The Rapsodia tool for Fast Higher-Order Derivative Tensor computations

I. Charpentier[1] and J. Utke[2]

A number of practical problems in physics can be solved by using accurate higher-order derivatives. One example is the double ionization of a molecule modeled using Brauner's method [1]. This method is based on Hylleraas type wavefunctions

$$\sum_{a,b,c} r_1^a r_2^b r_{12}^c e^{-ar_1} e^{-br_2} e^{-cr_{12}}, \quad (1)$$

whose terms are derivatives (in a, b and c) of the complex function $\psi_i = e^{-ar_1} e^{-br_2} e^{-cr_{12}}$. Such derivatives can be obtained with automatic differentiation. However, one has to be concerned with the complexity of computing higher-order derivative tensors even for a modest order and number of independents.

The initial impetus for developing Rapsodia [2] came from the practical need to compute higher-order derivatives in complex arithmetics. Like AD02 and Adol-C, Rapsodia relies on operator overloading as the vehicle of attaching derivative computations to the elementary operations provided by the programming language. The main idea of Rapsodia is to combine operator overloading with code generation. The Python generator creates a library consisting of active types and operators overloaded for these active types for a given number n of input variables and a given derivative order o. Because n and o are fixed, the generator can create a specialized code which yields a performance advantage. Not surprisingly, very few differences exist between general-purpose languages such as Fortran and C/C++ in the representation of the elementary operations that have to be overloaded. Therefore, the idea to use a single generator to produce both, a Fortran and a C++ library, is plausible. Rapsodia provide a Fortran and a C++ implementation of the efficient approach to compute a derivative tensor [3].

Initial experiments using univariate Taylor polynomials with interpolation and operator overloading with unrolled loops showed better runtimes than using other automatic differentiation tools. The generator and the library are open source and can be downloaded under the LGPL terms at [4].

References

[1] M. Brauner, J. Briggs, and H. Klar. J. Phys. B.: At. Mol. Phys., 1989.
[2] I. Charpentier and J. Utke. Optimization Methods and Software, 2009.
[3] A. Griewank, J. Utke, and A. Walther. Mathematics of Computation, 2000.
[4] Rapsodia website. http://www.mcs.anl.gov/rapsodia.

[1]Laboratoire de Physique et Mécanique des Matériaux (CNRS 3236), Ile du Saulcy, 57045 Metz Cedex 1, France, isabellecharpentier@hotmail.com
[2]Mathematics and Computer Science Division, Argonne National Laboratory, 9700 S. Cass Avenue Argonne, IL 60439,USA, utke@mcs.anl.gov