8th. World Congress on Computational Mechanics (WCCM8)
5th European Congress on Computational Methods in Applied Sciences and Engineeering (ECCOMAS 2008)
June 30 –July 5, 2008
Venice, Italy

SPARSE FORWARD MODE AUTOMATIC DIFFERENTIATION APPLIED TO SHAPE OPTIMIZATION

*** Jukka I. Toivanen[1], Raino A. E. Mäkinen[2]**

University of Jyväskylä
Dept. Mathematical Information Technology
P.O. Box 35 (Agora)
FI-40014 University of Jyväskylä
[1] jukka.toivanen@jyu.fi
[2] raino.makinen@jyu.fi

Key Words: *Automatic Differentiation, Shape Optimization*

ABSTRACT

Automatic (or algorithmic) differentiation [1] is a technique developed to differentiate computer codes exactly (up to numerical precision) and with minimal user intervention. The technique exploits the fact that every computer program executes a sequence of elementary arithmetic operations. A set of independent variables is defined in the beginning of the computation. In the case of shape optimization [2] the independent variables can be for example the geometrical design parameters or the mesh nodal co-ordinates. The chain rule of differentiation is then successively applied to every elementary operation throughout the computation. In so called forward mode of automatic differentiation the derivative information is propagated forward in the execution chain.

We have implemented a simple lightweight dynamic sparse forward derivative propagation technique. Dynamic means that we automatically at run time capture the relationships between the dependent and the independent variables (index domains of the dependent variables). By sparse we mean that the derivative information of each intermediate variable is saved in a sparse representation. This allows the computation of non-zero partial derivatives only, without the need of separate sparsity pattern detection and graph coloring [3] phases. Details of our implementation are given in [4]. Bischof et al. present a similar technique in [5], using an additional library called SparsLinC for the sparse storage of the derivatives.

Our implementation is based on the operator overloading technique of C++ programming language. Thanks to the operation overloading the code exploiting automatic differentiation is mostly identical to the one that uses regular real variables, since the compiler takes care of calling the appropriate functions implementing the derivative computation. Thus the work of converting an original simulator into a one that computes also the geometrical sensitivities of the solution comprises mostly of replacing the variables with their AD counterparts where needed.

This automatic index domain capturing makes the technique easy on the developer of the code, since the developer does not have to manually keep track of the dependencies of the variables, which could

require a special structure from the code. This is especially important when the technique is applied to an existing solver.

Applicability of the implementation is demonstrated by shape optimization examples. We have applied so called pseudo-solid approach to solve free boundary problems of Bernoulli type [6]. This approach treats the free boundary problem as a coupled non-linear problem, which is solved using Newton iteration with an exact Jacobian. The location of the free boundary is then optimized by adjusting the shape of another boundary. To do this, the non-linear state problem is differentiated with respect to the geometry. Implementation of this technique is greatly facilitated by the use of the presented automatic differentiation technique.

An existing antenna simulation software based on the method of moments (integral equation formulation) has also been differentiated with respect to the geometry using the presented technique [4]. Virtually no changes to the program structure had to be made. Computational performance of the original simulation code and an automatically differentiated version were compared, and it was found out that assembly time of the system matrix was only a little over two times slower when the regular `double` and `complex<double>` types were replaced by their AD counterparts. The computation time naturally increases in the number of derivatives that have to be computed, but the increase is linear only if all the independent variables affect all the system matrix elements.

REFERENCES

[1] A. Griewank. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation.* SIAM, 2000.

[2] J. Haslinger, and R.A.E. Mäkinen, *Introduction to Shape Optimization: Theory, Approximation, and Computation*, SIAM, 2003.

[3] A.H. Gebremedhin, F. Manne, and A. Pothen, "What Color is Your Jacobian? Graph Coloring for Computing Derivatives", *Siam Review*, Vol. **47**, 629–705, 2005.

[4] J.I. Toivanen, R.A.E Mäkinen, S. Järvenpää, P. Ylä-Oijala, and J. Rahola, "Electromagnetic Sensitivity Analysis and Shape Optimization Using Method of Moments and Automatic Differentiation", Submitted to *IEEE Transactions on Antennas & Propagation.*

[5] C.H. Bischof, P.M. Khademi, A. Buaricha, and C. Alan. "Efficient Computation of Gradients and Jacobians by Dynamic Exploitation of Sparsity in Automatic Differentiation". *Optimization Methods and Software*, Vol. **7**, 1–39, 1996.

[6] J.I. Toivanen, R.A.E. Mäkinen, and J. Haslinger, "Shape Optimization of Systems Governed by Bernoulli Free Boundary Problems", Submitted to *Computer Methods in Applied Mechanics and Engineering.*