ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue
Argonne, Illinois 60439

ANL/MCS-TM-236

Second Argonne Theory Institute on Differentiation of Computational Approximations to Functions

held at Argonne National Laboratory

May 18–20, 1998

edited by

Christian H. Bischof, Peter Eberhard, and Paul D. Hovland

Mathematics and Computer Science Division

Technical Memorandum No. 236

June 1998

This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Computational and Technology Research, U.S. Department of Energy, under Contract W-31-109-Eng-38.

Contents

Introduction

A Theory Institute* on "Differentiation of Computational Approximations to Functions" was held at Argonne National Laboratory on May 18–20, 1998. The institute was organized by Christian Bischof and Paul Hovland of the Mathematics and Computer Science Division at Argonne National Laboratory.

The Theory Institute brought together 38 researchers from the United States, Great Britain, France, and Germany. Mathematicians, computer scientists, physicists, and engineers from diverse disciplines discussed advances in automatic differentiation (AD) theory and software and described benefits from applying AD methods in application areas. These areas include fluid mechanics, structural engineering, optimization, meteorology, and computational mathematics for the solution of ordinary differential equations (ODEs) or differential algebraic equations (DAEs).

This meeting was the fourth workshop dedicated to automatic differentiation. Earlier meetings were the 1991 SIAM conference in Breckenridge, Colorado; the first Argonne Theory Institute on computational differentiation in 1993; and the 1996 SIAM conference in Santa Fe, New Mexico.

AD methods can be used whenever gradient information or higher-order derivative information must be computed. The problem is defined by a computer program (without gradient information) that is able to compute numerical values of some output variables for a given set of input variables. As a result of applying AD methods to this computer program, a new computer program is generated automatically to compute the derivatives of the output variables with respect to the input variables. This—at first glance, astonishing—fact can be easily understood by viewing the program from a compiler angle. A complicated computational sequence is split into a sequence of simple operations. Then, to compute the gradients, the chain rule of differentiation is applied successively to this sequence—completely automatically. The resultant gradients are accurate up to roundoff errors (which are always present in numerical evaluations).

AD tools traditionally work in two different ways to achieve this augmentation of the original program source. One way is to generate from the original source code new source code for the gradients, for example, by code transformation techniques. The other way is to use operator overloading to redefine the required active variables and statements.

The algorithmic underpinnings of AD traditionally are defined by either the forward or the reverse mode. The forward mode is easier to implement, and the time to compute gradients is proportional to the number of independent variables; the reverse mode requires more computer memory, but the computation time for the gradients is independent of the number of independent variables and only a small (< 5) multiple of the time to compute the output variables alone. Both methods require no approximation, and the computed gradients are essentially identical.

*For more information see `http://www.mcs.anl.gov/autodiff/workshop.html`

Recent experience with AD tools has shown that both the implementation and algorithmic frameworks need to be expanded with an eye toward exploiting the strengths of these different approaches. Thus, the myriad possibilities of computing derivatives that arise from the associativity of the chain rule and the challenges for building systems that combine runtime and compile-time techniques provide a fertile ground for future challenging research.

AD methods are widely used for solving problems in which the output variables are computed "directly" from the input variables. However, it is not clear what happens if the output variables are computed iteratively or approximately. That is, do "differentiate" and "approximate" commute? To discuss this issue, the workshop was dedicated to the differentiation of computational approximations to functions. Some observations and experiences indicate that AD tools often compute correct gradients for approximations without any human modification, but some problems have been reported that require a closer investigation.

During the three-day workshop, 20 talks were presented and extensively discussed. In addition, lively discussions took place during the breaks.

A. Griewank, one of the pioneers in AD, set the stage by describing in his introductory talk the turbulent history of automatic differentiation during the past few decades. Using the table of contents of his soon-to-appear book on automatic differentiation, he also summarized current AD-related research.

Next, H.-G. Bock discussed the problem of evaluating gradients for functions computed by numerical time integration of ODEs. He presented an internal differentiation method based on a selective activation of variables during time integration in specialized algorithms. This issue was also discussed by P. Eberhard, who described the differentiation of general-purpose integration routines for ODEs used in multibody dynamics, and discussed potential sources of problems and ways to overcome them. Although this black-box integration often is feasible, manual deactivation of differentiation of adaptive elements (e.g., stepsize control) sometimes is required.

In a related context, L. Petzold explored the use of AD for the solution of DAEs. Typically, inaccurate finite differences are used within DAE solvers, but it has been shown that the gradients can be computed more efficiently and accurately with AD. For often-arising problem structures in numerical integration schemes, A. Verma proposed methods to exploit the structure of continuous problems that, for example, maintain the sparsity of problems.

As an alternative to DAE methods solely relying on first-order derivative information, J. Pryce described the use of Taylor series expansions to solve smooth DAEs. AD is used to compute the Taylor coefficients, and interesting consistency conditions can be derived. The capability of AD to compute higher-order derivatives accurately and efficiently also underpins the work of M. Berz and K. Makino. Berz presented rigorous methods for obtaining verified results for the simulation of particle accelerators using higher-order remainder algebra. Tight bounds are obtainable where the accuracy is determined by the order of the AD-computed Taylor coefficients. Makino described applications of these techniques to verified quadrature and the integration of ODEs.

Several talks were motivated by applications illustrating the need for accurate derivatives in engineering contexts and the potential savings that can be achieved by AD-based approaches. O. Pironneau gave an introduction to shape optimization applied to wing design and brake water design. The results are very sensitive: it is crucial to detect and to avoid unphysical designs. B. Mohammadi discussed aerodynamical flow control problems and the use of AD tools for shape optimization. Fixed as well as moving boundaries are considered to control the flow. In this context, Mohammadi considered it sufficient and more efficient to compute only an approximate gradient, but, of course, one must be able to rely on the "leading digits." T. Slawig also investigated shape optimization of airfoils and illustrated the benefits of computing the the gradient matrix in the nonlinear BFGS Quasi-Newton method via AD. G. Corliss and J. Walters presented a rigorous global search procedure using interval arithmetics and applications in economics. Derivatives must be computed for both floating-point and interval arithmetics. Fortran 90 is used to interpret code lists or to do code transformation.

Two talks discussed Burger's equation, which is used for advection-diffusion systems in meteorology and fluid dynamics. S. K. Park discussed applications of AD in weather modeling and presented a new numerical scheme for data assimilation. A. Walther described investigations on computing adjoints of discretizations of Burger's equation. An optimal check-pointing scheme was applied successfully to overcome memory restrictions in the reverse mode.

While it is tempting to use AD tools in a black-box mode, this is not the most efficient use of the technology. In his talk, B. Christianson stated that applications must expose more of their structure to AD tools. Such an approach not only yields more efficient code, as shown in examples, but also can help the researcher become aware of potential pitfalls. Moreover, in some cases, black-box mode may be impossible, as illustrated by C. Faure in her talk on the application of the AD tool Odyssee to an industrial thermohydraulic code. Faure showed how to overcome problems such as inaccessible sources for software libraries. She also showed that the results are mostly very accurate, but for some input parameters unphysical oscillations in the derivatives are observed.

S. Brown addressed the design of future AD tools. These tools must offer a high degree of flexibility, for example, to perform mixed-mode computations or specialized evaluation strategies. There seems to be a trend leading from general black-box tools to more specialized tools where user knowledge is introduced to optimize time and memory efficiency. AD tool design was also the focus of N. Di Cesare's talk. He discussed an AD implementation using operator overloading and expression templates in C++. This approach leads to a very flexible tool, at the expense of longer compilation times. The computation of second-order derivatives was the topic of J. Abate's talk. Whereas it is possible to repeatedly use AD tools to compute Hessians, one can do much better by exploiting the structure of the computation. If only Hessian-vector products are required, even higher savings are possible.

Lastly, P. Hovland's talk illustrated the benefits that arise from the differentiation of an algorithm. He described the application of AD to a successive overrelaxation algorithm to solve linear equations, where the relaxation parameter was adaptively modified to improve the convergence. To obtain improved values, Hovland used ADIFOR to compute the derivative of a cost function with respect to the relaxation parameter.

To summarize, there has been a big change in the AD community during the past decade. In the first phase, it was necessary to understand the basic mechanisms and applications of AD and to develop a "common language," enabling scientists from many different areas to communicate. In the second phase, AD was used successfully in some large applications; and, based on this experience, intensive discussions about the best way to design and implement tools were started. Perhaps we have now entered the third phase, in which all basic methods and some efficient tools are available and AD has proven its reliability in some very complex problems. Nevertheless, many open questions remain, and scientists from all over the world are actively seeking solutions and pushing the envelope in this rapidly developing field and by its very nature interdisciplinary field. The hope is that in the not-too-distant future, derivatives can be computed so efficiently and conveniently that nobody will understand why, in the late twentieth century, this task was such a serious problem in many fields of research.

Peter Eberhard
Chris Bischof
Paul Hovland

Acknowledgments

The editors express their deep appreciation and thanks to Judy Beumer for her tireless work both in the organization of the institute and in the publication of this report. We also thank Boyana Norris for her assistance in organizing the institute. Finally, we thank Gail Pieper for numerous suggestions that greatly improved the organization and readability of this report.

Agenda

Monday, May 18

8:30	Welcome	
9:00	Andreas Griewank	"Introducing Computational Differentiation by Book"
9:45	Hans-Georg Bock	"Internal Numerical Differentiation Revisited: Differentiation of Adaptive Integrators for ODE and DAE, Including Discontinuous Models"
10:30	*Break*	
11:00	Linda Petzold	"Automatic Differentiation in the Numerical Solution and Sensitivity Analysis of Differential Algebraic Equations"
11:30	Arun Verma	"Structured Automatic Differentiation for Numerical Integration Methods"
12:00	*Lunch*	
1:30	Andrea Walther	"Adjoining Discretizations of Burgers' Equation"
2:00	Seon Ki Park	"Fast 4-D Variational Data Assimilation Using an Inverse Linear Model"
2:30	*Break*	
3:00	Peter Eberhard	"Differentiation of Ordinary Differential Equations and Numerical Integration Algorithms"
3:45	John Pryce	"AD Aspects of Solving DAEs by Taylor Series"
4:30	Discussion	
5:00	Group picture	
6:00	Barbecue	Location: Argonne Park

Tuesday, May 19:

9:00	Martin Berz	"Higher-Order Methods and Adjoined Remainder Bounds"
9:45	Olivier Pironneau	"Optimal Shape Design and Automatic Differentiation"
10:30	*Break*	
11:00	Christèle Faure	"Oscillation in the Derivatives?"
11:30	Thomas Slawig	"Shape Optimization of a Multi-Element Airfoil Using AD"
12:00	*Lunch*	Location: Cafeteria, Dining Room B
1:30	Stephen Brown	"Toward Second-Generation Automatic Differentiation"
2:00	Nicolas Di Césaré	"A Flow Control Problem Using Automatic Differentiation in C++"
2:30	*Break*	
3:00	Bijan Mohammadi	"Analysis of Shape Optimization and Flow Control Control Problems Using an AD Tool"
3:45	George Corliss	"CD in Support of Rigorous Global Optimization"
4:30	Discussion	
6:00	Conference Dinner	Location: Argonne Guest House

Wednesday, May 20:

9:00	Bruce Christianson	"How to Make Differentiate and Approximate Commute"
9:45	Kyoko Makino	"Differential Algebraic Methods on Taylor Models"
10:30	*Break*	
11:00	Paul Hovland	"Adaptive SOR: Differentiation of Algorithms Can Be Beneficial"
11:30	Jason Abate	"Computing Second-Order Derivatives Using Automatic Differentiation"
12:00	*Lunch*	
1:30	Final Discussion	
3:00	Group Discussions	

List of Attendees

Jason Abate
Texas Inst. for Computational & Applied Mathematics
The University of Texas at Austin
2.400 Taylor Hall
Austin, TX 78712
Tel: (512) 471-1721
Fax: (512) 471-8694
E-Mail: abate@ticam.utexas.edu

Martin Berz
Department of Physics
Michigan State University
East Lansing, MI 48824
Tel: (517) 333-6313
Fax: (517) 353-5967
E-Mail: berz@pilot.msu.edu

Christian Bischof
Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, IL 60439
Tel: (630) 252-8875
Fax: (630) 252-5986
E-Mail: bischof@mcs.anl.gov

Hans-Georg Bock
Interdisciplinary Center for Scientif. Computing
University of Heidelberg
Im Neuenheimer Feld 368
69120 Heidelberg
Germany
Tel: (+49) 6221-54-8237
Fax: (+49) 6221-54-5444
E-Mail: bock@iwr.uni-heidelberg.de

Stephen Brown
Numerical Optimisation Centre
University of Hertfordshire
College Lane, Hatfield
Hertfordshire, AL10 9AB
England
Tel: (+44) 1707 285113
Fax: (+44) 1707 284303
E-Mail: s.brown@hert.ac.uk

Martin Bücker
Central Inst. for Applied Mathematics (ZAM)
Research Centre Jülich
D-52425 Jülich
Germany
Tel: (+49) 2461 61 6753
Fax: (+49) 2461 61 6656
E-Mail: m.buecker@fz-juelich.de

Alan Carle
Rice University
CITI/CRPC – MS41
6100 S. Main Street
Houston, Texas 77005–1892
Tel: (713) 285-5368
Fax: (713) 285-5136
E-Mail: carle@cs.rice.edu

Bruce Christianson
Computer Science
University of Hertfordshire
Hatfield
Hertfordshire AL10 9AB
England
Tel: (+44) 1707 284335
Fax: (+44) 1707 284303
E-Mail: b.christianson@herts.ac.uk

George F. Corliss
Dept. Math, Stat., Comp. Sci
Marquette University
P.O. Box 1881
Milwaukee, WI 53201
Tel: (414) 288-6599
Fax: (414) 288-5472
E-Mail: georgec@mscs.mu.edu

Diane Crann
Dept. of Mathematics
University of Hertfordshire
Hatfield Campus
Hertfordshire, AL10 9AB
England
Fax: (+44) 1707 284303
E-Mail: D.S.Cran@herts.ac.uk

Alan Davies
Dept. of Mathematics
University of Hertfordshire
Hatfield Campus
Hertfordshire, AL10 9AB
England
Tel: (+44) 1707 284385
Fax: (+44) 1707 284303
E-Mail: a.j.davies@herts.ac.uk

Nicolas Di Césaré
Laboratoire d'Analyse Numerique, BP 187
Tour 55-65
Université Pierre et Marie Curie (PARIS VI)
4, place Jussieu
75252 Paris Cedex 05
France
Tel: (+33) 1 44 27 42 98
Fax: (+33) 1 44 27 72 00
E-Mail: nicolas.dicesare@ann.jussieu.fr

Peter Eberhard
Institute B of Mechanics
University of Stuttgart
Pfaffenwaldring 9
70550 Stuttgart
Germany
Tel: +49 711-685-6392
Fax: +49 711-685-6400
E-Mail: pe@mechb.uni-stuttgart.de

Michael Fagan
Rice University
CITI/CRPC – MS41
6100 S. Main Street
Houston, Texas 77005–1892
Tel: (713) 285-2733
Fax: (713) 285-5136
E-Mail: fagan@cs.rice.edu

Christèle Faure
INRIA
2004, Rue des Lucioles
B.P. 93
06902 Sophia Antipolis Cedex
France
Tel: (+33) 4-92387908
Fax: (+33) 4-93657633
E-Mail: christele.faure@sophia.inria.fr

Mark Final
Computer Science
University of Hertfordshire
Hatfield
Hertfordshire AL10 9AB
England
Tel: (+44) 1707 284766
Fax: (+44) 1707 284303
E-Mail: M.Final@herts.ac.uk

Andreas Griewank
Inst. of Scientific Computing
Technical University Dresden
Mommsenstr. 13
D-01062 Dresden
Germany
Tel: +49 351-464266
Fax: +49 351-4637114
E-Mail: griewank@math.tu-dresden.de

Alan Hindmarsh
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
P.O. Box 808, L-561
Livermore, CA 94551
Tel: (925) 422-4276
Fax: (925) 423-2993
E-Mail: alanh@daphne.llnl.gov

Jens Hoefkens
National Superconducting Cyclotron Laboratory
Michigan State University
East Lansing, MI 48824
Tel: (517)-333-6441
Fax: (517)-333-6320
E-Mail: hoefkens@pilot.msu.edu

Paul Hovland
Mathematics and Computer Science Division
Argonne National Laboratory
9700 South Cass Avenue
Argonne, IL 60439
Tel: (630) 252-6384
Fax: (630) 252-5986
E-Mail: hovland@mcs.anl.gov

David Juedes
School of Electrical Engineering
and Computer Science
Ohio University
Morton Hall
Athens, OH 47601
Tel: (740) 593-1596
Fax: (740) 593-0406
E-Mail: juedes@ohiou.edu

Kyoko Makino
Department of Physics
NSCL/Cyclotron Laboratory
Michigan State University
East Lansing, MI 48824-1321
Tel: (517) 333-6441
Fax: (517) 353-5967
E-Mail: makino@nscl.msu.edu

Hristo Mitev
Institute of Scientific Computing
Technical University Dresden
Mommsenstr. 13
D-01062 Dresden
Germany
Tel: +49 351-464266
Fax: +49 351-4637114
E-Mail: mitev@math.tu-dresden.de

Bijan Mohammadi
Mathematics Department CC51
University Montpellier II
34095 Montpellier Cedex 5
France
Tel: +33 (4) 67-14-35-62
Fax: +33 (4) 67-14-35-58
E-Mail: bijan.mohammadi@inria.fr

Boyana Norris
Mathematics and Computer Science Division
Argonne National Laboratory
9700 South Cass Avenue
Argonne, IL 60439
Tel: (630) 252-4279
Fax: (630) 252-5986
E-Mail: norris@mcs.anl.gov

Seon Ki Park
Inst. for Mesoscale Meteorological
Studies (CIMMS)
University of Oklahoma
Room 1110, Sarkeys Energy Center
100 East Boyd
Norman, OK 73919
Tel: (405) 325-0402
Fax: (405) 325-7614
E-Mail: skpark@bluesky.ecas.ou.edu
http://wwwcaps.ou.edu/ spark/

Linda Petzold
Dept. of Mechanical and Environmental
Engineering
Engr. 2 Bldg., Room 2355
University of California, Santa Barbara
Santa Barbara, CA 93106
Tel: (805) 893-5362
Fax: (805) 893-8651
E-Mail: petzold@engineering.ucsb.edu

Udo Piram
Institute B of Mechanics
University of Stuttgart
Pfaffenwaldring 9
70550 Stuttgart
Germany
Tel: +49 711-685-6395
Fax: +49 711-685-6400
E-Mail: up@mechb.uni-stuttgart.de

Olivier Pironneau
Laboratoire d'Analyse Numerique,
B.C. 187, Tour 55-65
University Pierre et Marie Curie
4, place Jussieu
75252 Paris FRANCE
Tel: (+33) 1 44274298
Fax: (+33) 1 44277200
E-Mail: pironneau@ann.jussieu.fr

John D. Pryce
Computer Info. Systems Engineering Dept.
Royal Military College of Science
Shrivenham
Swindon SN6 8LA
UK
Tel: +44 (0)1793-785683
Fax: +44 (0)1793-785366
E-Mail: pryce@gw.rmcs.cranfield.ac.uk

Lucas Roh
Mathematics and Computer Science Division
Argonne National Laboratory
9700 South Cass Avenue
Argonne, IL 60439
Tel: (630) 252-2983
Fax: (630) 252-5986
E-Mail: roh@mcs.anl.gov

Aaron Ross
Mathematics and Computer Science Division
Argonne National Laboratory
9700 South Cass Avenue
Argonne, IL 60439
Tel: (630) 252-1842
Fax: (630) 252-5986
E-Mail: aross@mcs.anl.gov

Thomas Slawig
Technische Universitaet Berlin
Fachbereich Mathematik MA 6-2
Strasse des 17. Juni 136
D-10623 Berlin
Germany
Tel: (+49) 30 31425733
Fax: (+49) 30 31479546
E-Mail: slawig@math.tu-berlin.de

Jean Utke
Framework Technologies Inc.
E-Mail: utke@mcs.anl.gov

Arun Verma
computer Science
4130 Upson Hall
Cornell University
Ithaca, NY 14853
Tel: (607) 254-8807
Fax: (607) 254-8888
E-Mail: verma@cs.cornell.edu

James Walters
Dept. Math., Stat., Comp. Sci.
Marquette University
P.O. Box 1881
Milwaukee, WI 53201
Tel: (414) 288-6599
Fax: (414) 288-5472
E-Mail: walters@mscs.mu.edu

Andrea Walther
Institute of Scientific Computing
Technical University Dresden
Mommsenstr. 13
D-01062 Dresden
Germany
Tel: (+49) 351 4635018
Fax: (+49) 351-4637114
E-Mail: awalther@math.tu-dresden.de

Michael Winckler
Interdisziplinares Zentrum für
Wissenschaftliches Rechnen
Universität Heidelberg
Im Neuenheimer Feld 368
D-69120 Heidelberg
Germany
Tel: (+49) 6221 54 - 8245
Fax: (+49) 6221 54 - 5444
E-Mail: michael.winckler@iwr.uni-heidelberg.de

Photo of Attendees

For a photo of the attendees, send email to beumer@mcs.anl.gov.

Computing Second-Order Derivatives
Using Automatic Differentiation

Jason Abate
abate@ticam.utexas.edu
Texas Institute for Computational and Applied Mathematics
The University of Texas at Austin
Austin, Texas, U.S.A.

Automatic differentiation (AD) provides an efficient and accurate method to obtain derivatives for use in sensitivity analysis, parameter identification, and optimization. Current tools are targeted primarily at computing first-order derivatives, namely, gradients and Jacobians. Prior to AD, derivative values were obtained through divided difference methods, symbolic manipulation, or hand coding, all of which have drawbacks when compared with AD. Accurate second-order derivatives are even harder to obtain than first-order ones; it is possible to end up with no accurate digits in the derivative value when using a divided-difference scheme.

One can repeatedly apply first-order derivative tools to obtain higher-order derivatives, but this approach is complicated and ignores structural information about higher-order derivatives such as symmetry. Additionally, in cases where a full Hessian, H, is not required, such as with Hessian-vector products ($H \cdot V$) and projected Hessians ($V^T \cdot H \cdot W$) where V and W are matrices with many fewer columns than rows, it is possible to compute the desired values much more efficiently than with the repeated differentiation approach.

AD by source transformation provides great flexibility in implementing sophisticated algorithms that exploit the associativity of the chain rule of calculus. Unfortunately, the development of robust source transformation tools is a substantial effort. ADIFOR and ADIC, source transformation tools for Fortran and C, respectively, both implement relatively simple algorithms for propagating derivatives. Most of the development time so far has concentrated on producing tools that handle the full range of the language, rather than on developing more efficient algorithms to propagate derivatives.

No "best" approach to computing Hessians exists; the most efficient approach to computing second-order derivatives depends on the specifics of the code and the target platform on which the code will be run. We have timed the execution of the core Hessian kernel operations on a variety of machines, which results in a machine-specific performance model. This model is used to select between the Hessian algorithms at code-generation time based on the number of independent variables to be used and on the target architecture. This approach has consistently produced more efficient code than have machine-independent strategies. In all cases, however, the generated codes compute derivatives to machine precision on any platform.

To make it easier to experiment with algorithmic techniques, we have developed AIF, the Automatic Differentiation Intermediate Form. AIF acts as the glue layer between a language-specific front-end and a largely language-independent transformation module that implements AD transformations at a high level of abstraction.

We have implemented an AIF-based module for computing second-order derivatives. The Hessian module, as we call it, implements several different algorithms and selectively chooses them in a fashion that is determined by the code presented to it. However, this context-sensitive logic, which is based on machine-specific performance models, is transparent to the AD front-end. The Hessian module currently interfaces with ADIFOR and ADIC.

Higher-Order Methods and Adjoined Remainder Bounds

Martin Berz
berz@pilot.msu.edu
Department of Physics, Michigan State University
East Lansing, Michigan, U.S.A.

An overview of higher-order differentiation methods is given, addressing some questions of implementation as well as some important applications. It is then shown how these methods can be enhanced to also allow rigorous computation of bounds for the remainder of a high-order expansion. Compared with other verified methods, the resulting Taylor Models yield levels of sharpness that scale with a high order of the width of the domain. Furthermore, for extended calculations, the problem of blow-up typical for many interval approaches is significantly reduced. Finally, the methods are advantageous for multidimensional applications in that the expense of higher dimensions increases only modestly. We will illustrate these points with various verified computations, including global optimization of complicated blowup-prone functions.

Internal Numerical Differentiation Revisited: Differentiation of Adaptive Integrators for ODE and DAE, Including Discontinuous Models

Hans-Georg Bock
SciCom@IWR.Uni-Heidelberg.De
IWR (Interdisciplinary Center for Scientific Computing), University of Heidelberg
Heidelberg, Germany

Numerical methods for sensitivity analysis, parameter estimation, optimum experiment design, or optimal control for DAE or ODE models frequently require the computation of first- or second-order derivatives of the solution of initial value problems with respect to initial values and parameters. Multiple shooting procedures, for example, for these kinds of optimization problems, have been developed since the late seventies.

If one uses what we call "external numerical differentiation" (END), one tries to approximate these derivatives by the differentiation of an adaptive numerical integrator (e.g., by finite differences or automatic differentiation). However, such an integrator contains many adaptive components that usually cause nondifferentiabilities or discontinuities of the numerical result of an integration. The latter result, for example, from stepsize and order selection, pivoting in linear system solutions, Newton-type iterations due to projection, implicit schemes, and switching point location. Hence the results of END are generally meaningless, or at least highly inaccurate.

As an alternative one may use what we have called "internal numerical differentiation" (IND). The idea is to compute the derivative of the adaptively chosen discretization scheme for an initial value problem while avoiding differentiation of the adaptive code components. The adaptive procedure and the discretization must be chosen such that the discretization scheme approximates not only the solution, but also its required derivatives with a desired accuracy. It should also be chosen such that the derivative calculation is as efficient as possible.

To implement IND in an initial value problem solver with a certain degree of sophistication usually means, however, that is has to be thoroughly rethought and rewritten. Step and order selection are not differentiated; iterative procedures are reformulated and reinterpreted – in the sense of backward analysis – to allow for application of the implicit function theorem. Since intermediate coefficients, matrices, decompositions, and so on may be used for the computation of both the solution and its derivatives, high computational savings compared with END may be gained. Also, since the exact derivative of an approximate problem solution is computed, IND is stable in the sense of backward analysis and yields useful derivatives even for coarse approximation accuracies.

Details of IND implementations and numerical results will be discussed for various integrators for ODE and DAE.

URLs of interest:

`http://www.iwr.uni-heidelberg.de/ agbock`

`http://www.iwr.uni-heidelberg.de/iwr/im-net`

(Network on Industrial Mathematics (monthly digest))

Toward Second-Generation Automatic Differentiation

Stephen Brown
S.Brown@herts.ac.uk
Numerical Optimisation Centre, University of Hertfordshire
College Lane, Hatfield, Hertfordshire, United Kingdom

The majority of AD tools in use today evaluate derivatives using either the forward or reverse mode. The selection of evaluation strategy is usually based on the number of independent and dependent variables and, to a lesser degree, the structure of the mathematical problem. However, it appears that a trend is emerging whereby AD is applied to problems of increasing complexity (in terms of the number of floating-point operations used to evaluate the dependent variables). In this context, AD applied naively takes too much time (in the case of the forward mode) and too much space (in the case of the reverse mode). It also appears that there is a general agreement among the AD development community that if the challenges of large problem evaluation is to be met, the next generation of tools should provide at least the following:

- Mixed-mode evaluation selected at the program construct level
- Control flow and data analysis
- Specialized evaluation strategies applied to recognized program configurations
- Dynamic mode and evaluation strategy selection
- Efficient use of memory through careful use of temporary variable management, interface checkpointing, and invertible code
- Intuitive user interfaces

It is our view that the information required to achieve these goals is best obtained not by using AD library calls, but by analyzing the intermediate data structures generated by the compilation process.

In this talk, we shall show that if the intermediate data structures are opened out, there may be many exciting opportunities to develop the theoretical foundations of AD (at the program level), harness the complexities of AD, and begin to apply AD code optimizations and transformations.

How to Make
Differentiate and Approximate Commute

Bruce Christianson
B.Christianson@herts.ac.uk
Numerical Optimisation Centre, University of Hertfordshire
Hatfield, United Kingdom

Do we want the derivative of the function we actually calculated, or an approximation to the derivative of the function we are approximating, and when does it matter? We look at some technical results involving iterative processes and suggest the following:

Optimization applications must expose more of their structure to AD tools; in particular, the equations being solved should be explicitly coded, not just the code to solve them.

Optimization codes must signal explicitly to the AD software what use they intend to make of derivative information, and the accuracy to which they require it.

AD software must be capable of exploiting this additional information, and should make available various by-products of the differentiation process.

As an example, we consider the function z_* of u defined by $z_* = f(y)_*)$; $\psi(y_*, u) = 0$.

Let y be an approximate solution to the implicit equation, with $\psi(y, u) = w$ and reverse accumulate $\bar{u} = \partial z/\partial u$.

If, on the reverse pass, we find ζ such that

$$\|\zeta\psi_y(y, u) - \bar{y}\| \leq \|w\|,$$

then

$$(i) \quad z_* = f(y) - \zeta w \text{ to } O(\|w\|^2)$$

and

$$(ii) \quad \frac{\partial z_*}{\partial u} = -\zeta\psi_u(y, u) \text{ to } O(\|w\|).$$

In the linear case, where we solve $Ay = b$ approximately for y, this amounts to finding v such that $\|vA - \bar{y}\| \leq \|Ay - b\|$. Then

$$z_* = f(y) - v(Ay - b); \quad \bar{b} = v; \quad \bar{A} = -yv.$$

Retaining an LU decomposition for A to find v can result in an order-of-magnitude speedup over naive reverse mode.

CD in Support of Rigorous Global Optimization

George F. Corliss
georgec@mscs.mu.edu

James Walters
walters@mscs.mu.edu

Department of Mathematics, Marquette University
Milwaukee, Wisconsin, U.S.A.

R. Baker Kearfott
rbk@usl.edu
Department of Mathematics, University of Southwestern Louisiana
Lafayette, Louisiana, U.S.A.

We are developing for Sun Microsystems a rigorous global search software package using interval arithmetic. The algorithms solve nonlinear systems and (unconstrained, equality or inequality constrained) global optimization problems. We compute narrow boxes in which the roots or optima are guaranteed to lie. As part of the effort, we are working on several industrial applications including problems from MacNeal-Schwindler (finite element analysis), Banc One (portfolio management), Swiss Bank (currency trading), GE Medical Systems (MRI instrument design - ODE PID), and Genome Therapeutics (gene search - neural networks).

Of course, computational differentiation is a fundamental enabling technology. We need gradients of objective function and constraints, Hessian information for the objective, and higher-derivative information for problems whose solutions form manifolds in the solution space. What is unusual about our CD needs is that we must be able to evaluate the derivative objects in both floating-point and interval arithmetics.

Objective functions and constraints are coded in Fortran 90 by using overloaded operators to record the code list. The semantics of IF statements are even less understood for intervals than for CD, so data-dependent branching is supported *only* by our own CHI function. We support user coding of large-scale nodes in the computational graph (e.g., dot product). The code list may be differentiated symbolically and passed to a code optimization step.

From there, we use both interpreted code lists and code transformation techniques. The optimization algorithm may call about 15 different functions, each of which interprets the code list in a different way. Interpreters use either forward or reverse modes; provide first-, second-, or higher-derivative information; and work in either floating-point or interval arithmetic.

Alternatively, we use code templates to generate from the code list either Fortran 90 or g77 + intervals code to replace the 15 different interpreters. That is, instead of generating Fortran code for gradients, we generate seven different Fortran codes for objective, constraints, gradients, Hessians, slopes, and so forth, all in both floating-point and interval arithmetics. Then, the optimization algorithm calls the generated code instead of the interpreters.

In our initial experiments, the global optimization algorithm execution times are roughly in the following ratio:

- Interpreted code list: 1
- Generated F90 code 0.9 - 1.1
- Generated g77 code 0.5 - 0.9

For more details, see
`www.mscs.mu.edu/~globsol/`,
especially `www.mscs.mu.edu/~globsol/Marquette/autodiff.html`.

A Flow Control Problem Using Automatic Differentiation in C++

Nicolas Di Césaré
Nicolas.Dicesare@ann.jussieu.fr
Laboratoire d'Analyse Numérique, Université Pierre et Marie Curie
Paris, France

This work deals with a simple implementation of automatic differentiation in direct mode using operator overloading and expression templates. We apply this tool to an optimal control problem: minimize the drag of a cylinder, in subsonic unsteady turbulent flow, by controlling the boundary conditions of the cylinder. The NSC2KE Fortran code of B. Mohammadi was converted to C++. Then, using the C++ template feature, we provide an easy way to use the code in black box without any modification. The code can be used in "normal" or in "automatic differentiation" mode. Maximum flexibility is achieved.

Differentiation of Ordinary Differential Equations and Numerical Integration Algorithms

Peter Eberhard
pe@mechb.uni-stuttgart.de

Udo Piram
up@mechb.uni-stuttgart.de

Institute B of Mechanics, University of Stuttgart
Stuttgart, Germany

Often dynamical problems are described by sets of differential equations. For complicated nonlinear systems these equations cannot be solved explicitly, but there exist reliable and efficient integration algorithms that help to obtain at least a numerical solution.

For optimization problems where time-dependent criteria have to be optimized, the situation becomes complicated. The problem formulation for complex systems, for example, in vehicle dynamics or robotics, often leads to a large set of nonlinear equations, and the optimization procedures used have to be efficient. Fast converging optimization algorithms always require the computation of sensitivity functions. Thus, not only the system behavior and the criteria have to be evaluated, but also their sensitivities with respect to design variables.

Automatic differentiation tools allow to differentiate arbitrary algorithms given by computer programs, but several pitfalls exist. For such a tool an algorithm is basically determined by its inputs (e.g., the design variables) and its outputs (e.g., the integral type performance criteria). The duty of the tool is then "simply" to find the total derivative of the outputs with respect to the inputs. Obvious problems occur if the derivative of non-differentiable algorithms should be computed, but these ill-posed problems can usually be avoided by the user or can be recognized by the tool using exception handlers. A remaining question is, what happens if iterative or recursive algorithms such as numerical integration algorithms are differentiated?

Several widely used integration schemes have been differentiated and analyzed. From this, several sources of problems have been encountered. Nevertheless, solutions seem to be possible. Mainly the differentiation of the stepsize- and approximation order-control has to be studied and corrected carefully. The results of the generated code is verified using the adjoint variable approach, an efficient and approximation-free method to compute the sensitivities.

In the talk an example is shown where sensitivities for an ICE high-speed train are computed in order to optimize parameters for improved damping of vibrations. Here the ODE for the sensitivities is solved, and no differentiation of the integration algorithm is required. However, in applying this approach, a highly specialized algorithm has to be used.

Finally, we report on current work differentiating NEWSIM, our multibody system simulation program. This will allow us to include properties of deformable bodies into the optimization. Later, this will even require the "differentiation" of a FEM preprocessor that computes some necessary matrices defining, for example, mass and stiffness properties.

Oscillation in the Derivatives?

Christèle Faure
Christele.Faure@sophia.inria.fr
INRIA Sophia Antipolis
Sophia-Antipolis, France

The automatic differentiation tool Odyssee has been used to compute gradients of the industrial thermal-hydraulic in bundles code called Thyc-3D. A first study has demonstrated the feasibility on a mockup of this code called Thyc-1D (see [1, 2] for more details).

Two difficulties have been encountered in differentiating Thyc-1D and Thyc-3D. First, the program has black-box subroutines, whose code is not available. Second, Thyc contains linear solvers implementing iterative algorithms with stopping tests, so that the number of iterations is not known beforehand and depends on the value given to the independent variables. The derivatives of the black-box subroutines have been hand coded, using (accurate) finite differences to approximate the derivatives. For the linear solvers, one can supply the associated subroutines to Odyssee or let it generate the code. Supplying the associated routine simplifies the code generation and makes the resulting code very efficient.

On Thyc-1D, we have demonstrated that using derivatives of the linear solvers generated by Odyssee was as accurate as using hand-coded ones. But is it always the case, or do automatically generated derivatives of such solvers sometimes introduce oscillations on the derivatives?

References

[1] C. Duval, P. Erhard, C. Faure, and J. C. Gilbert. Application of the automatic differentiation tool `odyssée` to a system of thermohydraulic equations. In J.-A. Désidéri, P. Le Tallec, E. Oñate, J. Périaux, and E. Stein, editors, *Proc. of ECCOMAS'96*, volume Numerical Methods in Engineering'96, pages 795–802. John Wiley & Sons, September 1996.

[2] C. Faure and C. Duval. Automatic differentiation for sensitivity analysis: A test case. In K. Chan, S. Tarantola, and F. Campolongo, editors, *Proceedings of Second International Symposium on Sensitivity Analysis of Model Output (SAMO'98)*, volume EUR report 17758, pages 107–110. EN, Luxembourg, 1998.

Introducing Computational Differentiation by Book

Andreas Griewank
griewank@math.tu-dresden.de
Inst. of Scientific Computing
Technical University Dresden
Mommsenstr. 13
D-01062 Dresden, Germany

Bruce Christianson
B.Christianson@herts.ac.uk
Numerical Optimisation Centre, University of Hertfordshire
Hatfield, United Kingdom

The authors are currently involved in a valiant effort to write an introductory yet comprehensive book entitled *Evaluating Derivatives, Principles and Techniques of Computational Differentiation.* Postscript versions of the current draft can be obtained in the directory ftp/pub/cdbook on ubtj02.math.tu-dresden.de. Comments and suggestions are needed and appreciated, especially with regard to the basic framework and terminology. Issues of particular concern are the relation between mathematical variables and program variables, models for the quantification of computational complexity, and the discussion of implementation strategies for the forward and reverse mode.

Adaptive SOR: Differentiation of Algorithms Can Be Beneficial

Paul Hovland
hovland@mcs.anl.gov
Mathematics and Computer Science Division, Argonne National Laboratory
Argonne, IL, U.S.A.

Michael Heath
Dept. of Computer Science, University of Illinois at Urbana-Champaign
Urbana, IL, U.S.A.

Many science and engineering codes include parameters that affect their behavior and running time. An example is the relaxation parameter ω in the successive overrelaxation (SOR) algorithm for solving systems of linear equations. If some figure of merit, such as a residual, is differentiated with respect to such a parameter, the resulting information can be used to adjust the parameter to a more favorable value. We apply this idea to formulate a new adaptive SOR algorithm, using automatic differentiation (AD) to obtain the necessary derivatives. Unlike previous adaptive SOR algorithms, which attempt to achieve good asymptotic behavior by approximating the largest eigenvalue of the Jacobi iteration matrix, our algorithm seeks maximum near-term improvement in the residual. Empirical results indicate that under the conditions that prevail when SOR is used as a preconditioner, this new adaptive SOR algorithm converges faster than standard SOR using the asymptotically optimal value for ω. Other parameterized algorithms may benefit from similar use of AD.

Differential Algebraic Methods on Taylor Models

Kyoko Makino and Martin Berz
makino@nscl.msu.edu and berz@pilot.msu.edu
Department of Physics, NSCL/Cyclotron Laboratory, Michigan State University
East Lansing, Michigan, U.S.A.

The method of Taylor models is augmented to a differential algebraic framework by the introduction of the operator del inverse for integration. Within this framework, several problems typically addressed by differential algebraic techniques can be treated, including one- and higher-dimensional quadrature, the solution of ODEs and their flows, and the solution of PDEs. We begin with applications in verified quadrature of up to four dimensions. Next, a framework for the verified integration of ODEs is developed. Based on Schauder's fixed-point theorem and other functional analysis tools, it allows the rigorous bounding of solutions, including their dependence on initial conditions. Applications from the field of beam physics are given.

Analysis of Shape Optimization and Flow Control Problems Using an AD Tool

Bijan Mohammadi
Bijan.Mohammadi@inria.fr
University of Montpellier and INRIA
Montpellier, France

We aim to show how to use a gradient-based shape optimization tool, first designed for steady configuration, as a tool for aerodynamical flow control. The gradients are provided, in discrete level, using automatic differentiation by program. We use AD tools to analyze the different contribution to the gradient. This approach showed that in the evaluation of the gradients of cost functions and constraints lying on the shape, the sensitivity with respect to the state can be neglected.

Cost functions being often based on boundary integrals, the previous situation is currently the case in applications. A cheap evaluation of the gradient avoiding the flow solver differentiation is possible.

In the past, we have used this approach and the approximate gradient for shape optimization on various 2 and 3D incompressible and compressible configurations of inviscid and viscous turbulent flows [1,2,3].

In control problems, the aim is to minimize some unsteady cost function using the same approximation of the gradient we use for steady flows. This gives a control law to be applied by available control devices: piezoelectric or injection/suction devices. However, for a control law to be efficient and realizable by such control mechanism, the amount of the required deformation or injection/suction velocity has to be as small as possible. For this reason, we use transpiration boundary conditions rather than an ALE formulation to simulate the equivalent shape deformation.

Using the approximate gradient we introduced above, we obtain control laws to be applied through existing control devices (e.g., injection or piezoelectric). This is therefore a cheap alternative to feedback laws obtained through control theory. One advantage is that this approach does not require any evaluation of the flow to build the transfer function. Indeed, the control law is built in real time.

This approach has been used to control various 2 and 3D inviscid and viscous turbulent flows. Both fixed and moving geometries have been considered. The moving geometry case is based on either forced movements of the structure or on aeroelastic simulations. The elastic behavior of the structure has been taken into account by using a spring-based elastic model coupling structure movements and aerodynamical forces.

References

[1] B. Mohammadi (1997), *Practical Applications to Fluid Flows of Automatic Differentiation for Design Problems*, VKI lecture series, 1997–05.

[2] B. Mohammadi (1997), *A New Optimal Shape Design Procedure for Inviscid and Viscous Turbulent Flows*, Int. J. for Numerical Meth. in Fluid, vol. 25, 183–203.

[3] G. Medic, B. Mohammadi, and S. Moreau (1998), *Optimal Airfoil and Blade Design in Compressible and Incompressible Flows*, AIAA-98-124.

Fast 4-D Variational Data Assimilation Using an Inverse Linear Model

Seon Ki Park
spark@rossby.ou.edu
Cooperative Institute for Mesoscale Meteorological Studies, University of Oklahoma

Eugenia Kalnay
ekalnay@rossby.ou.edu
School of Meteorology, University of Oklahoma
Norman, Oklahoma, U.S.A.

The idea of accelerating 4DVAR (four-dimensional variational data assimilation) using the inverse TLM (tangent linear model) has been introduced by Wang et al. (1997), and made more general and far simpler by Kalnay and Pu (1998). The inverse TLM means running the TLM backward in time (i.e., with negative time steps). In the backward integration, the quasi-inverse TLM includes the dissipative terms with the sign reversed to suppress numerical instability, while the exact-inverse TLM includes them with the sign unreversed.

We have developed an algorithm for the fast 4DVAR and tested with a simple advection-diffusion system (Burgers' equation) using both quasi- and exact-inverse TLMs. It is demonstrated that our method performs the 4DVAR in much fewer iterations and CPU time compared with the conventional method, which requires running an adjoint model and a minimization algorithm (LBFGS in this case). We will show that our method is equivalent to using the Newton algorithm without the need to compute the Hessian. We will also discuss the possibility of parallel computation of this fast 4DVAR scheme to generate ensemble set of optimal initial conditions.

References

Kalnay, E., and Z.-X. Pu, 1998: Application of the quasi-inverse method to accelerate 4-D VAR. Preprints, *12th Conf. on Numerical Weather Prediction*, 11–16 January 1998, Phoenix, AZ, Amer. Meteor. Soc., 41–42.

Wang, Z., K. K. Droegemeier, L. White, and I. M. Navon, 1997: Application of a new adjoint Newton algorithm to the 3D ARPS storm-scale model using simulated data. *Mon. Wea. Rev.*, 125, 2460–2478.

Automatic Differentiation in the Numerical Solution and Sensitivity Analysis of Differential Algebraic Equations

Linda Petzold
petzold@chameleon.ucsb.edu
Computational Science and Engineering Group, University of California
Santa Barbara, California, U.S.A.

Douglas Clancey
clancey@cs.umn.edu
Department of Computer Science, University of Minnesota
Minneapolis, Minnesota, U.S.A.

In this talk we explore the use of automatic differentiation (AD) in the numerical solution and sensitivity analysis of differential-algebraic equations (DAEs). A new interface is introduced that enables the seamless use of DASPKSO with code generated by the AD software package Adifor2.0. Results in accuracy and efficiency are reported for several application problems.

Optimal Shape Design and Automatic Differentiation

Olivier Pironneau
Olivier.Pironneau@inria.fr
Laboratoire d'Analyse Numerique, University of Paris
Paris, France

Automatic differentiation (AD) is ideal for optimal shape design problems because the analytical derivation of derivatives for these problems is very difficult. Indeed, the variables are the shape of the domain of partial differential equations, i.e. the node position for their parametric representation.

In this talk a survey of applications with partial of full results will be given for

- Drag reduction
- Brake water optimization
- Optimal shapes in electromagnetism
- Wing design

Then one example will be given for the derivation of derivatives. A few implementations using semi-automatic computation of derivatives will then be shown with and without mesh adaption. Finally, we will discuss the problem of the choice of the norm and scalar product in the optimization algorithm.

AD Aspects of Solving DAEs by Taylor Series

John D. Pryce
J.D.Pryce@rmcs.cranfield.ac.uk
Computer Information Systems Engineering Dept, Royal Military College of Science
Shrivenham, Swindon, United Kingdom

We present a method of solving smooth DAEs by expanding the solution as a Taylor series. The method involves a pre-analysis stage where one solves an *assignment problem* (AP) as defined, for example, in Bertsekas's *Linear Network Analysis* (1991). The resulting structural data is essentially equivalent to that produced by the algorithm of Pantelides (1988) but easier to reason about. It depends only on the sparsity structure of the system, and in particular it computes the *structural index.*

If the method succeeds in computing a series, the point of expansion is necessarily consistent for the DAE, the series locally converges to the unique solution through that point, and the computed index equals the *uniform index.* The way in which the method determines consistent initial values and thereafter keeps the underlying constraints satisfied is similar to the index reduction method described in Mattsson and Söderlind (1993).

Despite this relation with existing techniques the method has something new to offer:

1. The resulting series generator can be used to convert software for validated ODE solution into software for *validated DAE solution.*
2. We believe the method, with its built-in symbolic understanding of the DAE system, has the potential to compete with traditional numerical DAE methods on appropriate problems.

The talk addresses the appropriate (AD and other) technology needed to realize this potential.

Shape Optimization of a Multi-Element Airfoil Using AD

Thomas Slawig
slawig@math.tu-berlin.de
Department of Mathematics, Technische Universität Berlin MA 6-2
Strasse des 17, Juni 136
D-10623 Berlin, Germany

We study the dependency of drag and lift of a four-element airfoil configuration with respect to the position and/or angle of some of the elements. A 2-D compressible Navier-Stokes or Euler computation using explicit or implicit method is used. The Jacobian in the nonlinear solver is computed with automatic differentiation. Moreover, we perform an optimization of drag and lift with respect to the design parameters position and angle. We use a BFGS quasi-Newton method. The gradient information is again supplied by automatic differentiation.

Structured Automatic Differentiation for Numerical Integration Methods

Arun Verma
verma@cs.cornell.edu
http://www.cs.cornell.edu/home/verma/AD/research.html
Computer Science Department, Cornell University
Ithaca, New York, U.S.A.

We present a positive result about using AD for numerical integration schemes.

A lot of computations dealing with continuous physical models are actually carried out numerically on a grid by means of a finite difference or finite element method. These numerical schemes compute only a computational approximation to the continuous function. We attempt to answer a fundamental question in this regard, namely, how do the derivatives of such approximate computer models (as they are computed with AD) relate to the true derivatives of the mathematical function?

We show that the "direct" derivatives (with AD) of the computer models share nice properties of the model the compute function itself (e.g., stability and convergence), while the "adjoint" derivatives in general do not have these properties.

We also present a novel scheme to "exploit structure" in the continuous problems defined by differential equations. The scheme deals with the actual numerical scheme only abstractly, thus generalizing the notion of structure by Coleman and Verma to handle abstract "processes". We present a few examples.

Adjoining Discretizations of Burgers' Equation

Andrea Walther and Andreas Griewank
[awalther,griewank]@math.tu-dresden.de
Institute of Scientific Computing
Technical University Dresden
Germany

In control theory and for the identification of parameters, adjoint or costate equations play an important role. They can be derived from the direct equation in a conditions setting and subsequently discretized. Alternatively, one may form the adjoint of the discretized direct equation in the sense of reverse automatic differentiation.

We study the relationship between both approaches for various discretizations of inviscid and viscous Burgers' equation.

In the inviscid case, three conservative schemes based on the fluxes due to Lax-Friedrichs, Godunov, and Roe are considered. Whereas the Lax-Friedrichs scheme turns out to be self-adjoint, the other two leave adjoints that are stable and consistent with the same order but do not represent conventional discretizations of the costate equations. These theoretical results are configured by our numerical experiments. In the singularly perturbed case, we also observe convergence for two discretization schemes, even though it does not appear to be theoretically guaranteed. Since the time integrations were performed up to 5000 steps, the forward trajectory cannot be saved for the subsequent reverse integration. Instead, we use our checkpointing software treeverse, which dramatically reduces the memory requirement, at a marginal increase in the operation count.

Web site: `http://www.math.tu-dresden.de/wir/institut.html`

Software:
ADOL-C: `http://www.math.tu-dresden.de/wir/institut.html`
treeverse: `ftp://ftp.math.tu-dresden.de/pub/TREEVERSE`