1. An Introduction to Automatic Differentiation

Louis B. Rall* George F. Corliss†

Abstract

This paper provides a gentle introduction to the field of automatic differentiation (AD), with the goal of equipping the reader for the other papers in this book. AD is the systematic application of the familiar rules of calculus to computer programs, yielding programs for the propagation of numerical values of first, second, or higher derivatives. AD can be regarded as traversing the code list (or computational graph) in the forward mode, the reverse mode, or a combination of the two. Algorithms for numerical optimization, differential equations, and interval analysis all could use AD technology to compute the required derivatives. AD typically is implemented by using either source code transformation or operator overloading. We give examples of code for each. Finally, we outline some pitfalls of AD for naive users, and we present opportunities for future research.

Keywords: Code list, forward mode, reverse mode, source code transformation, operator overloading.

1 What Is Automatic Differentiation?

This paper is a general introduction to automatic differentiation (AD), also known as computational differentiation, algorithmic differentiation, and differentiation of algorithms. We present the basic ideas with a minimum of technicalities. AD is a process for evaluating derivatives which depends only on an algorithmic specification of the function to be differentiated. In actual practice, the specification of the function is all or part of a computer program, and the derivative values are produced by the execution of the program derived from the program for the original function, hence the term "automatic."

Algorithmic specification of the derivatives of the function to be differentiated is not required, just specification of the function itself. Consequently, it should be made clear from the outset what automatic differentiation is *not*. It is neither the process of symbolic differentiation taught in calculus nor the divided differences of classical numerical analysis. For example, given the function

$$f(x,y) = (xy + \sin x + 4)(3y^2 + 6), \tag{1}$$

defined by a formula, symbolic differentiation produces formulas for its derivatives,

$$\begin{aligned} \frac{\partial f}{\partial x} &= (y + \cos x)(3y^2 + 6) = 3y^2 \cos x + 6\cos x + 3y^3 + 6y, \\ \frac{\partial f}{\partial y} &= 6y(xy + \sin x + 4) + x(3y^2 + 6) = 9xy^2 + 6y \sin x + 24y + 6x. \end{aligned}$$

*Department of Mathematics, University of Wisconsin, Madison, WI (rall@math.wisc.edu).

†Department of Mathematics, Marquette University, Milwaukee, WI (georgec@mscs.mu.edu). Supported in part by National Science Foundation Grant No. DMS–9413525.

In principle, evaluation of these formulas gives exact values of the derivatives of $f(x,$ However, an actual computation is subject to unavoidable roundoff error resulting from t individual floating-point operations.

As in the case with symbolic differentiation, the values for derivatives obtained by A are exact (to roundoff). Although great advances have been made in symbolic differentiatic of formulas, automatic differentiation generally requires less memory and CPU time a also applies to functions defined by computer programs or subroutines for which formula may be available. AD also works well in conjunction with symbolic processi capabilities [Monagan1996a]. AD is the method of choice in many applications requiri gradient vectors or Hessian matrices of functions of a substantial number of variables, in cases in which many (even hundreds or thousands of) terms of Taylor series expansio of solutions of systems of ordinary differential equations are needed.

Divided differences, on the other hand, produce approximations to values of derivativ based on the use of difference quotients involving only function evaluations, for example.

$$\frac{\partial f}{\partial x} \approx \frac{f(x+\Delta x, y) - f(x-\Delta x, y)}{2\Delta x} = \frac{\partial f}{\partial x} + O(\Delta x^3), \tag{2}$$

where the term $O(\Delta x^3)$ denotes the (unknown) truncation error of this central differe approximation to $f_x(x,y)$. The rate at which the truncation error approaches zero c be increased at the cost of more function evaluations [Milne1949a], but divided differenc are inherently inexact except for polynomials of sufficiently low degree in the variable interest. Reduction of the truncation error in a finite-difference approximation such a (2) by reducing the size of Δx can be thwarted by roundoff error. Significant digits a cancelled in the numerator, and the resulting error is magnified by division by a sma divisor. In contract, the values for derivatives obtained by AD are exact and are oft *much* less expensive to compute.

2 How Does AD Work?

AD works whenever the chain rule holds. The techniques of AD are based on the systema application of the chain rule, but go far beyond it, as the papers in this book demonstra

Much of the ordinary introduction to differential calculus focuses on training studen to produce formulas for derivatives of functions also defined by formulas. Consequently, t idea of evaluating derivatives exactly, without formulas for them, is novel to many. In t section, an informal introduction to the methodology will be given for those unfamiliar wi the concept; technicalities are postponed to the following sections. The theoretical exactne of automatic differentiation stems from the fact that it uses the same rules of differentiatic as learned in elementary calculus, but these rules are applied to an algorithmic specificatic of the function rather than to a formula. It is precisely this feature that makes automa differentiation suitable for machine computation (i.e., "automatic"). To illustrate this, le us back up a little and consider how to *evaluate* (rather than differentiate) the functic defined by equation (1). One starts with the values of x and y, builds up each factor, a then multiplies them to obtain the final result. The steps involved could be written:

$$\begin{aligned} t_1 &= x, & t_6 &= t_5 + 4, \\ t_2 &= y, & t_7 &= t_2^2, \\ t_3 &= t_1 t_2, & t_8 &= 3t_7, \\ t_4 &= \sin t_1, & t_9 &= t_8 + 6, \\ t_5 &= t_3 + t_4, & t_{10} &= t_6 t_9. \end{aligned}$$