the theoretical findings.

Keywords: Reverse automatic differentiation, linear multistep methods, consistency, stability.

1 Introduction

Consider an ordinary differential equation (ODE)

$$y' = f(y)\,, \quad y(t_{\rm ini}) = y_{\rm ini}\,, \quad t_{\rm ini} \le t \le t_{\rm end}\,, \quad y \in \Re^d\,. \tag{1}$$

We are interested to find the initial conditions for which the following cost function is minimized

$$\min_{y_{\rm ini}} \overline{\Psi}(y_{\rm ini}) \quad \text{subject to (1)}\,; \qquad \overline{\Psi}(y_{\rm ini}) = g\big(y(t_{\rm end})\big)\,. \tag{2}$$

The general optimization problem (2) arises in many important applications including control, shape optimization, parameter identification, data assimilation, etc. This cost function depends on the initial conditions of (1). We note that general problems where the solution depends on a set of arbitrary parameters can be transformed into problems where the parameters are the initial values. Similarly, cost functions that involve an integral of the solution along the entire trajectory can be transformed into cost functions that depend on the final state only via the introduction of quadrature variables. Consequently the formulation (1)–(2) implies no loss of generality.

To solve (1)–(2) via a gradient based optimization procedure one needs to compute the derivatives of the cost function $\overline{\Psi}$ with respect to the initial conditions. This can be done effectively using continuous or discrete adjoint approaches.

In the *continuous adjoint* ("differentiate-then-discretize") approach [8] one derives the adjoint ODE associated with (1)

$$\overline{\lambda}' = -J^T\Big(t, y(t)\Big)\,\overline{\lambda}\,, \quad \overline{\lambda}\,(t_{\rm end}) = \left(\frac{\partial g}{\partial y}\big(y(t_{\rm end})\big)\right)^T\,, \quad t_{\rm end} \ge t \ge t_{\rm ini}\,, \tag{3}$$

Here $J = \partial f / \partial y$ is the Jacobian of the ODE function. The system (3) is solved backwards in time from $t_{\rm end}$ to $t_{\rm ini}$ to obtain the gradients of the cost function with respect to the state [8]. Note that the continuous adjoint equation (3) depends on the forward solution $y(t)$ via the argument of the Jacobian. For a computer implementation the continuous adjoint ODE (3) is discretized and numerical solutions $\overline{\lambda}_n \approx \overline{\lambda}(t_n)$ are obtained at time moments $t_{\rm end} = t_N > t_{N-1} > \cdots > t_1 > t_0 = t_{\rm ini}$.

In the *discrete adjoint* ("discretize-then-differentiate") approach [8] one starts with a numerical discretization of (1) which gives solutions $y_n \approx y(t_n)$ at $t_{\rm ini} = t_0 < \cdots < t_N = t_{\rm end}$

$$y_0 = y_{\rm ini}\,, \quad y_n = \mathscr{M}_n\,(y_0, \cdots, y_{n-1})\,, \quad n = 1, \cdots, N\,. \tag{4}$$

The numerical solution at the final time is $y_N \approx y\,(t_{\rm end})$. The optimization problem (2) is reformulated in terms of the numerical solution minimized,

$$\min_{y_{\rm ini}} \Psi\,(y_{\rm ini}) = g\big(y_N\big) \quad \text{subject to (4)}\,. \tag{5}$$

The gradient of (5) is computed directly from (4) using the transposed chain rule. This calculation and produces the discrete adjoint variables λ_N, λ_{N-1}, $\cdots$, λ_0

$$\lambda_N = \left(\frac{\partial g}{\partial y}\big(y_N\big)\right)^T\,, \quad \lambda_n = 0\,, \quad n = N-1, \cdots, 0\,, \tag{6}$$

$$\lambda_\ell = \lambda_\ell + \left(\frac{\partial \mathscr{M}_n}{\partial y_\ell}\big(y_0, \cdots, y_{n-1}\big)\right)^T \lambda_n\,, \quad \ell = n-1, \cdots, 0\,, \quad n = N, \cdots, 0\,.$$

Note that the discrete adjoint equation (6) depends on the forward numerical solution $y_0, \cdots, y_N$ via the arguments of the discrete model. The discrete adjoint process gives the sensitivities of the numerical cost function (5) with respect to changes in the forward numerical solution (4).

Consistency properties of discrete Runge-Kutta adjoints have been studied by Hager [3], Walther [9], Giles [2], and Sandu [6]. Baguer et al. [1] have constructed discrete adjoints for linear multistep methods in the context of control problems. Their work does not discuss the consistency of these adjoints with the adjoint ODE solution.

In this paper we study the consistency of discrete adjoints of linear multistep methods (LMM) with the adjoint ODE. The analysis is carried out under the following conditions. The cost function depends (only) on the final solution values, and the (only) control variables are the initial conditions. The system of ODEs and its solution are continuously differentiable sufficiently many times to make the discussion of order of consistency meaningful. The analysis assumes small time steps, such that the error estimates hold for non-stiff systems. The sequence of (possibly variable) step sizes in the forward integration is predefined, or equivalently, the step control mechanism is not differentiated (special directives may have to be inserted in the code before automatic differentiation is applied).

2 Linear Multistep Methods

Consider the linear multistep method

$$y_0 = y_{\text{ini}} \,, \tag{7a}$$

$$y_n = \theta_n\left(y_0, \cdots, y_{n-1}\right) \,, \quad n = 1, \cdots, k-1 \,, \tag{7b}$$

$$\sum_{i=0}^{k} \alpha_i^{[n]}\, y_{n-i} = h_n \sum_{i=0}^{k} \beta_i^{[n]}\, f_{n-i} \,, \quad n = k, \cdots, N \,. \tag{7c}$$

The upper indices indicate the dependency of the method coefficients on the step number; this formulation accommodates variable step sizes. The numerical solution is computed at the discrete moments $t_{\text{ini}} = t_0 < t_1 < \cdots < t_N = t_{\text{end}}$. As usual y_n represents the numerical approximation at time t_n. The right hand side function evaluated at t_n using the numerical solution y_n is denoted $f_n = f(t_n, y_n)$, while its Jacobian is denoted by $J_n = J(t_n, y_n) = (\partial f/\partial y)\,(t_n, y_n)$.

The discretization time steps and their ratios are

$$h_n = t_n - t_{n-1} \,, \quad n = 1, \cdots, N \,; \qquad \omega_n = \frac{h_n}{h_{n-1}} \,, \quad n = 2, \cdots, N \,. \tag{8}$$

We denote the sequence of discretization step sizes and the maximum step size by

$$h = \left(h_1, \cdots, h_N\right) \quad \text{and} \quad |h| = \max_{1 \le n \le N} h_n \,. \tag{9}$$

The number of steps depends on the step discretization sequence, $N = N(h)$.

Equation (7a)–(7c) is a k-step method. The method coefficients $\alpha_i^{[n]}$, $\beta_i^{[n]}$ depend on the sequence of (possibly variable) steps, specifically, they depend on the ratios $\omega_{n-k+2}, \cdots, \omega_n$.

A starting procedure θ is used to produce approximations of the solution $y_i = \theta_i\left(y_0, \cdots, y_{i-1}\right)$ at times t_i, $i = 1, \cdots, k-1$. We will consider the starting procedures to be linear numerical methods. This setting covers both the case of self-starting LMM methods (a linear i-step method gives y_i for $i = 1, \cdots, k-1$) as well as the case where a Runge Kutta method is used for initialization ($y_i = \theta_i\left(y_{i-1}\right)$ for $i = 1, \cdots, k-1$).

We next discuss the discrete adjoint method associated with (7a)–(7c). The following result was obtained in [7].

Theorem 1 (The discrete LMM adjoint process).
The discrete adjoint method associated with the linear multistep method (7a)–(7c) and the cost function

$$\Psi\left(y_{\text{ini}}\right) = g\left(y_N\right)$$

reads:

$$\alpha_0^{[N]}\lambda_N = h_N\,\beta_0^{[N]}\,J_N^T\cdot\lambda_N + \left(\frac{\partial g}{\partial y}\left(y_N\right)\right)^T\,, \tag{10a}$$

$$\sum_{i=0}^{N-m}\alpha_i^{[m+i]}\,\lambda_{m+i} = J_m^T\cdot\sum_{i=0}^{N-m}h_{m+i}\,\beta_i^{[m+i]}\,\lambda_{m+i}\,,\quad m=N-1,\cdots,N-k+1, \tag{10b}$$

$$\sum_{i=0}^{k}\alpha_i^{[m+i]}\,\lambda_{m+i} = h_{m+1}\,J^T\left(y_m\right)\cdot\sum_{i=0}^{k}\widehat{\beta}_i^{[m+i]}\,\lambda_{m+i}\,,\ \ m=N-k,\cdots,k, \tag{10c}$$

$$\lambda_{k-1}+\sum_{i=1}^{k}\alpha_i^{[k-1+i]}\,\lambda_{k-1+i} = J_{k-1}^T\cdot\sum_{i=1}^{k}\left(h_{k-1+i}\,\beta_i^{[k-1+i]}\,\lambda_{k-1+i}\right), \tag{10d}$$

$$\begin{aligned}\lambda_m+\sum_{i=k-m}^{k}\alpha_i^{[m+i]}\,\lambda_{m+i} = {} & \sum_{i=m+1}^{k-1}\left(\frac{\partial\theta_i}{\partial y_m}\right)^T\lambda_i \\ & + J_m^T\cdot\sum_{i=k-m}^{k}h_{m+i}\,\beta_i^{[m+i]}\,\lambda_{m+i}\,,\quad m=k-2,\cdots,0\,.\end{aligned} \tag{10e}$$

where

$$\widehat{\beta}_0^{[m]} = \omega_{m+1}^{-1}\,\beta_0^{[m]}\,,\quad \widehat{\beta}_1^{[m+1]} = \beta_1^{[m+1]}\,,\quad \widehat{\beta}_i^{[m+i]} = \left(\prod_{\ell=2}^{i}\omega_{m+\ell}\right)\beta_i^{[m+i]}\,,\ \ i=2,\cdots,k\,.$$

The gradient of the cost function with respect to the initial conditions is

$$\nabla_{y_{\text{ini}}}\Psi = \left(\frac{\partial\Psi}{\partial y_{\text{ini}}}\right)^T = \lambda_0\,. \tag{11}$$

Proof. The proof is based on a tedious, but straightforward variational calculus approach.

□

The original LMM method (7a)–(7c) applied to solve the adjoint ODE has coefficients $\overline{\alpha}_i^{[n]}$, $\overline{\beta}_i^{[n]}$ which depend on the sequence of steps h_n in reverse order due to the backward in time integration. These coefficients depend on the ratios $\omega_{n+k}^{-1},\cdots$, ω_{n+2k-2}^{-1}. They are in general different than the forward method coefficients $\alpha_i^{[n]}$, $\beta_i^{[n]}$

which depend on the ratios $\omega_n, \cdots, \omega_{n-k+2}$. The one-leg counterpart [5, Section V.6] of the LMM method applied to solve the adjoint ODE reads

$$\overline{\lambda}_N = \left(\frac{\partial g}{\partial y}\big(y(t_N)\big)\right)^T , \tag{12a}$$

$$\overline{\lambda}_m = \theta_m\left(\overline{\lambda}_N, \cdots, \overline{\lambda}_{m+1}\right) , \quad m = N-1, \cdots, N-k+1 , \tag{12b}$$

$$\sum_{i=0}^{k} \overline{\alpha}_i^{[m]} \overline{\lambda}_{m+i} = h_{m+1} J^T\big(y(\tau^{[m]})\big) \cdot \sum_{i=0}^{k} \overline{\beta}_i^{[m]} \overline{\lambda}_{m+i} , \tag{12c}$$

$$\tau^{[m]} = \sum_{\ell=0}^{k} \frac{\overline{\beta}_\ell^{[m]}}{\overline{\beta}^{[m]}} t_{m+\ell} , \quad \overline{\beta}^{[m]} = \sum_{\ell=0}^{k} \overline{\beta}_\ell^{[m]} , \quad m = N-k, \cdots, 0 .$$

Note that, due to linearity of the right hand side, the scaling by the $\overline{\beta}^{[m]}$ does not appear in the sum of $\overline{\lambda}$'s multiplied by J^T. The order of accuracy of the discretization (12c) is $\min(p, r+1)$, where r is the interpolation order of the method [5, Section V.6].

The discrete adjoint step (10c) looks like the one-leg method (12c). The argument at which the Jacobian is evaluated is, however, different. The initialization of the discrete adjoint (10a)–(10b) and of the one-leg continuous adjoint (12a)–(12b) are also different. Moreover the termination relations for the discrete adjoint calculation (10d), (10e) are different and depend on the initialization procedure of the forward method. We will analyze the impact of these differences on the accuracy of the the discrete adjoint as a numerical method to solve the adjoint ODE.

2.1 Consistency Analysis for Fixed Step Sizes

Consider first the case where the multistep method is applied with a fixed step size. With some abuse of notation relative to (9) in this section we consider $h_n = h$ for all n. The LMM coefficients are the same for all steps and the discrete adjoint step (10c).

Theorem 2 (Fixed stepsize consistency at interior trajectory points).
In the general case equation (10c) with fixed steps is a first order consistent method for the adjoint ODE. The order of consistency equals that of the one-leg counterpart for LMMs with

$$\sum_{\ell=1}^{k} \ell \beta_\ell = 0. \tag{13}$$

Proof. The consistency analysis can be done by direct differentiation. We take an approach that highlights the relation between (10c) and the one-leg continuous adjoint step (12c). For the smooth forward solution it holds that

$$\tau^{[m]} = t_m + h \sum_{\ell=0}^{k} \frac{\ell \beta_\ell}{\beta} , \quad \beta = \sum_{\ell=0}^{k} \beta_\ell \neq 0, \quad y(\tau^{[m]}) - y(t_m) = \mathscr{O}\left(h \sum_{\ell=0}^{k} \frac{\ell \beta_\ell}{\beta}\right) .$$

The step (10c) can be regarded as a perturbation of the one-leg step (12c)

$$\sum_{i=0}^{k} \alpha_i \lambda_{m+i} = h J^T \left(y(\tau^{[m]})\right) \sum_{i=0}^{k} \beta_i \lambda_{m+i} + \varepsilon_m \,.$$

The perturbation comes from the change in the Jacobian argument. Under the smoothness assumptions all derivatives are bounded and we have that the size of the perturbation is given by the size of the argument difference:

$$\varepsilon_m = \left(\sum_{\ell=0}^{k} \ell \beta_\ell \right) \cdot \mathscr{O}\left(h^2\right) + \mathscr{O}\left(h^{\min(p+1,r+1)}\right)$$

The order of consistency of the discrete adjoint step (10c) is therefore equal to one in the general case, and is equal to the order of consistency of the associated one-leg method when (13) holds. For Adams methods the order of consistency of the discrete adjoint is one. For BDF methods $\beta_0 \neq 0$ and $\beta_\ell = 0$, $\ell \geq 1$, therefore the order of consistency equals that of the one-leg counterpart, i.e., equals that of the original method.

We are next concerned with the effects of the initialization steps (10a), (10b) and of the termination steps (10d) and (10e).

Theorem 3 (Accuracy of the adjoint initialization steps).
For a general LMM the discrete adjoint initialization steps (10a), (10b) do not provide consistent approximations of the adjoint ODE solution. For Adams methods the initialization steps are $\mathscr{O}(h)$ approximations of the continuous adjoint solution.

Proof. By Taylor series expansion.

Theorem 4 (Accuracy of the adjoint termination steps).
For a general LMM the discrete adjoint termination steps (10d) and (10e) are not consistent discretizations of the adjoint ODE.

Proof. By Taylor series expansion.

Note that one can change the discrete adjoint initialization and the termination steps to consistent relations, as discussed in [7]. In this case we expect the method to be at least first order consistent with the adjoint ODE.

2.2 Consistency Analysis for Variable Step Sizes

For variable steps the consistency of the discrete adjoint with the adjoint ODE is not automatic. In this section we will use the notation (8).

Theorem 5 (Variable stepsize consistency at the intermediate trajectory points).
In general the numerical process (10a)–(10e) is not a consistent discretization of the adjoint ODE (3).

Proof. The relation (10c) can be regarded as a perturbation of a one-leg discretization method (10c) applied to the adjoint ODE. Replacing $J^T\left(y_m\right)$ by $J^T\left(y(t_m)\right)$ in (10c) introduces an $\mathscr{O}(h^{p+1})$ approximation error

$$\sum_{i=0}^{k} \alpha_i^{[m+i]} \lambda_{m+i} = h_{m+1} J^T\left(y(t_m)\right) \cdot \sum_{i=0}^{k} \widehat{\beta}_i^{[m+i]} \lambda_{m+i} + \mathscr{O}\left(h^{p+1}\right) , \quad m = N-k, \cdots, k .$$

The following consistency analysis of (10c) will be performed on this modified equation and its results are valid within $\mathscr{O}(h^{p+1})$.

A Taylor series expansion around t_m leads to the zeroth order consistency condition

$$\sum_{i=0}^{k} \alpha_i^{[m+i]} = 0 . \qquad (14)$$

For a general sequence of step sizes h_m the values of $\alpha_i^{[m+i]}$ at different steps m are not necessarily constrained by (14). A general discrete adjoint LMM process is therefore inconsistent with the adjoint ODE.

In the case where the forward steps are chosen automatically to maintain the local error estimate under a given threshold the step changes are smooth [4, Section III.5] in the sense that

$$|\omega_n - 1| \le \text{const} \cdot h_{n-1} \quad \Rightarrow \quad \omega_n = 1 + \mathscr{O}(|h|) . \qquad (15)$$

Recall that we do not consider the derivatives of the step sizes with respect to system state. Nevertheless, let us look at the impact of these smooth step changes on the discrete adjoint consistency. If the LMM coefficients $\alpha_i^{[n]}$ and $\beta_i^{[n]}$ depend smoothly on step size ratios ω_n, then for each n they are small perturbations of the constant step size values: $\alpha_i^{[n]} = \alpha_i + \mathscr{O}(|h|)$ and $\beta_i^{[n]} = \beta_i + \mathscr{O}(|h|)$. It then holds that $\sum_{i=0}^{k} \alpha_i^{[m+i]} = \mathscr{O}(|h|)$. Consequently the zeroth order consistency condition (14) is satisfied. The $\mathscr{O}(|h|)$ perturbation, however, prevents first order consistency of the discrete adjoint method.

For Adams methods in particular the relation (14) is automatically satisfied. The first order consistency condition for Adams methods reads $\sum_{i=0}^{k} \widehat{\beta}_i^{[m+i]} = 1$. For a general sequence of step sizes h_m the values of $\beta_i^{[m+i]}$ at different steps m are not constrained by any relation among them and this condition is not satisfied. If the forward steps are chosen such that (15) holds [4, Section III.5], and if the LMM coefficients depend smoothly on step size ratios, we have that $\sum_{i=0}^{k} \widehat{\beta}_i^{[m+i]} = 1 + \mathscr{O}(|h|)$. In this situation the discrete Adams adjoint methods are first order consistent with the adjoint ODE.

□

3 Zero-Stability of the Discrete Adjoints

The method (7a)–(7c) is zero-stable if it has only bounded solutions when applied to the test problem

$$y' = 0\,, \quad y(t_{\rm ini}) = y_{\rm ini}\,, \quad t_{\rm ini} \le t \le t_{\rm end}\,. \tag{16}$$

To be specific consider the LMM (7a)–(7c) scaled such that $\alpha_0^{[n]} = 1$ for all n. Using the notation $\mathbb{e}_1 = [1,0,\cdots,0]^T$, $\mathbb{1} = [1,1,\cdots,1]^T$, and

$$Y_n = \begin{bmatrix} y_n \\ \vdots \\ y_{n-k+1} \end{bmatrix}\,, \quad A_n = \begin{bmatrix} -\alpha_1^{[n]} I & \cdots & -\alpha_{k-1}^{[n]} I & -\alpha_k^{[n]} I \\ I & & 0 & 0 \\ \vdots & \ddots & & \vdots \\ 0 & \cdots & I & 0 \end{bmatrix}\,.$$

The LMM (7a)–(7c) is zero-stable if [4, Definition 5.4]

$$\|A_{n+\ell} A_{n+\ell-1} \cdots A_{n+1} A_n\| \le \text{const} \quad \forall\, n, \ell > 0\,. \tag{17}$$

A consequence of zero-stability (17) is that small changes $\delta y_{\rm ini}$ in the initial conditions of the test problem lead to small changes $\delta\Psi$ in the cost function.

The discrete adjoint of the numerical process (7a)–(7c) applied to (16) is zero stable if

$$\left\|A_n^T A_{n+1}^T \cdots A_{n+\ell-1}^T A_{n+\ell}^T\right\| \le \text{const} \quad \forall\, n, \ell > 0\,, \tag{18}$$

which ensures that all its numerical solutions remain bounded. The product of matrices in (18) is the transpose of the product of matrices in (17), and consequently if (17) holds then (18) holds. In other words if a variable-step LMM is zero-stable then its discrete adjoint is zero-stable. A consequence of the discrete adjoint zero-stability (18) is that small perturbations of $(\partial g/\partial y)^T\,(y_N)$ lead to only small changes in the adjoint initial value λ_0.

4 Derivatives at the Initial Time

We now prove a remarkable property of the discrete LMM adjoints. Even if the discrete adjoint variables λ_n are poor approximations of the continuous adjoints $\overline{\lambda}(t_n)$ at the intermediate grid points, the discrete adjoint at the initial time converges to the continuous adjoint variable with the same order as the original LMM.

Theorem 6 (Consistency at the initial time).

Consider a LMM (7a)–(7c) convergent of order p, and initialized with linear numerical methods. (This covers the typical situation where the initialization procedures $\theta_1, \cdots, \theta_{k-1}$ are Runge Kutta or linear multistep numerical methods). The numerical solutions at the final time are such that

$$\left\|y^h_{N(h)} - y\left(t_{\rm end}\right)\right\|_\infty = \mathscr{O}\left(|h|^p\right)\,, \quad \forall h\,:\, |h| \le H\,,$$

for a small enough H. Let λ_n^h be the solution of the discrete LMM adjoint process (10a)–(10e).

Then the discrete adjoint solution λ_0^h is an order p approximation of the continuous adjoint $\lambda(t_0)$ at the initial time, i.e.

$$\left\| \lambda_0^h - \overline{\lambda}(t_0) \right\|_\infty = \mathscr{O}(|h|^p) \ , \quad \forall h \ : \ |h| \leq H \ , \tag{19}$$

for a small enough H.

Proof. The proof is based on the linearity of the LMM and of its starting procedures, which makes the tangent linear LMM to be the same as the LMM applied to solve the tangent linear ODE. The tangent linear LMM solves the entire sensitivity matrix as accurately as it solves for the solution. which leads to an order p approximation of the full sensitivity matrix.

The continuous sensitivity matrix $S(t) \in \mathbb{R}^{d\times d}$ contains the derivatives of the ODE solution components at time t with respect to the initial value components. The discrete sensitivity matrix $Q_n \in \mathbb{R}^{d\times d}$ contains the derivatives of the numerical solution components at (the discrete approximation time) t_n with respect to the initial value components. These matrices are defined as

$$S^{i,j}(t) = \frac{\partial y^i(t)}{\partial y^j(t_{\text{ini}})} \ , \quad \left(Q_n^h\right)^{i,j} = \frac{\partial y_n^i}{\partial y_0^j} \ , \quad 1 \leq i,j \leq d \ .$$

Superscripts are indices of matrix or vector components.

The entire sensitivity $d\times d$ matrix $S(t_{\text{end}})$ can be obtained column by column via d forward solutions of the *tangent linear ODE model* initialized with $\delta y(t_{\text{ini}}) = \mathbb{e}_j$. It is well known that the tangent linear model of a linear numerical methods gives the same computational process as the numerical method applied to the tangent linear ODE. Since both the initialization steps θ_i and the LMM are linear numerical methods we have that $Q_{N(h)}^h$ is a numerical approximation of S obtained by applying the method (7a)–(7c) to the tangent linear ODE. Since the LMM method is convergent with order p we have that $\|Q_{N(h)}^h - S(t_{\text{end}})\|_\infty = \mathscr{O}(|h|^p) \ \forall h : |h| \leq H$.

The continuous and discrete adjoint variables at the initial time are

$$\overline{\lambda}(t_{\text{ini}}) = S^T(t_{\text{end}}) \cdot \left(\frac{\partial g}{\partial y}(y(t_{\text{end}}))\right)^T \ , \quad \lambda_0 = \left(Q_{N(h)}^h\right)^T \cdot \left(\frac{\partial g}{\partial y}\left(y_{N(h)}^h\right)\right)^T \ .$$

Their difference is

$$\begin{aligned} \overline{\lambda}(t_{\text{ini}}) - \lambda_0 = & \left(S(t_{\text{end}}) - Q_{N(h)}^h\right)^T \cdot \left(\frac{\partial g}{\partial y}(y(t_{\text{end}}))\right)^T \\ & + \left(Q_{N(h)}^h\right)^T \cdot \left(\frac{\partial g}{\partial y}(y(t_{\text{end}})) - \frac{\partial g}{\partial y}\left(y_{N(h)}^h\right)\right)^T \ . \end{aligned} \tag{20}$$

Taking infinity norms in (20), using the smoothness of g, the convergence of the LMM, and the fact that $\|\partial g/\partial y(y(t_{\text{end}}))\|_\infty$ is independent of the discretization h, leads to the bound (19).

□

5 Numerical Experiments

We illustrate the theoretical findings with numerical results on the Arenstorf orbit with the parameters and the initial conditions presented in [4]. We consider the adjoints of the cost functional

$$\Psi = g\big(y(t_{\rm end})\big) = y^1\left(t_{\rm end}\right) \quad \text{where} \quad \left(\frac{\partial g}{\partial y}\big(y(t_{\rm end})\big)\right)^T = \mathbb{e}_1 \,.$$

For the integration we choose the explicit Adams-Bashforth methods of order two (AB2) and three (AB3) and the second order BDF2 method. AB2 is initialized with the forward Euler method, AB3 is initialized with a second order explicit Runge Kutta method, and BDF2 is initialized with the implicit Euler method. This allows each method to converge at the theoretical order. The simulations are performed in Matlab. The reference solutions for the Arenstorf system and its continuous adjoint ODE are obtained with the ode45 routine with the tight tolerances RelTol = 1.e-8, AbsTol = 1.e-8. The root mean square (RMS) norms of the difference between the numerical adjoint solution $(\lambda_n)_{\rm num}$ and the reference continuous adjoint solution $(\overline{\lambda}_n)_{\rm ref}$ at each time moment define instantaneous errors E_n, $n=0,\cdots,N$. The trajectory errors measure the total difference between the numerical and the reference adjoint solutions throughout the integration interval

$$E_n = \sqrt{\frac{1}{d}\sum_{i=1}^{d}\left(\frac{(\lambda_n^i)_{\rm num} - (\overline{\lambda}_n^i)_{\rm ref}}{(\lambda_n^i)_{\rm ref}}\right)^2}\,, \qquad \|E\| = \sqrt{\frac{1}{N+1}\sum_{n=0}^{N} E_n^2}\,. \qquad (21)$$

We compute the discrete adjoint solutions with $N=$150, 210, 300, 425, 600, 850, and 1200 steps and obtain the errors E_0 and $\|E\|$ against the reference continuous adjoint solution. We then estimate the convergence orders and report them in Table 1

For all cases both the trajectory and the final time errors of the continuous adjoint methods decrease at the theoretical rates [7]. The discrete adjoint BDF2 solution is not consistent with the continuous adjoint solution at intermediate integration times, and the numerical error is heavily influenced by the pattern of step size changes. The

Table 1. Experimental convergence orders for different continuous and discrete adjoints. We consider both the trajectory error $\|E\|$ and the initial time error E_0.

	Continuous Adjoint			Discrete Adjoint		
	AB2	BDF2	AB3	AB2	BDF2	AB3
$\|E\|$, fixed steps	1.99	1.99	2.94	0.97	0.00	1.00
E_0, fixed steps	2.00	2.00	2.96	1.99	2.00	2.94
$\|E\|$, fixed steps, modified initialization/termination	–	–	–	0.97	1.99	1.00
E_0, fixed steps, modified initialization/termination	–	–	–	0.97	2.00	1.01
$\|E\|$, variable steps	2.03	2.03	2.98	1.01	-0.01	1.01
E_0, variable steps	2.00	2.00	2.96	1.99	2.00	2.94

fixed step BDF2 adjoint is not consistent with the adjoint ODE due to initialization and termination procedures. When these steps are changed the solution converges at second order. The discrete AB2 and AB3 adjoints converge to the adjoint ODE solution at first order. For all methods the discrete adjoints at the initial time convergence at the theoretical order of the forward methods.

6 Conclusions

In this paper we derive the discrete adjoints of linear multistep formulas and have analyzed their consistency properties. Discrete adjoints are very popular in optimization and control since they can be constructed automatically by reverse mode automatic differentiation.

In general the discrete LMM adjoints are not consistent with the adjoint ODE along the trajectory when variable time steps are used. If the forward LMM integration is zero-stable then the discrete adjoint process is zero-stable as well. For fixed time steps the discrete adjoint steps are consistent with the adjoint ODE at the internal grid points but not at the initial and terminal points. The initialization and termination steps in the fixed step discrete adjoint process can be changed to obtain consistent schemes. The discrete adjoints at the initial time, however, converge to the continuous adjoint at a rate equal to the convergence order of the original LMM. This remarkable property is due to the linearity of the method and of its initialization procedure. Numerical tests on the Arenstorf orbit system confirm the theoretical findings.

Future work will be devoted to the error analysis of discrete adjoints in the case of stiff systems. The effect of automatic differentiation on step-size control mechanisms will also be considered in a follow-up work.

Acknowledgement. This work was supported by the National Science Foundation (NSF) through the awards NSF CAREER ACI-0413872, NSF ITR AP&IM 020519, and NSF CCF–0515170, by the National Oceanic and Atmospheric Administration (NOAA) and by the Texas Environmental Research Consortium (TERC). The author thanks Mihai Alexe who helped with running the TAMC automatic differentiation on some of the codes.

References

1. Baguer, M., Romisch, W.: Computing gradients in parametrization-discretization schemes for constrained optimal control problems. Approximation and Optimization in the Caribbean II. Peter Lang, Frankfurt am Main (1995)
2. Giles, M.: On the use of Runge-Kutta time-marching and multigrid for the solution of steady adjoint equations. Technical Report NA00/10, Oxford University Computing Laboratory (2000)
3. Hager, W.: Runge-Kutta methods in optimal control and the transformed adjoint system. Numerische Mathematik **87**(2), 247–282 (2000)

4. Hairer, E., Norsett, S., Wanner, G.: Solving Ordinary Differential Equations I. Nonstiff Problems. Springer-Verlag, Berlin (1993)
5. Hairer, E., Wanner, G.: Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems. Springer-Verlag, Berlin (1991)
6. Sandu, A.: On the Properties of Runge-Kutta Discrete Adjoints. In: Lecture Notes in Computer Science, vol. LNCS 3994, Part IV, pp. 550–557. "International Conference on Computational Science" (2006)
7. Sandu, A.: On consistency properties of discrete adjoint linear multistep methods. Tech. Rep. CS–TR–07–40, Computer Science Department, Virginia Tech (2007)
8. Sandu, A., Daescu, D., Carmichael, G.: Direct and adjoint sensitivity analysis of chemical kinetic systems with KPP: I – Theory and software tools. Atmospheric Environment **37**, 5083–5096 (2003)
9. Walther, A.: Automatic differentiation of explicit Runge-Kutta methods for optimal control. Computational Optimization and Applications **36**(1), 83–108 (2007)

Call Tree Reversal is NP-Complete

Uwe Naumann

LuFG Informatik 12, Department of Computer Science, RWTH Aachen University, Aachen, Germany, naumann@stce.rwth-aachen.de

Summary. The data flow of a numerical program is reversed in its adjoint. We discuss the combinatorial optimization problem that aims to find optimal checkpointing schemes at the level of call trees. For a given amount of persistent memory the objective is to store selected arguments and/or results of subroutine calls such that the overall computational effort (the total number of floating-point operations performed by potentially repeated forward evaluations of the program) of the data-flow reversal is minimized. CALL TREE REVERSAL is shown to be NP-complete.

Keywords: Adjoint code, call tree reversal, NP-completeness

1 Background

We consider implementations of multi-variate vector functions $F : \mathbb{R}^n \to \mathbb{R}^m$ as computer programs $\mathbf{y} = F(\mathbf{x})$. The interpretation of reverse mode automatic differentiation (AD) [8] as a semantic source code transformation performed by a compiler yields an adjoint code $\bar{\mathbf{x}} += \bar{F}(\mathbf{x}, \bar{\mathbf{y}})$. For given $\mathbf{x}$ and $\bar{\mathbf{y}}$ the vector $\bar{\mathbf{x}}$ is incremented with $(F'(\mathbf{x}))^T \cdot \bar{\mathbf{y}}$ where $F'(\mathbf{x})$ denotes the Jacobian matrix of F at $\mathbf{x}$. Adjoint codes are of particular interest for the evaluation of large gradients as the complexity of the adjoint computation is independent of the gradient's size. Refer to [1, 2, 3, 4] for an impressive collection of applications where adjoint codes are instrumental to making the transition form pure numerical simulation to optimization of model parameters or even of the model itself.

In this paper we propose an extension to the notion of joint call tree reversal [8] with the potential storage of the results of a subroutine call. We consider call trees as runtime representations of the interprocedural flow of control of a program. Each node in a call tree corresponds uniquely to a subroutine call.[1] We assume that no checkpointing is performed at the intraprocedural level, that is, a "store-all" strategy is employed inside all subroutines. A graphical notation for call tree reversal

[1] Generalizations may introduce nodes for various parts of the program, thus yielding arbitrary checkpointing schemes.

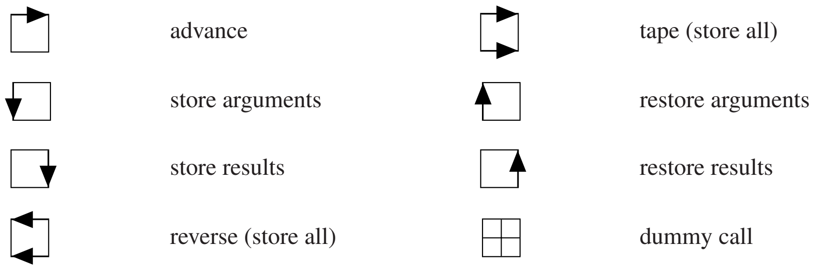

Fig. 1. Calling modes for interprocedural data-flow reversal.

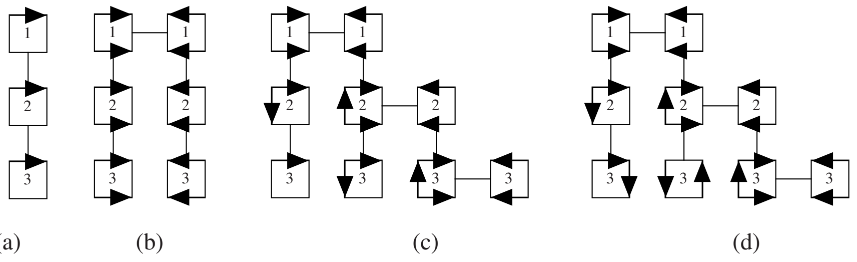

Fig. 2. Interprocedural data-flow reversal modes: Original call tree (a), split reversal (b), joint reversal with argument checkpointing (c), joint reversal with result checkpointing (d).

under the said constraints is proposed in Fig. 1. A given subroutine can be executed without modifications ("advance") or in an augmented form where all values that are required for the evaluation of its adjoint are stored (taped) on appropriately typed stacks ("tape (store all)"). We refer to this memory as the *tape* associated with a subroutine call, not to be confused with the kind of tape as generated by AD-tools that use operator overloading such as ADOL-C [9] or variants of the differentiation-enabled NAGWare Fortran compiler [14]. The arguments of a subroutine call can be stored ("store arguments") and restored ("restore arguments"). Results of a subroutine call can be treated similarly ("store results" and "restore results"). The adjoint propagation yields the reversed data flow due to popping the previously pushed values from the corresponding stacks ("reverse (store all)"). Subroutines that only call other subroutines without performing any local computation are represented by "dummy calls." For example, such wrappers can be used to visualize arbitrary checkpointing schemes for time evolutions (implemented as loops whose body is wrapped into a subroutine). Moreover they occur in the reduction used for proving CALL TREE REVERSAL to be NP-complete. Dummy calls can be performed in any of the other seven modes.

Figure 2 illustrates the reversal in split (b), classical joint (c), and joint with result checkpointing (d) modes for the call tree in (a). The order of the calls is from left to right and depth-first.

For the purpose of conceptual illustration we assume that the sizes of the tapes of all three subroutine calls in Fig. 2 (a) as well as the corresponding computational complexities are identically equal to 2 (memory units/floating-point operation (flop) units). The respective calls are assumed to occur in the middle, e.g. the tape associated with the statements performed by subroutine 1 prior to the call of subroutine 2 has size 1. Consequently the remainder of the tape has the same size. One flop unit is performed prior to a subroutine call which is followed by another unit. The size of argument and result checkpoints is assumed to be considerably smaller than that of the tapes. Refer also to footnotes 2 and 3.

Split call tree reversal minimizes the number of flops performed by the forward calculation (6 flop units). However an image of the entire program execution (6 memory units) needs to fit into persistent memory which is infeasible for most relevant problems. This shortcoming is addressed by classical joint reversal (based solely on argument checkpointing). The maximum amount of persistent memory needed is reduced to 4 (half of subroutine 1 plus half of subroutine 2 plus subroutine 3)[2] at the cost of additional 6 flop units (a total of 12 flop units is performed). This number can be reduced to 10 flop units (while the maximum memory requirement remains unchanged[3]) by storing the result of subroutine 3 and using it for taping subroutine 2 in Fig. 2 (d). The impact of these savings grows with the depth of the call tree.

It is trivial to design toy problems that illustrate this effect impressively. An example can be found in the appendix. The computation of the partial derivative of y with respect to x as arguments of the top-level routine f0 in adjoint mode requires the reversal of a call tree (a simple chain in this case) of depth five. The leaf routine f5 is computationally much more expensive than the others. Classical joint reversal takes about 0.6 seconds whereas additional result checkpointing as in Fig. 5 reduces the runtime to 0.25 seconds. These results were obtained on a state-of-the-art Intel PC. The full code can be obtained by sending an email to the author. The use of result checkpointing in software tools for AD such as Tapenade [12], OpenAD [15], or the differentiation-enabled NAGWare Fortran compiler [14] is the subject of ongoing research and development.

Finding an optimal (or at least near-optimal) distribution of the checkpoints or, equivalently, corresponding combinations of split and joint (with argument checkpointing) reversal applied to subgraphs of the call tree has been an open problem for many years. In this paper we show that a generalization of this problem that allows for subsets of subroutine arguments and/or results to be taped is NP-complete. Hence, we believe that the likelihood of an efficient exact solution of this problem is low. Heuristics for finding good reversal schemes are currently being developed in collaboration with colleagues at INRIA, France, and at Argonne National Laboratory, USA.

[2] ...provided that the size of an argument checkpoint of subroutine 3 is less than or equal to one memory unit, i.e. $\texttt{sizeof}(argchp_3) \leq 1$, and that $\texttt{sizeof}(argchp_2) \leq 2$.

[3] ...provided that $\texttt{sizeof}(argchp_2) + \texttt{sizeof}(reschp_3) \leq 2$ and $\texttt{sizeof}(argchp_3) + \texttt{sizeof}(reschp_3) \leq 2$, where $\texttt{sizeof}(reschp_i)$ denotes the size of a result checkpoint of subroutine i (in memory units).