Forward, reverse and Taylor AD by examples

Daniel Wilczak

Department of Mathematics
University of Uppsala, Sweden
and
Department of Computer Science
Jagiellonian University, Poland

The Automatic differentiation tools allows us to compute numerical values of derivatives of a wide class of functions. These include explicit given functions, but also functions defined in a more complicated way (implicit functions, local Taylor expansions of manifolds, solutions to ODEs, Poincaré maps, etc.). Here we will present some very basic schemes for explicit given functions.

By a *simple* function we mean a function $f : \mathbb{R}^n \to \mathbb{R}^m$ which is a finite composition of arithmetic operations $+, -, *$, and elemental functions $\sin, \cos, \exp, \cdots$.

1 Forward automatic differentiation.

1.1. One dimensional case.

The forward mode of AD is often implemented by means of operators and functions overloading technique. Instead of computing on numbers, we perform computations on pairs (u, u'). The first element will hold the actual value of some expression, the second component will hold the derivative of this quantity with respect to the initial variable.

We define the following operations on pairs of numbers (u, u'):

- $(u, u') \pm (v, v') = (u \pm v, u' \pm v')$,
- $(u, u') \times (v, v') = (uv, uv' + u'v)$,
- $(u, u') \div (v, v') = (u/v, (u' - (u/v)v')/v)$.

In a similar way we can define operations for all elemental functions. For example

- $\sin(u, u') = (\sin(u), \cos(u)u')$,
- $\exp(u, u') = (\exp(u), \exp(u)u')$,

etc.

1.2. Example.

Let $f(x) = \dfrac{(x+1)(x-2)}{x+3}$. We will compute the value of $f(3)$ and $f'(3)$ using forward mode. We replace each constant c in the formula defining f by the pair $(c, 0)$ and the variable x by the pair $(x, 1)$. This means that the derivative of input variable x with respect to x is equal to 1 and the derivative of each constant is just zero.

Applying arithmetic defined on pairs we compute simultaneously the value and the derivative of f at $x = 3$.

$$\frac{((3,1)+(1,0))\times((3,1)-(2,0))}{(3,1)+(3,0)} = \frac{(4,1)\times(1,1)}{(6,1)} = \frac{(4,5)}{(6,1)} = \left(\frac{2}{3}, \frac{13}{18}\right)$$

1.3. Directional derivatives of multivariate functions.

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a simple function and let us fix $u, x \in \mathbb{R}^n$. We will show how we can easily compute the numerical value of $Df(x)\cdot u$. Let us define the following function

$$g(t) = f(x + tu).$$

It is well known that

$$g'(0) = Df(x)\cdot u.$$

Hence, we reduced the problem of computing $Df(x)\cdot u$ to the problem of computing the derivative of an univariate function. We will apply already introduced method for univariate functions. In order to compute $g'(0)$ it is enough to evaluate $g(0)$ in the arithmetic defined on pairs. To this end, we replace t by the pair $(0, 1)$ (recall we compute the derivative at $t = 0$) and vectors x and u by

$$((x_1, 0), (x_2, 0), \ldots, (x_n, 0)), \quad ((u_1, 0), (u_2, 0), \ldots, (u_n, 0)),$$

respectively (here x and u are constant).

1.4. Example.

Let $f(x_1, x_2, x_3) = \dfrac{x_1 + x_2x_3 + 1}{x_1 + x_3}$. Let us fix $x = (1, 2, 3)$ and $u = (1, 4, 2)$. Proceeding as described above we obtain

$$g(t) = f(1+t, 2+4t, 3+2t) = \frac{1+t+(2+4t)(3+2t)+1}{(1+t)+(3+2t)}.$$

Then we replace any occurrence of t by the pair $(0, 1)$ and each constant c by $(c, 0)$. With some abuse of notation on f and g we obtain

$$\begin{aligned} g((0,1)) &= f((x_1,u_1),(x_2,u_2),(x_3,u_3)) \\ &= \frac{(1,1)+(2,4)\times(3,2)+(1,0)}{(1,1)+(3,2)} = \frac{(1,1)+(6,16)+(1,0)}{(4,3)} = \frac{(8,17)}{(4,3)} = \left(2, \frac{11}{4}\right). \end{aligned}$$

1.5. Full gradients of multivariate functions by forward mode.

In the previous section we have seen that the computation of all partial derivatives of a simple function $f : \mathbb{R}^n \to \mathbb{R}$ at a point $x \in \mathbb{R}^n$ can be obtained by evaluation of $Df(x) \cdot e_i$, $i = 1, \ldots, n$ where e_i are vectors from the standard basis in $\mathbb{R}^n$.

It turns out, however, that it is more efficient to compute simultaneously several partial (or in general directional) derivatives. To this end, we can extend the arithmetic defined on pairs to the arithmetic defined on $(m+1)$-dimensional vectors in the following way:

- $(u, u_1, u_2, \ldots, u_m) \pm (v, v_1, v_2, \ldots, v_m) = (u \pm v, u_1 \pm v_1, \ldots, u_m \pm v_m)$,
- $(u, u_1, u_2, \ldots, u_m) \times (v, v_1, v_2, \ldots, v_m) = (uv, u_1 v + v_1 u, \ldots, u_m v + v_m u)$,
- $(u, u_1, u_2, \ldots, u_m) \div (v, v_1, v_2, \ldots, v_m) = (uv, (u_1 - (u/v)v_1)/v, \ldots, (u_m - (u/v)v_m)/v)$

and similarly for elemental functions. Here, the first element of the vector will store the actual value of an expression. The remaining elements will hold actual derivatives with respect to m arbitrary chosen direction vectors.

Let us fix $x \in \mathbb{R}^n$ and m direction vectors $u_1, \ldots, u_m \in \mathbb{R}^n$. In order to compute a vector

$$(Df(x) \cdot u_1, Df(x) \cdot u_2, \ldots, Df(x) \cdot u_m)$$

we replace in the formula for f

- each constant c by the sequence $(c, 0, \ldots, 0) \in \mathbb{R}^{m+1}$,
- each occurrence of x_i, $i = 1, \ldots, n$ by the sequence $(x_i, u_{1,i}, \ldots, u_{m,i})$,

and evaluate this expression in our vector arithmetic. In particular, to obtain just a gradient of f we can set $m = n$ and take $u_i = e_i$, where e_i are vectors from the standard basis in $\mathbb{R}^n$.

1.6. Example.

Let $f(x, y) = \frac{x+y+1}{xy-1}$. We will compute the value and gradient of f at $(2, -2)$. In this situation, our directional vectors will be $e_1 = (1, 0)$ and $e_2 = (0, 1)$. Therefore, we replace each occurrence of x by the vector $(x = 2, 1, 0)$, and each occurrence of y by $(y = -2, 0, 1)$. The constants c will be replaced by $(c, 0, 0)$. We have

$$\frac{(2,1,0) + (-2,0,1) + (1,0,0)}{(2,1,0) \times (-2,0,1) - (1,0,0)} = \frac{(1,1,1)}{(-4,-2,2) - (1,0,0)} = \frac{(1,1,1)}{(-5,-2,2)} = \left(\frac{-1}{5}, \frac{-3}{25}, \frac{-7}{25} \right). \quad (1)$$

Hence, $f(2, -2) = \frac{-1}{5}$, $\frac{\partial f}{\partial x}(2, -2) = \frac{-3}{25}$ and $\frac{\partial f}{\partial y}(2, -2) = \frac{-7}{25}$.

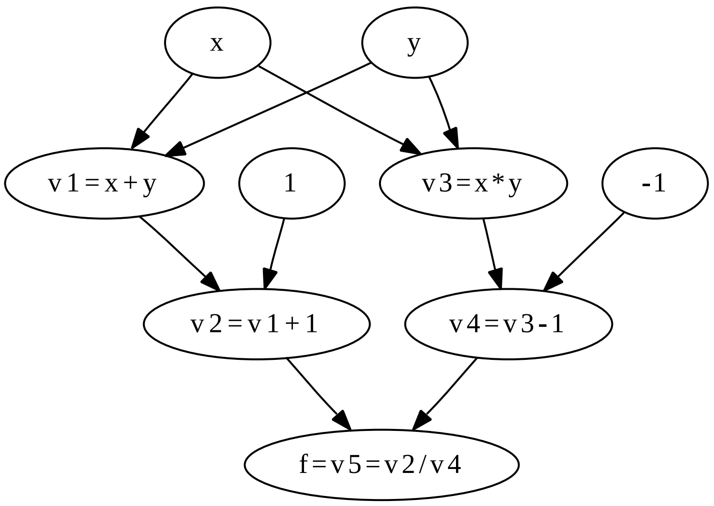
2 Reverse mode.

The reverse mode for computing derivatives seems a bit unnatural after the first look. We will see, however, that backward accumulation of derivatives has in some situations significant advantages over the forward mode.

2.1. Example.

Consider a function $f(x,y) = \frac{x+y+1}{xy-1}$. In the previous section we already computed derivatives of f at $(x,y) = (2,-2)$ using the forward mode.

Any simple function by its definition is a finite composition of arithmetic operations and elemental functions. Hence, it can be represented as a *direct acyclic graph* (DAG). In some cases, the representation is not unique due to associativity rules of addition and multiplication, but it is always possible to find some representation. In our case, the function f can be represented by



On the graph we defined new intermediate variables v_i, $i = 1, \dots, 5$. Given a point at which we want to compute derivatives we can easily compute and **store in the memory** all the values of the intermediate variables. We have

$$\begin{aligned}
v_1 &= x + y = 2 - 2 = 0,\\
v_2 &= v_1 + 1 = 0 + 1 = 1,\\
v_3 &= x * y = 2 * (-2) = -4,\\
v_4 &= v_3 - 1 = -4 - 1 = -5,\\
f = v_5 &= v_2/v_4 = 1/(-5) = -\frac{1}{5}.
\end{aligned}$$

Of course in the last line we computed the value of f at the point $(2,-2)$. With some abuse of notation on f we define the following symbols

$$\bar{v}_i = \frac{\partial f}{\partial v_i}.$$

Of course formally f is a function of variables x, y but we may think about $\bar{v}_i$'s as a sensitivities of f with respect to intermediate variable v_i. By the definition

$$f(v_5) = v_5,$$

therefore $\bar{v}_5 = 1$. We can think about v_5 as a function of v_2 and v_4 given by $v_5(v_2, v_4) = v_2/v_4$. Again 'overloading' the symbol f we see that

$$\bar{v}_2 = \frac{\partial}{\partial v_2}\left(f(v_5(v_2, v_4))\right) = \frac{\partial f}{\partial v_5} \cdot \frac{\partial v_5}{\partial v_2} = \bar{v}_5/v_4 = 1/(-5) = -\frac{1}{5}.$$

Similarly,

$$\bar{v}_4 = \frac{\partial}{\partial v_4}\left(f(v_5(v_2, v_4))\right) = \frac{\partial f}{\partial v_5} \cdot \frac{\partial v_5}{\partial v_4} = \bar{v}_5 \cdot \frac{-v_2}{v_4^2} = -\bar{v}_5 \cdot v_5/v_4 = -1 \cdot \frac{-1}{5}/(-5) = -\frac{1}{25}.$$

We see that v_2 is a function of v_1 given by $v_2(v_1) = v_1 + 1$. Therefore,

$$\bar{v}_1 = \frac{\partial f}{v_2} \cdot \frac{\partial v_2}{\partial v_1} = \bar{v}_2 = -\frac{1}{5}.$$

Similarly,

$$\bar{v}_3 = \frac{\partial f}{v_4} \cdot \frac{\partial v_4}{\partial v_3} = \bar{v}_4 = -\frac{1}{25}.$$

There remains for us to accumulate derivatives towards the main input variables x and y. We see that

$$\frac{\partial f}{\partial x}(2, -2) = \bar{v}_1 \cdot \frac{\partial v_1}{\partial x} + \bar{v}_3 \cdot \frac{\partial v_3}{\partial x} = \bar{v}_1 + \bar{v}_3 \cdot y = -\frac{1}{5} + \left(-\frac{1}{25}\right) \cdot (-2) = -\frac{3}{25},$$
$$\frac{\partial f}{\partial y}(2, -2) = \bar{v}_1 \cdot \frac{\partial v_1}{\partial y} + \bar{v}_3 \cdot \frac{\partial v_3}{\partial y} = \bar{v}_1 + \bar{v}_3 \cdot x = -\frac{1}{5} + \left(-\frac{1}{25}\right) \cdot 2 = -\frac{7}{25}.$$

What we computed agrees with the results obtained from forward propagation (1).

Remarks:

- In the reverse mode we simultaneously compute partial derivatives of a dependent quantity f with respect to all intermediate variables and in consequence with respect to all input variables.

- The method does not have any over-computations which can appear in 'naive' implementation of the forward mode. In the forward propagation the initial vectors often contain a lot of zeroes. In this case we propagate (and compute!) many zeroes until the vectors of actual derivatives become dense. One can avoid this problem in forward propagation but this requires some extra effort and more complicated implementation.

  This effect does not exist in the reverse mode.

- The reverse accumulation is strongly recommended to compute derivatives of functions $\mathbb{R}^n \to \mathbb{R}^m$, where $n >> m$. Otherwise, the forward mode is usually faster and less memory consuming.

3 The Taylor method for ODEs.

For a function $p : \mathbb{R} \to \mathbb{R}^n$ by $p^{(k)}(t)$ we will denote a vector of the k-th order derivatives of p at t, i.e $(p_1^{(k)}(t), \ldots, p_1^{(k)}(t))$.

By $p^{[k]}(t)$ we will denote a vector of the k-th order Taylor coefficients of p at t. Obviously $p^{(k)}(t) = k!p^{[k]}(t)$.

3.1. Univariate Taylor ring arithmetic.

Let us fix an integer $k \geq 0$. Let $p : \mathbb{R} \to \mathbb{R}$ be C^k-smooth function in a neighbourhood of zero. We can encode its k-th order Taylor polynomial at zero as a $(k + 1)$-dimensional vector $(p^{[0]}, \ldots, p^{[k]})$.

Let $p, q \in \mathcal{C}^k$ and assume that we know the Taylor coefficients of p and q at zero up to order k. It turns out that it is easy to compute k-th order Taylor approximations for $p \pm q$, $p \cdot q$, p/q and also for elemental functions $\sin(p)$, $\exp(p)$, etc. Below, we present an incomplete list of these formulas (see also presentation by Warwick Tucker).

- If $r = p \pm q$ then $r^{[i]} = p^{[i]} \pm q^{[i]}$, for $i = 0, \ldots, k$.
- If $r = p \cdot q$ then $r^{[i]} = \sum_{j=0}^{i} p^{[j]}q^{[i-j]}$, for $i = 0, \ldots, k$ (convolution).
- If $r = p/q$ then $r^{[i]} = \frac{1}{q^{[0]}}\left(p^{[i]} - \sum_{j=0}^{i-1} r^{[j]}q^{[i-j]}\right)$, for $i = 0, \ldots, k$.

The computational complexity of these formulas is very small - all the operations can be perform with the number of multiplications of the order $\mathcal{O}(k^2)$.

3.2. Computing of Taylor series of the solutions to ODE's.

Consider the Cauchy problem

$$x' = f(x), \quad x(0) = x_0, \tag{2}$$

where $f : \mathbb{R}^n \to \mathbb{R}^n$ is a simple function. Since f is analytic, the solution to (2) exists in a neighbourhood of zero. Let us denote this function by $x : t \to x(t)$.

Put $F = f \circ x$. The function F is again an analytic function in a neighbourhood of zero, hence it make sense to compute its Taylor coefficients. According to (2) we have the following equality of two power series

$$\frac{d}{dt}\left(\sum_{i=0}^{\infty} x^{[i]}(0)t^i\right) = \sum_{i=0}^{\infty} F^{[i]}(0)t^i.$$

Differentiating left side and shifting the indexes we obtain

$$\frac{d}{dt}\left(\sum_{i=0}^{\infty} x^{[i]}(0)t^i\right) = \sum_{i=1}^{\infty} i\, x^{[i]}(0)t^{i-1} = \sum_{i=0}^{\infty}(i+1)x^{[i+1]}(0)t^i.$$

Therefore, for $i = 0, 1, 2, \ldots$ there holds

$$\begin{aligned} x^{[0]}(0) &= x_0, \\ x^{[i+1]}(0) &= \frac{1}{i+1}F^{[i]}(0). \end{aligned} \tag{3}$$

This iterative procedure together with the Taylor arithmetic allows us to compute the Taylor expansion of x up to arbitrary order. Given already computed coefficients $x^{[0]}(0), \ldots, x^{[k]}(0)$ we can compute $F^{[0]}(0), \ldots, F^{[k]}(0)$. By (3) we have computed the $x^{[k+1]}$. Clearly, we can start this iterative procedure from $x^{[0]}(0) = x_0$.

3.3. Example.

Consider a differential equation given by

$$\begin{cases} \dot{x} = y(x^2 + y^2) \\ \dot{y} = -x(x^2 + y^2) \end{cases}$$

The system is solvable. For an initial condition $u_0 = (x_0, y_0)$ the solution is given by

$$\begin{aligned} x(t) &= r\cos(\alpha_0 - r^2 t), \\ y(t) &= r\sin(\alpha_0 - r^2 t), \end{aligned}$$

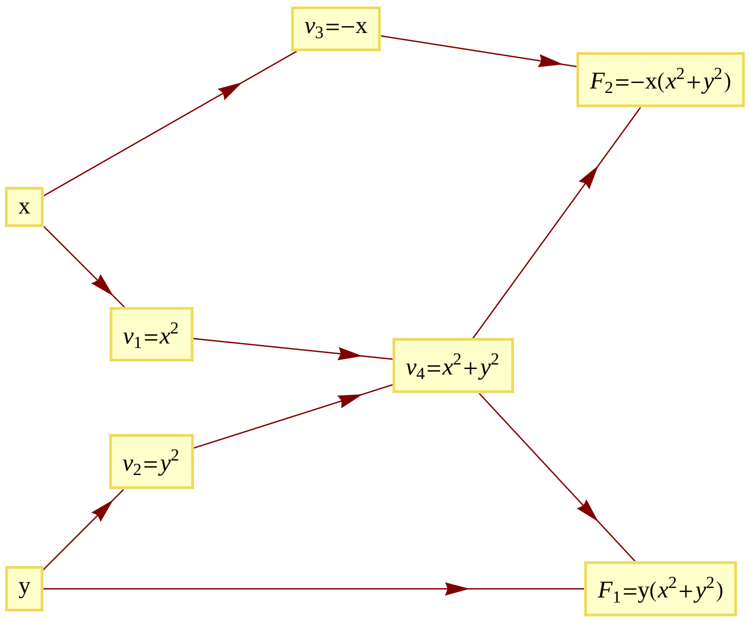
where $r = \sqrt{x_0^2 + y_0^2}$ and $\alpha_0 = \mathrm{Arg}(x_0 + iy_0)$.

Consider an initial condition $u_0 = (x_0, y_0) = (1, -1)$. The Taylor expansion of the solution at u_0 can be easily computed. We have

$$\begin{aligned} x(t) &= \sqrt{2}\cos(2t + \pi/4) = 1 - 2t - 2t^2 + \cdots, \\ y(t) &= -\sqrt{2}\sin(2t + \pi/4) = -1 - 2t + 2t^2 + \ldots. \end{aligned}$$

In the sequel we will see that the Taylor coefficients of $x(t)$ and $y(t)$ can be computed without any knowledge on the explicit solution and without computing derivatives of the vector field.

The vector field under consideration can be seen as a direct acyclic graph



On the graph we have defined intermediate variables v_i, $i = 1, \ldots, 4$. The v_i's are univariate functions of one variable t. Our goal is to compute all Taylor coefficients of x and y at zero. Therefore, in what follows we will skip the arguments of x, y and v_i's. The zero order coefficients of F can be computed by direct evaluation of the formula defining the vector field

$$\begin{aligned}
x^{[0]} &= x_0 = 1,\\
y^{[0]} &= y_0 = -1,\\
v_1^{[0]} &= x^{[0]} * x^{[0]} = 1,\\
v_2^{[0]} &= y^{[0]} * y^{[0]} = 1,\\
v_3^{[0]} &= -x^{[0]} = -1,\\
v_4^{[0]} &= v_1^{[0]} + v_2^{[0]} = 2,\\
F_1^{[0]} &= y^{[0]} * v_4^{[0]} = -2,\\
F_2^{[0]} &= v_3^{[0]} * v_4^{[0]} = -2.
\end{aligned}$$

According to (3), the last two numbers are the first Taylor coefficients of x and y, respectively. Hence

$$\begin{aligned}
x^{[1]} &= F_1^{[0]} = -2,\\
y^{[1]} &= F_2^{[0]} = -2.
\end{aligned}$$

Now we will compute in the first order Taylor coefficients of F_1 and F_2. Using Taylor arithmetic we obtain

$$\begin{aligned}
v_1^{[1]} &= 2 * x^{[0]} * x^{[1]} = -4,\\
v_2^{[1]} &= 2 * y^{[0]} * y^{[1]} = 4,\\
v_3^{[1]} &= -x^{[1]} = 2,\\
v_4^{[1]} &= v_1^{[1]} + v_2^{[1]} = 0,\\
F_1^{[1]} &= y^{[0]} * v_4^{[1]} + y^{[1]} * v_4^{[0]} = -4,\\
F_2^{[1]} &= v_3^{[0]} * v_4^{[1]} + v_3^{[1]} * v_4^{[0]} = 4.
\end{aligned}$$

Again using (3) we obtain

$$\begin{aligned}
x^{[2]} &= \frac{1}{2} F_1^{[1]} = -2,\\
y^{[2]} &= \frac{1}{2} F_2^{[1]} = 2.
\end{aligned}$$

Therefore, the beginning of the Taylor series of x and y are given by

$$\begin{aligned}
x(t) &= 1 - 2t - 2t^2 + \dots\\
y(t) &= -1 - 2t + 2t^2 + \dots
\end{aligned}$$

which agrees with what we got from the explicit solution.

Remarks:

- In order to compute Taylor coefficients of the solution to an ODE **we do not need** to compute **derivatives of the vector field!**; we only compute the Taylor coefficients of the univariate function $F = f \circ x$.
- **Repeating the above scheme one can compute the Taylor coefficients of the solution up to arbitrary order.**
- The implementation of the Taylor method for ODE's is easy by means of operators and functions overloading.
- We do not need to re-implement the Taylor method for different ODE's. The code is the same; only the formula defining the vector field must be specified. One can find efficient C++ implementations at [1, 2].
- The method is very fast. The number of multiplications grows quadratically with order of the Taylor expansion.

References

[1] CAPD – Computer Assisted Proofs in Dynamics group, http://capd.ii.uj.edu.pl.

[2] FADBAD++ – Flexible Automatic differentiation using templates and operator overloading in C++, http://www.fadbad.com.