Math. 455-01
Spring, 1992
R. B. Kearfott

Elements of Automatic Differentiation

Automatic Differentiation is a procedure for simultaneously computing values of both a function and its derivatives. It differs from *symbolic differentiation*, in which the computer produces a symbolic expression for derivatives. However, though automatic differentiation gives numerical values for both the function value and its derivative or derivatives, it does not involve any truncation error, as in numerical differentiation.

The idea behind automatic differentiation is definition of the operations underlying function evaluation on *ordered pairs* $[u, u']$, instead of just on real numbers u. The automatic differentiaton then proceeds by performing these operations in the order that would have been done had the function been evaluated in real arithmetic.

Rules for the four elementary operations are

$$[u, u'] + [v, v'] = [u + v, u' + v'], \tag{1}$$

$$[u, u'] - [v, v'] = [u - v, u' - v'], \tag{2}$$

$$[u, u'] \cdot [v, v'] = [uv, uv' + vu'] \tag{3}$$

$$[u, u']/[v, v'] = [u/v, (u'v - uv')/v^2] \tag{4.}$$

We also have rules such as

$$\sin([u, u']) = [\sin u, u' \cos u], \tag{5}$$

$$[u, u']^2 = [u^2, 2uu'], \tag{6}$$

$$[u, u']^3 = [u^3, 3u^2 u'], \tag{7}$$

etc.

With these rules, we may evaluate, for example, $f(x)$ and $f'(x)$ at $x = 1$, where

$$f(x) = \left[\sin\left(\pi x^3\right)\right]^2.$$

We obtain

$$[x, 1] \to [x, 1]^3 = [x^3, 3x^2] = [1, 3],$$

$$[1, 3] \to \pi[1, 3] = [\pi, 3\pi],$$

$$[\pi, 3\pi] \to \sin([\pi, 3pi]) = [\sin \pi, 3\pi \cos \pi] = [0, -3\pi],$$

$$[0, -3\pi] \to [0, -3\pi]^2 = [0, 2 \cdot 0 \cdot (-3\pi)] = [0, 0].$$

Above, we identified π with a constant whose derivative was zero, so that we used (3) to get $\pi[1, 3] = [\pi, 3\pi]$.

Typeset by $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX

Note that symbolic differentiation (by human or by machine) gives

$$f'(x) = 2\left[\sin(\pi x^3)\right]\cos(\pi x^3)\pi(3x^2),$$

which evaluates to 0.

Generally, we would have subroutines to do the operations in (1) through (7). For example, we would have subroutines

(1) SUBROUTINE ADD(U,UP,V,VP,R,RP),
(2) SUBROUTINE SUB(U,UP,V,VP,R,RP),
(3) SUBROUTINE MULT(U,UP,V,VP,R,RP),
(4) SUBROUTINE DIV(U,UP,V,VP,R,RP),
(5) SUBROUTINE SIN(U,UP,R,RP),
(6) SUBROUTINE SQR(U,UP,R,RP), and
(7) SUBROUTINE CUB(U,UP,R,RP).

Let us assume that the above subroutines are "safe" in the sense that we may duplicate variables as both input arguments and output arguments (e.g. CALL CUB(R,RP,R,RP)). If we were programming them, we could certainly arrange this by storing the input arguments in temporary variables upon entry, and using those values in computing the output variables. Evaluation of f as above could then proceed by the following lines.

```
U = 1
UP = 1
CALL CUB(U,UP,R,RP)
V = 4.*ATAN(1.)
VP = 0.
CALL MULT(V,VP,R,RP,R,RP)
CALL SIN(R,RP,R,RP)
CALL SQR(R,RP,R,RP)
```

Some computer languages (such as Pascal-SC, C++, Fortran-SC, or Fortran-8X) allow *operator overlaying.* In those languages, we may define a new datatype consisting of pairs of the form $[u, u']$, and we may define the operations on that datatype in terms of the subroutine package we supply. When the compiler then encounters an operation with one or more of the new datatypes, it translates it to a call to a subroutine from our package. Writing a program to automatically differentiate a function thus becomes as simple as writing a program to evaluate the function itself.

Exercises.

1. *Write automatic differentiation subroutines, as described above, for addition, subtraction, multiplication, and division. Make them "safe" but efficient.*
2. *Write a subroutine, which uses your routines, to evaluate f and f', where*

$$f(x) = 1 + 3x - x(x-1) + \frac{2}{3}x(x-1)(x-2).$$

3. *Use your subroutine to evaluate the f and its derivative at*

$$x_i = (i-1)h, \quad 1 \le i \le n+1, \quad h = 1/n, n = 100.$$