Available online at www.sciencedirect.com

SCIENCE DIRECT®

Future Generation Computer Systems 21 (2005) 1345–1355

FGCS
FUTURE GENERATION COMPUTER SYSTEMS

www.elsevier.com/locate/fgcs

Generating efficient derivative code with TAF Adjoint and tangent linear Euler flow around an airfoil

R. Giering[a], T. Kaminski[a,*], T. Slawig[b]

[a] FastOpt, Schanzenstr. 36, 20357 Hamburg, Germany
[b] Technische Universität Berlin, Straße des 17. Juni 136, D-10623 Berlin, Germany

Available online 18 December 2004

Abstract

FastOpt's new automatic differentiation tool TAF is applied to the two-dimensional Navier–Stokes solver NSC2KE. For a configuration that simulates the Euler flow around an NACA airfoil, TAF has generated the tangent linear and adjoint models as well as the second derivative (Hessian) code. Owing to TAF's capability of generating efficient adjoints of iterative solvers, the derivative code has a high performance: running both the solver and its adjoint requires 3.4 times as long as running the solver only. Further examples of highly efficient tangent linear, adjoint, and Hessian codes for large and complex three-dimensional Fortran 77-90 climate models are listed. These examples suggest that the performance of the NSC2KE adjoint may well be generalised to more complex three-dimensional CFD codes. We also sketch how TAF can improve the adjoint's performance by exploiting self-adjointness, which is a common feature of CFD codes.
© 2004 Elsevier B.V. All rights reserved.

Keywords: Computational fluid dynamics; Adjoint; Hessian; Automatic differentiation; Navier–Stokes; Shape optimisation

1. Introduction

Many applications in computational fluid dynamics (CFD) do benefit from availability of sensitivity information. Examples span the range from multidisciplinary shape design of airfoils or turbomachinery blades to modelling of atmosphere or ocean dynamics [1–3]. Sensitivities quantify the impact of a change in certain control variables on particular target quantities of interest. In aerodynamics or aeroacoustics applications, lift and drag or kinetic energy are examples of such target quantities, and the control variables define the shape of the object under consideration. In atmosphere and ocean modelling, typical target quantities are integrals of the large-scale circulation or the difference between observations and their model-simulated counterparts. Typical control variables are the initial state, boundary values, or parameters in the model formulation.

In some applications, the sensitivity information is interpreted directly. In others, this interpretation is done

* Corresponding author. Fax: +49 40 48096357.
E-mail addresses: ralf.giering@fastopt.com (R. Giering), thomas.kaminski@fastopt.com (T. Kaminski), slawig@math.tu-berlin.de (T. Slawig)

by an optimisation algorithm, which iteratively exploits sensitivity information to vary the control variables in order to improve the value of the target quantity. Second-order sensitivity (Hessian) information [4] is important to speed up this search process and to analyse robustness of the solution [5,6].

There are different strategies of deriving sensitivity information. A first approach, also called continuous approach, applies perturbation theory [7] to the model: the model equations are linearised, discretised and the tangent linear model is coded. For most problems, however, the desired number of control variables is much larger than that of the target quantities. For many of these problems, sensitivity calculation is only computationally feasible via an adjoint formulation of the model equations [8]. The adjoint equations are then discretised and coded, which yields the adjoint model.

An alternative strategy of obtaining sensitivity information applies automatic differentiation (AD) (see [9] and references therein) directly to the code of the model: To generate the derivative code (tangent linear or adjoint model), the model code is decomposed into elementary functions, which more or less correspond to the individual statements in the code. These elementary functions are differentiated (this derivative is also called local Jacobian). The derivative code multiplies these local Jacobians, which, according to the chain rule, yields the derivative of the composite function. As opposed to derivative approximation by divided differences (also known as numerical differentiation), AD provides sensitivity information that is accurate within round-off error.

Like the continuous approach, AD can construct both tangent linear and adjoint models. The tangent linear model uses the order, in which the model evaluates the statements, to evaluate the product of their Jacobians. The adjoint model does this evaluation in reverse order. In AD terminology, the tangent linear model operates in forward mode and the adjoint model operates in reverse mode. Similar to the finite difference approximation, the computational resources needed in forward mode increase with the number of control variables. In reverse mode, they are roughly proportional to the number of target quantities, but virtually independent of the number of control variables. The availability of the reverse mode is another major advantage of AD over the finite difference approximation.

Applying the continuous approach requires the choice of discretisation schemes for both the model equations and the adjoint equations. Typically, in the discretisation step, the adjoint relation is only valid in an approximate sense. Consequently, the sensitivity information that is provided by the adjoint code is not fully consistent with the actual sensitivity of the model code. This inconsistency, which does not occur when using AD, can be problematic in an optimisation context [10]. Also, for particular applications, the rigorous derivation of the adjoint equations that form the basis of the continuous adjoint approach can become a cumbersome piece of analysis [11,8]. For second derivatives, this analysis gets even more complex [4]. By contrast, using AD to construct the adjoint code avoids this analytical effort at all.

AD can be carried out by hand or by an AD tool (e.g. [12–17]; see also http://www.autodiff.org). Applying an AD tool restricts the effort of development and maintenance to the model itself: based on the model code, the adjoint and tangent linear models as well as the Hessian code can be generated and maintained automatically. Especially for models under development, this constitutes a significant advantage and calls for an AD tool as integral component of a state-of-the-art modelling system [18,19]. By contrast, the continuous approach requires coding and maintaining the derivative code by hand. For CFD codes written in Fortran 77, AD tools have been applied to generate many tangent linear codes (e.g. [20–25]) and few adjoint (e.g. [26,28]) and Hessian (e.g. [29]) codes.

This paper introduces the relatively new AD tool transformation of algorithms in Fortran (TAF, [16]), which handles Fortran 77-95 code. For almost a decade, TAF's predecessor TAMC ([17]) has been generating tangent linear and adjoint models, as well as Hessian code of an ever increasing number of, among others, large models of atmosphere and ocean dynamics. The performance of the derivative code generated by TAMC and TAF is very high, which is crucial for the feasibility of most applications.

Recently, the first TAMC and TAF applications to Navier–Stokes solvers for shape design and instantaneous control have been completed ([30–34]). Using an example of a two-dimensional Navier–Stokes solver (NSC2KE, [35]), this paper describes how TAF can be applied to generate highly efficient adjoint and tangent linear and Hessian code. As many other CFD codes,

Download English Version:

https://daneshyari.com/en/article/10330393

Download Persian Version:

https://daneshyari.com/article/10330393

Daneshyari.com