Adjoint Solver for Commercial Computational Fluid Dynamics Codes using Algorithmic Differentiation Tools

Zahrasadat Dastouri * and Uwe Naumann †

Software and Tools for Computational Engineering (STCE), RWTH Aachen University, 52074 Aachen, Germany

The main topic of this paper is the application of discrete adjoint method in large legacy Computational Fluid Dynamics (CFD) codes using Algorithmic Differentiation tools (AD). It provides an adjoint solver for generating accurate and detailed sensitivities for shape optimization problems in industrial CFD applications.

I. Introduction

Algorithmic Differentiation (AD)[1,2] employs the rules of differential calculus in an algorithmic manner to determine accurate derivatives of a function defined by computer programs. For a given implementation of the flow model $F : \mathbb{R}^n \to \mathbb{R}^m$ over a computational grid, the computer program is developed to simulate the functional dependence of one or more objectives $\mathbf{y} \in \mathbb{R}^m$ on a potentially large number of input variables $\mathbf{x} \in \mathbb{R}^n$ by simulation of, $F : \mathbb{R}^n \to \mathbb{R}^m, \mathbf{y} = F(\mathbf{x})$. AD enables us to compute the corresponding derivatives $\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$ in forward (forward mode) or backward (reverse mode). For a given implementation of the primal function F, the function $F_{(1)} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n \times \mathbb{R}^m$, defined as

$$(\mathbf{y}, \mathbf{x}_{(1)}) = F_{(1)}(\mathbf{x}, \mathbf{y}_{(1)}) \equiv \left(F(\mathbf{x}), \left(\frac{d\mathbf{y}}{d\mathbf{x}} \right)^T \cdot \mathbf{y}_{(1)} \right) \tag{1}$$

is referred to as the *adjoint* model of F. The adjoint model implementation yields the objective $\mathbf{y}$ and the product $\mathbf{x}_{(1)} \equiv \nabla F(\mathbf{x})^T \cdot \mathbf{y}_{(1)}$ of its transposed jacobian at the current point $\mathbf{x} \in \mathbb{R}^n$.

There are two main methods for implementing AD: by source code transformation (S-T) or by use of derived data types and operator overloading (O-O). In O-O AD the code segments and arguments of the primal code are stored inside a memory structure called tape during the forward run of the primal. In reverse mode the stored values on the tape are interpreted to get the resulting adjoints, while the S-T approach parses the code at compile time and generates the actual derivative code.

Prior to this paper[3],[4] we have discussed the implementation of the AD tool `dco/fortran`[a] *(Derivative Code by Overloading in Fortran)* to an unstructured pressure-based steady Navier-Stokes solver. The solver is an incompressible flow solver with cell-centered storage and face-based residual assembly[7] uses the SIMPLE[6] pressure correction algorithm. We have addressed proper solution algorithms adapted to the code for the improvement of efficiency of the adjoint code by optimizing the checkpointing scheme for the iterative solver and development of symbolically differentiated of linear solver. In addition, we investigated the benefit of the reverse accumulation technique[8] for the fixed point iterative construction in the primal code.

Moreover, in [5] we have discussed an hybrid approach by combining the flexibility and robustness of operator overloading with the efficiency of source transformation by coupling `dco/fortran` and `TAPENADE`.[9] The latter was used for the derivation of computationally expensive kernels. Our emphasis was to automate the implementation of an adjoint software to decrease the development time of the differentiated code. Computational efficiency is proved through demonstration examples.

In this paper, based on the experiences on applying AD on a medium size solver as a benchmark, reverse mode of AD has been applied in the context of shape optimisation for fluid dynamics systems to a complex commercially licensed CFD code, CFD-ACE+.[10] The multi-physics ACE+ solver has many features for flow modeling including laminar and turbulence flow under steady transient states. The functionalities enable decouple or coupled pressure based flow solvers simulations.

---

*PhD Candidate, Software and Tools for Computational Engineering (STCE), RWTH Aachen University, 52074 Aachen, Germany.

†Professor, Software and Tools for Computational Engineering (STCE), RWTH Aachen University, 52074 Aachen, Germany.

[a]developed at the institute a*Software and Tools for Computational Engineering* at RWTH Aachen University implementing AD by overloading in Fortran[1]

II. Development a the Discrete Adjoint Model for Commercial CFD CODE CFD-ACE+

Required elements for adjoint solver are utilized in CFD solution to get discrete adjoint version of the CFD code using Operator Overloading (O-O) and Source Transformation (S-T) AD tools. In CFD application, the resulting memory usage of the black-box adjoint approach is not acceptable for real world problems. Therefore, the need for efficient solution techniques arising from the PDE discretization of Navier-Stokes system has been emphasised and discussed for the problems considered such techniques. This includes symbolically differentiating the linear solver in the adjoint code and employing optimized checkpointing scheme and reverse accumulation for fixed point iteration (SIMPLE) loop. A robust and efficient adjoint solver is constructed applying O-O or S-T approaches and the combination of these tools for computationally expensive part of the solver with the objective of minimizing the development time to get the adjoint code. In both cases the utility of the approaches has been demonstrated by numerical experiments and relevant flow test cases to reflects and compare the performance assessment of the adjoint solver. A significant contribution of this work is to investigate the properties that the adjoint systems generated by AD tools posses in a complex legacy commercial solver compared to the original CFD solution. This leads to the conclusion on how to deal with the complexity that the developer must carry out to construct an effective CFD adjoint solver that performs acceptable for industrial problems and applications.

References

[1]U. Naumann. The Art of Differentiating Computer Programs. An Introduction to Algorith- mic Differentiation. SIAM, Philadelphia, 2012.

[2]A. Griewank and A. Walther. Evaluating Derivatives: Principles and Techniques of Algo- rithmic Differentiation. SIAM, Philadelphia, Jan 2000.

[3]Z. Dastouri, J. Lotz, and U. Naumann. Development of a discrete adjoint cfd code using al- gorithmic differentiation by operator overloading. In OPT-i: An International Conference on Engineering and Applied Sciences Optimization, Athens, 2014. National Technical University of Athens.

[4]Z. Dastouri and U. Naumann. Improving efficiency of a discrete adjoint cfd code for design optimization problems. In EUROGEN: Proceeding of International Conference on Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems, Glasgow, UK, 2015.

[5]Z. Dastouri, S.M. Gezgin and U. Naumann. A Mixed Operator Overloading and source transformation approach for adjoint CFD computation. In Eccomass: Proceeding of European Congress on Computational Methods in Applied Sciences and Engineering, Crete Island, Greece, 2016.

[6]S.V. Patankar and D.B. Spalding. A calculation procedure for heat, mass and momen- tum transfer in three-dimensional parabolic flows. International Journal for Heat Mass Transfer, 15(10):17871806, 1972.

[7]D. Jones, F. Christakopoulos, and J-D. Muller. Preparation and assembly of adjoint cfd codes. Computers and Fluids, 46(1):282286, July 2011.

[8]B. Christianson. Reverse accumulation and attractive fixed points. Optimization Methods and Software, 3:311326, 1994.

[9]L. Hasco et and V. Pascual. The Tapenade Automatic Differentiation tool: Principles, Model, and Specification. ACM Transactions On Mathematical Software, 39(3), 2013.

[10]CFD-ACE+ Advanced CFD and Multiphysics Solver, ESI Group, http://www.esi-cfd.com/