MINIMIZING OPERATIONS COUNTS AND MAXIMIZING DATA LOCALITY FOR EFFICIENT DERIVATIVE CODES IN AUTOMATIC DIFFERENTIATION*

ANDREW LYONS† AND JEAN UTKE‡

Automatic differentiation (AD) [4], a technique for the accurate and efficient computation of derivative information, plays an important role in areas such as optimization and data assimilation. We consider the efficiency of Jacobian computations performed by generated code. Fully exploiting the associativity of the chain rule yields the combinatorial problem of minimizing an operations count as an efficiency measure [3]. Solving this combinatorial problem creates savings in terms of this measure. However, these savings do not always translate into runtime improvements, in particular when the generated code is pushed through an optimizing compiler; one suspects the lack of data locality as a slowing effect. Because of the complexity of the AD code transformation itself, currently no AD tool attempts to adapt this transformation toward specific optimizing compiler and hardware combinations. Only recently have attempts been made to develop a generic heuristic to improve data locality in the code generated by the EliAD tool [2].

We present the data locality concept in the context of the code generation from so-called vertex, edge, or face elimination as implemented in the AD tool OpenAD [6]. We introduce a variety of generic heuristics and, using application examples, compare the impact of data locality vs. minimizing the operations count in the presence of compiler optimization.

1. Locality Considerations for Jacobian Code. Consider the application of AD for computing the Jacobian of a vector function $F : \mathbb{R}^n \to \mathbb{R}^m$ that is implemented as a computer program. As a simple example, let $F : \mathbb{R}^2 \to \mathbb{R}^2$ be defined as $y_1 = sin(x_1 * (x_1 * x_2))$, $y_2 = cos(x_1 * (x_1 * x_2))$. We begin by using the code for F in Fig. 1.1 (a) to construct the computational graph $G = (V, E)$ as shown in Fig. 1.1 (b).

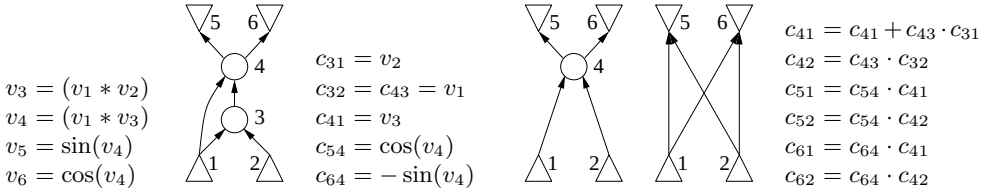

FIG. 1.1. *(a) Code for F, (b) G, (c) code for the c_{ji}, (d) G′, (e) G″, (f) accumulation code*

G is a directed acyclic graph whose vertices v_1, v_2 correspond to the $n = 2$ input variables, v_5, v_6 to the $m = 2$ output variables, and v_3, v_4 to the intermediate values in the evaluation of F. The local partial derivatives $c_{ji} = \frac{\partial v_j}{\partial v_i}$ may be seen as labels on the respective edges $(i, j) \in E$ and are computed by code as in the example in

*Abstract for the SIAM Workshop on Combinatorial Scientific Computing (CSC05)

† Department of Electrical Engineering and Computer Science, Vanderbilt University, 2201 West End Avenue, Nashville, TN 37235, USA (andrew.m.lyons@vanderbilt.edu)

‡Mathematics and Computer Science Division, Argonne National Laboratory, 9700 S. Cass Avenue, Argonne, IL 60439, USA (utke@mcs.anl.gov)

Fig. 1.1(c). The combinatorial problem is minimizing the number of *fused multiply-add operations* during the accumulation of F' as the application of the chain rule where incident edge labels are multiplied and labels of parallel edges are added. The problem can be represented by purely structural vertex or edge elimination in G or face elimination in the corresponding directed line graph [5] and is conjectured to be NP-hard. In the example we show in Fig. 1.1(f) the code resulting from eliminating vertices 3 (see Fig. 1.1(d)) and 4 (see Fig. 1.1(e)). The last four assignments in Fig. 1.1(f) representing the edge labels of the bipartite G'' in Fig. 1.1(e) are the entries of F'. In a small example as the one shown in Fig. 1.1(f) there are not many choices but we consider larger examples with hundreds of edges.

The complexity of the minimization problem requires the use of heuristics [1]. We introduce the code generation controlled by sequence of heuristics as implemented in the AD tool OpenAD. The first stage of addressing data locality is within the accumulation code. We introduce basic data locality heuristics, such as so-called sibling degree, for the code generation in the context of vertex, edge, and face elimination. Runtime measurements for practical examples illustrate the relevance of giving either the data locality or the operations count minimization preference in the code generation. In particular, we analyze the effect of the data locality heuristic with and without subsequent compiler optimization.

The second stage is the data locality, not just within the accumulation code (in the example Fig. 1.1(f)) but in relation to the computation of the local partials and the function evaluation code itself (in the example Fig. 1.1(a) and (c)). We compare Jacobian accumulation code introduced above with code that performs preaccumulation only at a statement level.

REFERENCES

[1] A. Albrecht, P. Gottschling, and U. Naumann. Markowitz-type heuristics for computing Jacobian matrices efficiently. In *ICCS 2003*, volume 2658 of *LNCS*, pages 575–584, Berlin, 2003. Springer.

[2] Shaun A. Forth, Mohamed Tadjouddine, John D. Pryce, and John K. Reid. Jacobian code generated by source transformation and vertex elimination can be as efficient as hand-coding. *ACM Trans. Math. Softw.*, 30(3):266–299, 2004.

[3] A. Griewank and S. Reese. On the calculation of Jacobian matrices by the Markovitz rule. In G. Corliss and A. Griewank, editors, *Automatic Differentiation: Theory, Implementation, and Application*, pages 126–135, Philadelphia, 1991. SIAM.

[4] Andreas Griewank. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation.* Number 19 in Frontiers in Appl. Math. SIAM, Philadelphia, PA, 2000.

[5] U. Naumann. Optimal accumulation of Jacobian matrices by elimination methods on the dual computational graph. *Math. Prog.*, 3(99):399–421, 2004. published online at `www.springerlink.com`.

[6] U. Naumann, J. Utke, and A. Walther. An introduction to using and developing software tools for automatic differentiation. In P. Neittaanmäki, T. Rossi, S. Korotov, E. O nate, J. Périaux, and D. Knörzer, editors, *Proceedings of the European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS 2004)*, Jyväskylä, Finland, 2004. University of Jyväskylä. published online at http://www.mit.jyu.fi/eccomas2004/proceedings/pdf/702.pdf.