Sensitivity Analysis of Differential-Algebraic Equations and Partial Differential Equations

L. Petzold, Y. Cao, S. Li, R. Serban

August 10, 2005

LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

Sensitivity Analysis of Differential-Algebraic Equations and Partial Differential Equations

Linda Petzold and Yang Cao
Department of Computer Science
University of California
Santa Barbara, California 93106

Shengtai Li
Theoretical Division
Los Alamos National Laboratory
Los Alamos, New Mexico 87545

Radu Serban
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
Livermore,California 94551

Abstract— **Sensitivity analysis generates essential information for model development, design optimization, parameter estimation, optimal control, model reduction and experimental design. In this paper we describe the forward and adjoint methods for sensitivity analysis, and outline some of our recent work on theory, algorithms and software for sensitivity analysis of differential-algebraic equation (DAE) and time-dependent partial differential equation (PDE) systems.**

I. INTRODUCTION

In recent years there has been a growing interest in sensitivity analysis for large-scale systems governed by both differential algebraic equations (DAEs) and partial differential equations (PDEs). The results of sensitivity analysis have wide-ranging applications in science and engineering, including model development, optimization, parameter estimation, model simplification, data assimilation, optimal control, uncertainty analysis and experimental design.

Recent work on methods and software for sensitivity analysis of DAE and PDE systems has demonstrated that forward sensitivities can be computed reliably and efficiently. However, for problems which require the sensitivities with respect to a large number of parameters, the forward sensitivity approach is intractable and the adjoint (backward) method is advantageous. Unfortunately, the adjoint problem is quite a bit more complicated both to pose and to solve. Our goal for both DAE and PDE systems has been the development of methods and software in which generation and solution of the adjoint sensitivity system are transparent to the user. This has been largely achieved for DAE systems. We have proposed a solution to this problem for PDE systems solved with adaptive mesh refinement.

This paper has three parts. In the first part we introduce the basic concepts of sensitivity analysis, including the forward and the adjoint method. In the second part we outline the basic problem of sensitivity analysis for DAE systems and examine the recent results on numerical methods and software for DAE sensitivity analysis based on the forward and adjoint methods. The third part of the paper deals with sensitivity analysis for time-dependent PDE systems solved by adaptive mesh refinement (AMR).

II. BASICS OF SENSITIVITY ANALYSIS

Generally speaking, sensitivity analysis calculates the rates of change in the output variables of a system which result from small perturbations in the problem parameters. To illustrate the basic ideas of sensitivity analysis, consider a general, parameter-dependent nonlinear system

$$F(x,p) = 0, \tag{1}$$

where $x \in \mathbb{R}^n$, $p \in \mathbb{R}^m$, $F : \mathbb{R}^{m+n} \to \mathbb{R}^n$, and $\frac{\partial F}{\partial x}$ is nonsingular for all $p \in \mathbb{R}^m$. In its most basic form, sensitivity analysis calculates the sensitivities $\frac{dx}{dp}$ of the solution variables with respect to perturbations in the parameters. In many applications, one is concerned with a function of the state variable x and parameters p, which is called a *derived function*, given by $g(x,p) : \mathbb{R}^{m+n} \to \mathbb{R}^k$, where k usually is much smaller than m and n. In this case, the objective of sensitivity analysis is to compute the sensitivities $\frac{dg}{dp}$. We have

$$\frac{dg}{dp} = g_x x_p + g_p, \tag{2}$$

where $g_x = \frac{\partial g}{\partial x}$, $x_p = \frac{\partial x}{\partial p}$, $g_p = \frac{\partial g}{\partial p}$. There are two methods for obtaining the sensitivity: the forward method and the adjoint (backward) method.

1. *Forward method*

Linearizing the nonlinear system (1), we obtain

$$F_x x_p + F_p = 0, \tag{3}$$

where x_p is an $n \times m$ matrix. To compute $\frac{dg}{dp}$, we first need to solve m linear systems of (3) to obtain x_p. This is an excellent method when the dimension m of parameters is small.

2. *Backward method*

When the dimension m of p is large, the forward method becomes computationally expensive due to the need to compute x_p. We can avoid solving for x_p by using the backward (adjoint) method. To do this, we first multiply (3) by λ to obtain

$$\lambda^T F_x x_p + \lambda^T F_p = 0. \tag{4}$$

Now let λ solve the linear adjoint system

$$\lambda^T F_x = g_x. \tag{5}$$

Then $g_x x_p = -\lambda^T F_p$, thus from (2)

$$\frac{dg}{dp} = -\lambda^T F_p + g_p. \tag{6}$$

Note that we need to solve the linear equation (5) just once, no matter how many parameters are involved in the system. The adjoint method is a powerful tool for sensitivity analysis. Its advantage is that when the dimension of parameters is large, we need not solve the large system for x_p, but instead only the adjoint system (5), which greatly reduces the computation time. Thus while forward sensitivity analysis is best suited to the situation of finding the sensitivities of a potentially large number of solution variables with respect to a small number of parameters, adjoint (backward) sensitivity analysis is best suited to the complementary situation of finding the sensitivity of a scalar (or small-dimensional) function of the solution with respect to a large number of parameters.

The derivation of the adjoint sensitivity method for DAEs and PDEs follows a similar idea as above but is more complicated. In this paper we will avoid the details and focus on the introduction of the corresponding adjoint systems and software.

III. Sensitivity Analysis for DAE Systems

Recent work on methods and software for sensitivity analysis of DAE systems [13], [28], [25], [26], [29] has demonstrated that forward sensitivities can be computed reliably and efficiently via automatic differentiation [6] in combination with DAE solution techniques designed to exploit the structure of the sensitivity system. For a DAE depending on parameters,

$$\begin{cases} F(x, \dot{x}, t, p) &= 0 \\ x(0) &= x_0(p), \end{cases} \tag{7}$$

these problems take the form: find dx/dp_j at time T, for $j = 1, \ldots, n_p$. Their solution requires the simultaneous solution of the original DAE system with the n_p sensitivity systems obtained by differentiating the original DAE with respect to each parameter in turn. For large systems this may look like a lot of work but it can be done efficiently, if n_p is relatively small, by exploiting the fact that the sensitivity systems are linear and all share the same Jacobian matrices with the original system.

A. The Adjoint DAE

Some problems require the sensitivities with respect to a large number of parameters. For these problems, particularly if the number of state variables is also large, the forward sensitivity approach is intractable. These problems can often be handled more efficiently by the adjoint method [12]. In this approach, we are interested in calculating the sensitivity of an objective function

$$G(x, p) = \int_0^T g(x, t, p) dt, \tag{8}$$

or alternatively the sensitivity of a function $g(x, T, p)$ defined only at time T. The function g must be smooth enough that g_p and g_x exist and are bounded.

In [11] we derived the adjoint sensitivity system for DAEs of index [3] up to two (Hessenberg) and investigated some

of its fundamental properties. Here we summarize the main results.

The adjoint system for the DAE

$$F(t, x, \dot{x}, p) = 0$$

with respect to the derived function $G(x, p)$ (8) is given by

$$(\lambda^* F_{\dot{x}})' - \lambda^* F_x = -g_x, \tag{9}$$

where $*$ denotes the transpose operator and prime denotes the total derivative with respect to t.

The adjoint system is solved backwards in time. For index-0 and index-1 DAE systems, the initial conditions for (9) are taken to be $\lambda^* F_{\dot{x}}|_{t=T} = 0$, and the sensitivities of $G(x, p)$ with respect to the parameters p are given by

$$\frac{dG}{dp} = \int_0^T (g_p - \lambda^* F_p) \, dt + (\lambda^* F_{\dot{x}})|_{t=0}(x_0)_p. \tag{10}$$

For Hessenberg index-2 DAE systems, the initial conditions are more complicated, and are described in detail along with an algorithm for their computation in [11].

For a scalar derived function $g(x, T, p)$, the corresponding adjoint DAE system is given by

$$(\lambda_T^* F_{\dot{x}})' - \lambda_T^* F_x = 0, \tag{11}$$

where λ_T denotes $\frac{\partial \lambda}{\partial T}$. For index-0 and index-1 DAE systems, the initial conditions $\lambda_T(T)$ for (11) satisfy $(\lambda_T^* F_{\dot{x}})|_{t=T} = [g_x - \lambda^* F_x]|_{t=T}$. We note that the initial condition $\lambda_T(T)$ is derived in such a way that the computation of $\lambda(t)$ can be avoided. This is the case also for index-2 DAE systems. The full algorithm for consistent initialization of the adjoint DAE system is given in [11]. The sensitivities of $g(x, T, p)$ with respect to the parameters p are given for index-0 and index-1 DAE systems by

$$\frac{dg}{dp} = (g_p - \lambda^* F_p)|_{t=T} - \int_0^T (\lambda_T^* F_p) + (\lambda_T^* F_{\dot{x}})|_{t=0}(x_0)_p. \tag{12}$$

Note that the values of both λ at $t = T$ and λ_T at $t = 0$ are required in (12). If $F_p \neq 0$, the transient value of λ_T is also needed. For an index-2 system, if the index-2 constraints depend on p explicitly, an additional term must be added to the sensitivity (12).

If the objective function is of the integral form $G(x, p)$ (8), it can be computed easily by adding a *quadrature variable*, which is equal to the value of the objective function, to the original DAE. For example, if the number of variables in the original DAEs is N, we append a variable x_{N+1} and equation

$$\dot{x}_{N+1} = g(x, t, p).$$

Then $G = x_{N+1}(x, T, p)$. In this way, we can transform any objective function in the integral form (8) into the scalar form $g(x, T, p)$. The quadrature variables can be calculated very efficiently [25] by a staggered method; they do not enter into the Jacobian matrix.

From [11] we know that for DAE systems of index up to two (Hessenberg), asymptotic numerical stability in solving the

forward problem is preserved by the backward Euler method, but only (for fully-implicit DAE systems) if the discretization of the time derivative is performed 'conservatively', which corresponds to solving an *augmented adjoint* DAE system,

$$\dot{\bar{\lambda}} - F_x^* \lambda = 0,$$
$$\bar{\lambda} - F_{\dot{x}}^* \lambda = 0. \qquad (13)$$

It was shown in [11] that the system (13) with respect to $\bar{\lambda}$ preserves the stability of the original system. Note that the augmented system (13) is of (one) higher index than the original adjoint system (11). This is not a problem in the implementation since the newly high-index variables do not enter into the error estimate and it can be shown that basic DAE structures such as combinations of semi-explicit index-1 and Hessenberg index-2 are preserved under the augmentation. Also, the linear algebra is accomplished in such a way that the matrix needed is the transpose of that required for the original system. Thus there are no additional conditioning problems for the linear algebra due to the use of the augmented adjoint system.

B. Sensitivity Analysis Tools

The DASPK3.0 [25], [26] software package was developed for forward sensitivity analysis of DAE systems of index up to two [8], [3], and has been used in sensitivity analysis and design optimization of several large-scale engineering problems [21], [32]. DASPK3.0 is an extension of the DASPK software [9], [8] developed by Brown, Hindmarsh, and Petzold for the solution of large-scale DAE systems. DASPKADJOINT, described in detail in [24], is an extension to DASPK3.0 which accomplishes the DAE solution along with adjoint sensitivity analysis.

Functionality similar to that of DASPK3.0-DASPKADJOINT is provided by the sensitivity-enabled solvers in SUNDIALS, the Suite of Nonlinear and Differential/Algebraic Equation Solvers [19]. CVODES [34] is an ODE integrator with sensitivity analysis capabilities (both forward and adjoint), while IDAS[1] provides the same capabilities for DAE systems. While the DASPK family of solvers is written in Fortran, the SUNDIALS solvers are written in ANSI-C.

Much of the challenge in writing adjoint solvers for DAE systems stems from the requirement of handling the complexities of formulation and solution of the adjoint sensitivity system while requiring as little additional information from the user as is mathematically necessary. Here we describe some of the details of the implementation.

In the adjoint system (11) and the sensitivity calculation (12), the derivatives F_x, $F_{\dot{x}}$ and F_p may depend on the state variables x, which are the solutions of the original DAEs. Ideally, the adjoint DAE (11) should be coupled with the original DAE and solved together as we did in the forward sensitivity method. However, in general it is not feasible to solve them together because the original DAE may be unstable

---

when solved backward. Alternatively, it would be extremely inefficient to solve the original DAE forward any time we need the values of the state variables.

The implementation of the adjoint sensitivity method consists of three major steps. First, we must solve the original ODE/DAE forward to a specific output time T. Second, at time T, we compute the consistent initial conditions for the adjoint system. The consistent initial conditions must satisfy the boundary conditions of (9). Finally, we solve the adjoint system backward to the start point, and calculate the sensitivities.

With enough memory, we can store all of the necessary information about the state variables at each time step during the forward integration and then use it to obtain the values of the state variables by interpolation during the backward integration of the adjoint DAEs. The memory requirements for this approach are proportional to the number of time steps and the dimension of the state variables x, and are unpredictable because the number of time steps varies with different options and error tolerances of the ODE/DAE solver.

To reduce the memory requirements and also make them predictable, we use a two-level checkpointing technique. First we set up a checkpoint after every fixed number of time steps during the forward integration of the original DAE. Then we recompute the forward information between two consecutive checkpoints during the backward integration by starting the forward integration from the checkpoint. This approach needs to store only the forward information at the checkpoints and at a fixed number of times between two checkpoints.

In the implementation we allocated a special buffer to communicate between the forward and backward integration. The buffer is used for two purposes: to store the necessary information to restart the forward integration at the checkpoints, and to store the state variables and derivatives at each time step between two checkpoints for reconstruction of the state variable solutions during the backward integration.

In order to obtain the fixed number of time steps between two consecutive checkpoints, the second forward integration should make exactly the same adaptive decisions as the first pass if it restarts from the checkpoint. Therefore, the information saved at each checkpoint should be enough that the integration can repeat itself. In the case of DASPK3.0, CVODES, and IDAS, the necessary information includes the order and stepsize for the next time step, the coefficients of the BDF formula, the history information array of the previous k time steps, the Jacobian information at the current time, etc.. To avoid storing Jacobian data (which is much larger than other information) in the buffer, we enforce a reevaluation of the iteration matrix at each checkpoint during the first forward integration.

If the size of the buffer is specified, the maximum number of time steps allowed between two consecutive checkpoints and the maximum number of checkpoints allowed in the buffer can be easily determined. However, the total number of checkpoints is problem-dependent and unpredictable. It is possible that the number of checkpoints is also too large for

some applications to be held in the buffer. We then write the data of the checkpoints from the buffer to a disk file and reuse the buffer again. Whenever we need the information on the disk file, we can access it from the disk. We assume that the disk is always large enough to hold the required information.

Various interpolation schemes can then be used to recover the forward solution at any time (between two consecutive check points) to be used in the generation of the adjoint system. For example, we can store x and $\dot{x}$ at each time step during the forward integration and reconstruct the solution at any time by cubic Hermite interpolation. This option is available for any of the three solvers mentioned above. The SUNDIALS solvers also provide an alternative interpolation scheme, based on storing only x values and then constructing a variable-order interpolation which attempts to approximate the polynomial underlying the linear multistep integration formula[2]. This second option is better suited than cubic Hermite interpolation especially for the Adams integration method (an option in CVODES) for which the method order can be as high as 12.

Another important issue is how to formulate the adjoint DAE and the initial conditions so that the user doesn't have to learn all about the adjoint method and derive these for themselves. The adjoint equations involve matrix-vector products from the left side (vector-matrix products). Although a matrix-vector product $F_x v$ can be approximated via a directional derivative finite difference method, it is difficult to evaluate the vector-matrix product $v F_x$ directly via a finite difference method. The vector-matrix product $v F_x$ can be written as a gradient of the function $v F(x)$ with respect to x. However, N evaluations of $v F(x)$ are required to calculate the gradient by a finite-difference method if we don't assume any sparsity in the Jacobian. Therefore, automatic differentiation (AD) is necessary to improve the computational efficiency. A forward mode AD tool cannot compute the vector-matrix products without evaluation of the full Jacobian. It has been shown [15] that an AD tool with reverse mode can evaluate the vector-Jacobian product as efficiently as a forward mode AD tool can evaluate the Jacobian-vector product. In our implementation with DASPK3.0, we use the AD tool TAMC [15] to calculate the vector-matrix products, while CVODES and IDAS are currently being interfaced to the AD tool ADIC [7]. Initialization of the adjoint DAE is quite a bit more complicated than in the ODE case. For details, see [10] and [11]. In general, one needs to be able to provide some information about the structure of the problem (i.e. which are the index-1 and index-2 variables and constraints).

IV. SENSITIVITY ANALYSIS FOR TIME-DEPENDENT PDE SYSTEMS

Sensitivity methods for steady-state PDE problems have been studied by many authors (see [1], [5], [16], [18], [20]). Here we outline some recent results on adjoint methods for

---

[2]Note that, due to the *fixed-leading coefficient* implementation in CVODES and IDAS, one does not exactly recover the BDF interpolant this way, but rather a polynomial of the same local order.

general transient PDE systems. Although many of the results from the steady-state system can be readily extended to the time-dependent PDE system, the time-dependent system has some unique features that must be treated differently. For example, apart from the boundary conditions, we now have initial conditions that must be determined. Two important classical fields that make extensive use of sensitivity analysis are inverse heat-conduction problems [14] and shape design in aerodynamic optimization [31].

Given a parameter-dependent PDE system

$$F(t, u, u_t, u_x, u_{xx}, p) = 0, \qquad (14)$$

and a vector of objective functions $G(x, u, p)$ that depend on u and p, the sensitivity problem usually takes the form: find $\frac{dG}{dp}$, where p is a vector of parameters. By the chain rule, the sensitivity $\frac{dG}{dp}$ is given by

$$\frac{dG}{dp} = \frac{\partial G}{\partial u}\frac{\partial u}{\partial p} + \frac{\partial G}{\partial p}. \qquad (15)$$

If we treat (14) as a nonlinear system about u and p, say $H(u, p) = F(t, u, u_t, u_x, u_{xx}, p)$, we have the following relationship

$$\frac{\partial H}{\partial u}\frac{\partial u}{\partial p} + \frac{\partial H}{\partial p} = 0.$$

Assuming that $\frac{\partial H}{\partial u}$ is boundedly invertible, the sensitivity $\frac{dG}{dp}$ is given by

$$\frac{dG}{dp} = -\frac{\partial G}{\partial u}\left(\frac{\partial H}{\partial u}\right)^{-1}\frac{\partial H}{\partial p} + \frac{\partial G}{\partial p}. \qquad (16)$$

There are two basic methods to calculate $\frac{dG}{dp}$ in (16): forward and adjoint. The forward methods calculate $\frac{\partial u}{\partial p} = \left(\frac{\partial H}{\partial u}\right)^{-1}\frac{\partial H}{\partial p}$ first, which is the solution of the sensitivity PDE for each uncertain parameter. The adjoint methods, however, compute $\frac{\partial G}{\partial u}\left(\frac{\partial H}{\partial u}\right)^{-1}$ first, which is the solution of the adjoint PDE. The sensitivity and adjoint PDEs will be defined later. Although both methods yield the same analytical sensitivities, the computational efficiency may be quite different, depending on the number of objective functions (dimension of G) and the number of sensitivity parameters (dimension of p). The forward method is attractive when there are relatively few parameters or a large number of objective functions, while the adjoint method is more efficient for problems involving a large number of sensitivity parameters and few objective functions. We have studied the forward method in [27] and shown how it is possible to make use of the methods and software for sensitivity analysis of DAEs in combination with an adaptive mesh refinement algorithm for PDEs. In [23], we have studied extensively the adjoint method for PDEs solved with adaptive mesh refinement. Those results are outlined in what follows.

Two approaches can be taken for each method. In the first, called the *discrete* approach, we approximate the PDE by a discrete nonlinear system and then differentiate the discrete system with respect to the parameters. The discrete approach is easy to implement with the help of *automatic differentiation* tools [6], [15]. However, when the mesh is

solution or parameter dependent (e.g., for an *adaptive* mesh or moving boundary), or a nonlinear discretization scheme (e.g., upwinding) is used, the discrete approach may not be computationally effective.

It is well-known that the method of lines (MOL) can transform a PDE system into an ODE or DAE system by spatial discretization. Thus the sensitivity calculation methods in [11] can be used if the semi-discretized PDE is obtained. However, we have observed that the *adjoint of the discretization* (AD) may not be consistent with a PDE, and the adjoint variables are not smooth on an adaptive grid. Therefore, if the adaptive region is changing with time (e.g. in adaptive mesh refinement (AMR) [4], [27]), the interpolation for the adjoint variables between different grids will introduce large errors. The AD method cannot be used in this case. On the other hand, one can show [23] that for linear discretization methods applied on a fixed grid with appropriate treatment of the boundary conditions, the sensitivities generated are accurate except in a small boundary layer.

In the second, called the *continuous* approach, we differentiate the PDE with respect to the parameters first and then discretize the sensitivity or adjoint PDEs to compute the approximate sensitivities. The system resulting from the continuous approach is usually much simpler than that from the discrete approach, and is naturally consistent with the adjoint PDE system. Therefore, the adaptive grid method and interpolation can be used without difficulties. Derivation of the adjoint PDE could be handled by symbolic methods such as MAPLE. However it is very difficult to formulate proper boundary conditions for the adjoint of a general PDE system, and to the best of our knowledge an algorithm for generating the boundary conditions does not exist for a general PDE system. Moreover, the adjoint system may become inadmissible for some objective functionals (see [1], [2]), where the boundary conditions (or initial conditions) for the adjoint PDE system cannot be formulated properly. The discrete approach does not have such difficulties.

We propose an approach to combine the AD method and the *discretization of the adjoint* (DA) method in an efficient manner so that it can be used with AMR. The new approach (called the ADDA method) not only solves the problem for AD on the adaptive grid, it also solves the inadmissibility problem for DA. Both the AD and DA methods are used in this new approach but are applied in different regions.

We developed the ADDA method based on an observation that the discretization from the AD method is consistent with the adjoint PDE (hence it can be replaced with the discretization of the DA method) at the internal points if the mesh and the (linear) discretization are uniform everywhere except at the boundaries. The basic idea of the ADDAmethod is illustrated in Fig. 1.

The results of the ADDA method should be equal or close to the results of the AD method on a nonadaptive fine grid. Given a reference nonadaptive fine grid, we first split the whole domain into two zones: boundary buffer zone and internal zone (see Fig. 1). The boundary buffer zone consists
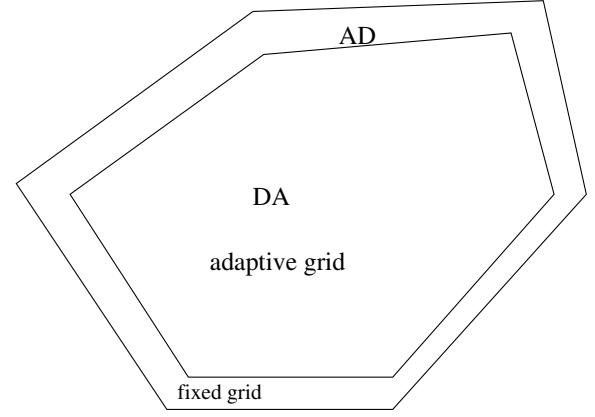

Fig. 1. Diagram of the ADDA method

of the boundary points and points that use the boundary points in their discretization. The remainder of the points belong to the internal zone. Since the discretization of the AD method may not be consistent with the adjoint PDE at or near the boundaries, the buffer zone is fixed and never adapted during the entire time integration. In the internal zone, the discretization from the AD method can be replaced with the discretization from the DA method if we assume that the discretization from the AD method is consistent with the adjoint PDE. It turns out [23] that this assumption is not always true for a general discretization and grid. However, if the mesh and discretization of the forward problem are uniform in the internal zone, the adjoint of the discretization is indeed consistent with the adjoint PDE.

After the discretization in the internal zone has been replaced by that from the DA method, the mesh can be adapted to achieve efficiency without loss of the accuracy. The adaptive mesh refinement in the internal zone is invisible to the AD method, which expects that the discretizations for the adjoint system are generated by the AD method on a nonadaptive fine grid. Instead the discretization is accomplished efficiently by the DA method on an adaptive grid. The internal zone looks like a black box to the AD method.

Since the sensitivity calculation is based on the AD method, the initial conditions for the adjoint system must be generated by the AD method. However, the initial conditions generated by the AD method may involve the grid spacing information, due to the objective functional evaluation [23]. A variable transformation [23] is used to eliminate the grid spacing information related to the integration scheme in the objective function evaluation. However, it cannot eliminate the grid spacing information related to the integrand function.

Strictly speaking, the values of the adjoint variables are different on different grids. That is why the sensitivity calculation by the AD method must be performed on a fixed mesh. The initial given mesh, which is the last mesh generated at $t = T$ in the forward adaptive method, may not be the same as the reference nonadaptive fine mesh we seek. Therefore, we must

calculate the initial conditions for the ADDA method on the reference mesh first and then project them onto the initial given mesh by interpolation.

The overall algorithm of the ADDA method is as follows: First we obtain the initial conditions for the adjoint system by the AD method on a virtual nonadaptive fine grid. Then we transform and project them to the adaptive grid with a fixed boundary buffer zone. We assume that the discretization has been chosen so that AD is consistent with the adjoint PDE internally. Then the spatial discretization in the boundary buffer zone is generated by the AD method via automatic differentiation, and the discretization in the internal zone is defined by discretization of the adjoint PDE. Finally, an ODE or DAE time solver is used to advance the solution to the next time step. After the adjoint variables have been computed, the sensitivity evaluations of the AD method are used to calculate the sensitivities.

Examples that demonstrate the effectiveness of the ADDA method are presented in [23].

REFERENCES

[1] W. K. Anderson and V. Venkatakrishnan, "Aerodynamic design optimization on unstructured grid with a continuous adjoint formulation", *AIAA 97-0643, 35'th Aerospace Science meeting & exhibit*, 1997.

[2] E. Arian and M. Salas, *Admitting the Inadmissible: Adjoint Formulation for Incomplete Cost Functionals in Aerodynamic Optimization*, ICASE Report No. 97-69, 1997.

[3] U. M. Ascher and L. R. Petzold, *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*, SIAM, 1998.

[4] M. J. Berger and J. Oliger, "Adaptive mesh refinement for hyperbolic partial differential equations", *J. Comput. Phys.* 53, pp. 484-512, 1984

[5] J. Borggaard and J. Burns, "A PDE sensitivity equation method for optimal aerodynamic design", *J. Comp. Phys.* 136, pp. 366-384, 1997.

[6] C. Bischof, A. Carle, G. Corliss, A. Griewank and P. Hovland, "ADIFOR - Generating derivative codes from Fortran programs", *Scientific Programming* 1, pp. 11-29, 1992.

[7] C. Bischof, L. Roh, and A. Mauer, *ADIC - An Extensible Automatic Differentiation Tool for ANSI-C*, Technical Report ANL/MCS-P626-1196, Mathematics and Computer Science Division, Argonne National Laboratory, 1996.

[8] K. E. Brenan, S. L. Campbell and L. R. Petzold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, Second edition, SIAM, 1996.

[9] P. N. Brown, A. C. Hindmarsh and L. R. Petzold, "Using Krylov methods in the solution of large-scale differential-algebraic systems", *SIAM J. Sci. Comput.* pp. 1467-1488, 1994.

[10] Y. Cao, S. Li, L. Petzold and R. Serban, "Adjoint Sensitivity Analysis for Differential-Algebraic Equations: The Adjoint DAE System and its Numerical Solution", *SIAM J. Sci. Comput.* 24(3), pp. 1076-1089, 2003.

[11] Y. Cao, S. Li and L. Petzold, "Adjoint sensitivity analysis for differential-algebraic equations: algorithms and software", *J. Comp. Appl. Math.* 149, pp. 171-192, 2002.

[12] R. M. Errico, "What is an adjoint model?", *Bulletin of the American Meteorological Society* 78, pp. 2577-2591, 1997.

[13] W. F. Feehery, J. E. Tolsma and P. I. Barton, "Efficient sensitivity analysis of large-scale differential-algebraic systems", *Applied Numerical Mathematics* 25, pp. 41-54, 1997.

[14] Y. Jarny, M.N. Ozisik and J.P. Barton, "A general optimization method using adjoint equation for solving multidimensional inverse heat conduction", *Int. J. Heat Mass. Transfer* 34, pp. 2911-2919, 1991.

[15] R. Giering and T. Kaminski, "Recipes for adjoint code construction", *ACM Trans. Math. Software* 24, pp. 437-474, 1998.

[16] M. B. Giles and N.A. Pierce, "Adjoint equations in CFD: duality, boundary conditions and solution behaviour", *AIAA Paper* 97-1850, 1997.

[17] M. B. Giles and N.A. Pierce, "An introduction to the adjoint approach to design", *ERCOFTAC Workshop on Adjoint Methods*, Toulouse, June 21-23, 1999.

[18] O. Ghattas and J.-H. Bark, "Optimal control of two and three-dimensional incompressible Navier-Stokes Flows", *J. Comp. Phys.*, 136 pp. 231-244, 1997.

[19] A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward, "SUNDIALS, Suite of Nonlinear and Differential/Algebraic Equation Solvers," *ACM Trans. Math. Software*, accepted, 2004.

[20] A. Jameson, "Aerodynamic Design via Control Theory", *J. of Scientific Computing*, 3, pp. 233-260, 1988.

[21] D. Knapp, V. Barocas, K. Yoo, L. Petzold and R. Tranquillo, "Rheology of reconstituted type I collagen gel in confined compression", *J. Rheology* 41, pp. 971-993, 1997.

[22] R. M. Lewis, "Numerical computation of sensitivities and the adjoint approach", in *Computational Methods for Optimal Design and Control*, J. Borggaard, J. Burns, E. Cliff and S. Schreck Eds., pp. 285-302, Birkhauser, 1998.

[23] S. Li and L. Petzold, "Adjoint Sensitivity Analysis for Time-Dependent Partial Differential Equations with Adaptive Mesh Refinement", *Journal of Computational Physics* 198(1), pp. 310-325, 2004.

[24] S. Li and L. Petzold, *Description of DASPKADJOINT: An Adjoint Sensitivity Solver for Differential-Algebraic Equations*, UCSB Technical Report 2002, www.engineering.ucsb.edu/ cse.

[25] S. Li and L. Petzold, "Software and algorithms for sensitivity analysis of large-scale differential-algebraic systems", *J. Comp. Appl. Math.* 125, pp. 131-145, 2000.

[26] S. Li and L. Petzold, *Design of New DASPK for Sensitivity Analysis*, Technical Report, Dept. of Computer Science, Technical Report UCSB, 1999.

[27] S. Li, L. Petzold and J. Hyman, "Solution adapted mesh refinement and sensitivity analysis for parabolic partial differential equations", *Lecture Notes in Computational Science and Engineering* 30, Ed. L. T. Biegler, O. Ghattas, M. Heinkenschloss and B. van Bloeman Waanders, Springer-Verlag, Heidelberg, 2003.

[28] S. Li, L. Petzold and W. Zhu, "Sensitivity analysis of differential-algebraic equations: A comparison of methods on a special problem", *Applied Numerical Mathematics* 32, pp. 161-174, 2000.

[29] T. Maly and L. R. Petzold, "Numerical methods and software for sensitivity analysis of differential-algebraic systems", *Applied Numerical Mathematics* 20, pp. 57-79, 1997.

[30] G. I. Marchuk, V. I. Agoshkov and V. P. Shutyaev, *Adjoint equations and perturbation algorithms*, CRC Press, Boca Raton, Fl, 1996.

[31] S. K. Nadarajah and A. Jameson, "A comparison of the continuous and discrete adjoint approach to automatic aerodynamic optimization", *AIAA paper* 00-0067, 2000.

[32] L. L. Raja, R. J. Kee, R. Serban and L. Petzold, "Dynamic optimization of chemically reacting stagnation flows", *Proc. Electrochemical Society*, 1998.

[33] A. Sei and W. W. Symes, *A note on consistency and adjointness for numerical schemes*, Tech. Report TR95-04, Department of Computational and Applied Math., Rice University, 1995.

[34] R. Serban and A.C. Hindmarsh, "CVODES, The Sensitivity-Enabled ODE Solver in SUNDIALS," Proceedings of the *ASME International Design Engineering Technical Conferences*, Long Beach, CA, 24-28 September 2005.