ON FACE ELIMINATION IN COMPUTATIONAL GRAPHS

UWE NAUMANN* AND JEAN UTKE†

The Jacobian matrix F' of a vector function $F : \mathbb{R}^n \to \mathbb{R}^m$ written as a computer program can be computed with machine accuracy by a source transformation technique known as *automatic differentiation* (AD) [2]. AD is built on the assumption that the arithmetic operators (for example, $*$ and $+$) and intrinsic functions (for example, sin and log) have jointly continuous local partial derivatives at the current argument $\mathbf{x} \in \mathbb{R}^n$. AD exploits the associativity of the chain rule to compute derivatives on the basis of the dependency information structurally represented by paths in the computational graph. The derivative computation can be represented by elimination steps in the computational graph. In this context we can define an *optimal* Jacobian accumulation criterion as the number of operations incurred by applying the chain rule.

In continuation of our presentation at the 2004 CSC workshop we outline a proof for the optimality of a particular approach and look at the impact of this optimality criterion on the efficiency of AD generated code.

1. Jacobians and Computational Graphs. For a given argument $\mathbf{x}$ the computation performed by the program that implements $F(\mathbf{x})$ can be represented by the computational graph G (directed and acyclic). For example, a function $(v_5, v_6) = F(v_1, v_2)$ that is implemented as $v_3 = \varphi_3(v_1, v_2), v_4 = \varphi_4(v_3), v_5 = \varphi_5(v_3, v_4), v_6 = \varphi_6(v_2)$, where the *elemental* functions φ_j represent arithmetic operations or intrinsic functions, leads to the computational graph shown in Fig. 1.1(a). Assuming local

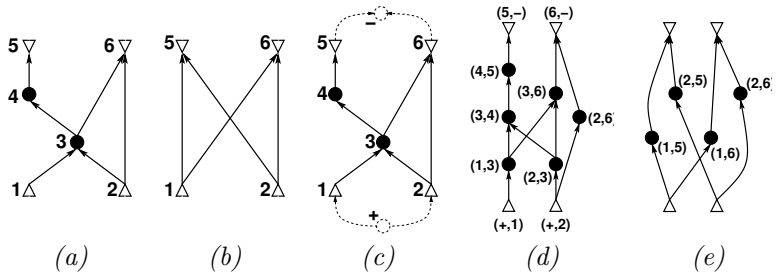

FIG. 1.1. *(a) Computational graph G; (b) bipartite G' representing F'; (c) $\hat{G}$, the augmented G; (d) directed line graph $\mathcal{G}$ of $\hat{G}$; (e) tripartite $\mathcal{G}'$ representing F'*

differentiability of the elemental functions, the values of the local partial derivatives are attached to the corresponding edges in G. For example, edge $(3, 6)$ is labeled with $c_{6,3} \equiv \frac{\partial \varphi_6}{\partial v_3}$. The accumulation of F' is represented by transforming G into the directed bipartite graph G' that has an edge for every structural nonzero element in F'. An example is depicted in Fig. 1.1(b). A sequence of vertex [4], edge, or face elimination [6] steps perform the transformation to G'. Face elimination can undercut the operations count of vertex or edge elimination in the general case [5].

*Software and Tools for Computational Engineering, RWTH Aachen University, D-52056 Aachen, Germany (naumann@stce.rwth-aachen.de)

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

Face eliminations operate on the directed line graph $\mathcal{G}$, (see Fig. 1.1(d)) of an augmented version $\hat{G}$ (see Fig. 1.1(c)) of G. The augmentation is needed for the bijective mapping $G \leftrightarrow \mathcal{G}$. Each face elimination step removes an edge in $\mathcal{G}$ and eventually leads to the Jacobian represented by a tripartite graph as shown in Fig. 1.1(e). Face elimination reduces the granularity of an elimination step to a single multiplication operation.

2. Problems and Outline of the Talk. We explain the augmentation of the computational graph, the construction of the directed line graph $\mathcal{G}$, and the face elimination process. Counting operations in vertex, edge, and face eliminations exhibits subtle differences. We elaborate on these differences in the context of the optimality criterion we want to use for the Jacobian accumulation.

Some graph modifications, such as *scarcity preserving* rerouting steps [3], are not covered by chain rule arithmetic. We define the optimal Jacobian accumulation (OJA) under chain rule arithmetic as a computation of Baur's formula [1] with minimal multiplication count. We highlight the differences to alternative graph modifications and the relation to the count of additions.

Although not formally proven it is conjectured and widely accepted that minimizing the operations count in this manner is an NP-hard problem. Until now there has been no proof that one can attain the minimum with face elimination either. Face elimination does not cover all possibilities of computing Baur's formula, particularly the commutativity and distributivity of the multiplications. We outline a proof showing that there is no exploit of chain rule arithmetic manipulations in Baur's formula that can undercut an optimal face elimination, thereby establishing that face elimination can indeed solve OJA.

The talk will conclude with application examples that investigate the practical impact that this kind of optimization has on the efficiency of codes generated by an AD tool.

REFERENCES

[1] W. Baur and V. Strassen. The complexity of partial derivatives. *Theoretical Computer Science*, 22:317–330, 1983.

[2] A. Griewank. *Evaluating Derivatives. Principles and Techniques of Algorithmic Differentiation.* Number 19 in Frontiers in Applied Mathematics. SIAM, Philadelphia, 2000.

[3] A. Griewank. A mathematical view of automatic differentiation. *Acta Numerica*, 12:321–398, 2003.

[4] 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.

[5] U. Naumann. Elimination techniques for cheap Jacobians. In G. Corliss, C. Faure, A. Griewank, L. Hascoet, and U. Naumann, editors, *Automatic Differentiation of Algorithms – From Simulation to Optimization*, pages 247–253, New York, 2002. Springer.

[6] U. Naumann. Optimal accumulation of Jacobian matrices by elimination methods on the dual computational graph. *Math. Prog.*, 3(99):399–421, 2004.