Automatic Differentiation
Brian Heinold
Mount St. Mary’s University
Derivatives are easy. Why estimate them numerically?
Derivatives are easy. Why estimate them numerically?

- Useful for really complicated functions (especially ones defined by a program)
- Useful in some other methods, like the finite-element method for differential equations
The usual approach

Start with the definition of the derivative:

\[
\lim_{{h \to 0}} \frac{f(x + h) - f(x)}{h}
\]

Choosing a small value of \( h \) gives an estimate of \( f'(x) \).
The usual approach

Start with the definition of the derivative:

\[ \lim_{h \to 0} \frac{f(x + h) - f(x)}{h} \]

Choosing a small value of \( h \) gives an estimate of \( f'(x) \).

For example, if \( f(x) = \sin x \), then

\[ f'(1) \approx \frac{\sin(1 + .0001) - \sin(1)}{.0001} = .54026 \]

(Exact value is .54030...)
Mathematically, smaller values of $h$ should give closer estimates, but that’s not the case in practice.

<table>
<thead>
<tr>
<th>$h$</th>
<th>$\frac{(3 + h)^2 - 3^2}{h}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>6.1000000000000012</td>
</tr>
<tr>
<td>0.001</td>
<td>6.000999999999479</td>
</tr>
<tr>
<td>$10^{-5}$</td>
<td>6.000009999951316</td>
</tr>
<tr>
<td>$10^{-7}$</td>
<td>6.00000087880153</td>
</tr>
<tr>
<td>$10^{-9}$</td>
<td>6.00000496442226</td>
</tr>
<tr>
<td>$10^{-11}$</td>
<td>6.00000496442226</td>
</tr>
<tr>
<td>$10^{-13}$</td>
<td>6.004086117172844</td>
</tr>
<tr>
<td>$10^{-15}$</td>
<td>5.329070518200751</td>
</tr>
</tbody>
</table>
The first 30 digits of the floating-point representation of \((3 + h)^2\), where \(h = 10^{-13}\):

\[
9.0000000000000600408611717284657
\]

The last three “correct” digits are 6 0 0. Everything after that is an artifact of the floating-point representation.

When we subtract 9 from this and divide by \(10^{-13}\), all of the digits starting with the that 6 are “promoted” to the front, and we get 6.004086\ldots, which is only correct to the second decimal place.
More accurate formulas

There are more accurate formulas, such as

\[ f'(x) \approx \frac{f(x + h) - f(x - h)}{2h}, \]

\[ f'(x) \approx \frac{f(x - h) - 8f(x - h/2) + 8f(x + h/2) - f(x + h)}{6h}. \]
There are more accurate formulas, such as

\[ f'(x) \approx \frac{f(x + h) - f(x - h)}{2h}, \]

\[ f'(x) \approx \frac{f(x - h) - 8f(x - h/2) + 8f(x + h/2) - f(x + h)}{6h}. \]

But these still suffer from the same problem.
Taylor series expansion for $f(x + h)$:

$$f(x + h) = f(x) + f'(x)h + \frac{f''(x)}{2!}h^2 + \frac{f'''(x)}{3!}h^3 + \ldots.$$  

From this we get

$$f'(x) = \frac{f(x + h) - f(x)}{h} + \frac{f''(x)}{2!}h + \frac{f'''(x)}{3!}h^2 + \ldots.$$  

\[
\underbrace{\frac{f''(x)}{2!}h + \frac{f'''(x)}{3!}h^2 + \ldots}_{\text{Error}}
\]
Recall that imaginary numbers are defined by creating a new (nonreal) number \( i \) with the property \( i^2 = -1 \).

Let’s create a new (nonreal) number \( \epsilon \) with the property that \( \epsilon^2 = 0 \).

Note that \( \epsilon \) is not 0.

The set of dual numbers consists of all expressions of the form \( a + b\epsilon \), with \( a, b \in \mathbb{R} \).
Operations

- **Addition**: 
  \[(a + b\epsilon) \pm (c + d\epsilon) = (a \pm c) + (b \pm d)\epsilon\]

- **Multiplication**: 
  \[(a + b\epsilon)(c + d\epsilon) = ac + (ad + bc)\epsilon\]

- **Division**: (Multiply by the conjugate)
  \[
  \frac{a + b\epsilon}{c + d\epsilon} \cdot \frac{c - d\epsilon}{c - d\epsilon} = \frac{a}{c} + \frac{bc - ad}{c^2}\epsilon
  \]
Taylor series expansion for $f(x + \epsilon)$:

$$f(x + \epsilon) = f(x) + f'(x)\epsilon + \frac{f''(x)}{2!} \epsilon^2 + \frac{f'''(x)}{3!} \epsilon^3 + \ldots.$$

All of the higher order terms are 0, since $\epsilon^2$, $\epsilon^3$, etc. are all 0.
Taylor series expansion for $f(x + \epsilon)$:

$$f(x + \epsilon) = f(x) + f'(x)\epsilon + \frac{f''(x)}{2!}\epsilon^2 + \frac{f'''(x)}{3!}\epsilon^3 + \ldots.$$

All of the higher order terms are 0, since $\epsilon^2$, $\epsilon^3$, etc. are all 0. So the following equation is exact:

$$f(x + \epsilon) = f(x) + f'(x)\epsilon$$
Taylor series expansion for $f(x + \epsilon)$:

$$f(x + \epsilon) = f(x) + f'(x)\epsilon + \frac{f''(x)}{2!}\epsilon^2 + \frac{f'''(x)}{3!}\epsilon^3 + \ldots.$$ 

All of the higher order terms are 0, since $\epsilon^2$, $\epsilon^3$, etc. are all 0. So the following equation is exact:

$$f(x + \epsilon) = f(x) + f'(x)\epsilon$$

If we solve this for the derivative, we get

$$f'(x)\epsilon = f(x) - f(x + \epsilon).$$

So the exact value of the derivative of $f$ at a real number $x$ is gotten from the dual component of $f(x) - f(x + \epsilon)$. 
A similar Taylor series argument gives us

\[ f(a + b\epsilon) = f(a) + b f'(a)\epsilon, \]
How do we evaluate functions of dual numbers?

A similar Taylor series argument gives us

\[ f(a + b\epsilon) = f(a) + bf'(a)\epsilon, \]

For instance,

\[ \sin(a + b\epsilon) = \sin(a) + b\cos(a)\epsilon. \]
How do we evaluate functions of dual numbers?

A similar Taylor series argument gives us

\[ f(a + b\epsilon) = f(a) + bf'(a)\epsilon, \]

For instance,

\[ \sin(a + b\epsilon) = \sin(a) + b\cos(a)\epsilon. \]

In particular,

\[ \sin \left( \frac{\pi}{3} + 3\epsilon \right) = \frac{1}{2} + \frac{3}{2}\epsilon. \]
Product, quotient, chain rules

Product, quotient and chain rules are easily shown:

Chain rule:
\[ f(g(x + \epsilon)) = f(g(x) + g'(x)\epsilon) = f(g(x)) + g'(x)f'(g(x))\epsilon. \]

Product rule:
\[ (fg)(x + \epsilon) = f(x + \epsilon)g(x + \epsilon) = (f(x) + f'(x)\epsilon)(g(x) + g'(x)\epsilon) = f(x)g(x) + (f'(x)g(x) + f(x)g'(x))\epsilon. \]
Product, quotient and chain rules are easily shown:

Chain rule:

\[ f(g(x + \epsilon)) = f(g(x) + g'(x)\epsilon) = f(g(x)) + g'(x)f'(g(x))\epsilon. \]
Product, quotient, and chain rules are easily shown:

Chain rule:

\[ f(g(x + \epsilon)) = f(g(x) + g'(x)\epsilon) = f(g(x)) + g'(x)f'(g(x))\epsilon. \]

Product rule:

\[ (fg)(x + \epsilon) = f(x + \epsilon)g(x + \epsilon) \]
\[ = (f(x) + f'(x)\epsilon)(g(x) + g'(x)\epsilon) \]
\[ = f(x)g(x) + (f'(x)g(x) + f(x)g'(x))\epsilon \]
So all we have to do is program in the rules for elementary operations and some common functions and everything will just work.
class Dual:
    def __init__(self, a, b):
        self.a = a
        self.b = b

    def __add__(self, y):
        if type(y) == int or type(y) == float:
            return Dual(self.a + y, self.b)
        else:
            return Dual(y.a+self.a, y.b+self.b)

    def __radd__(self, y):
        return self.__add__(y)

    def __mul__(self, y):
        if type(y) == int or type(y) == float:
            return Dual(self.a*y, self.b*y)
        else:
            return Dual(y.a*self.a, y.b*self.a + y.a*self.b)

    def __rmul__(self, y):
        return self.__mul__(y)

    def __pow__(self, e):
        return Dual(self.a ** e, self.b*e*self.a ** (e-1))
More of the Python implementation

```python
def create_func(f, deriv):
    return lambda D: Dual(f(D.a), D.b*deriv(D.a)) if type(D)==Dual else f(D)

sin = create_func(math.sin, math.cos)
exp = create_func(math.exp, math.exp)
ln = create_func(math.log, lambda x:1/x)
```
def autoderiv(s, x):
    f = eval('lambda x: ' + s.replace("^", "**")
    return (f(Dual(x,1))-f(Dual(x,0))).b
def autoderiv(s, x):
    f = eval('lambda x: ' + s.replace('^', '**'))
    return (f(Dual(x,1))-f(Dual(x,0))).b

>>> autoderiv("sin(x)",1)
0.5403023058681397
>>> cos(1)
0.5403023058681397
def autoderiv(s, x):
    f = eval('lambda x: ' + s.replace('^', '**'))
    return (f(Dual(x,1)) - f(Dual(x,0))).b

>>> autoderiv("sin(x)", 1)
0.5403023058681397
>>> cos(1)
0.5403023058681397

>>> autoderiv("1+2*x+3*x^2", 3)
20
>>> 2 + 6*3
20
def autoderiv(s, x):
    f = eval('lambda x: ' + s.replace("^", "**")
    return (f(Dual(x,1)) - f(Dual(x,0))).b

>>> autoderiv("sin(x)",1)
0.5403023058681397
>>> cos(1)
0.5403023058681397

>>> autoderiv("1+2*x+3*x^2", 3)
20
>>> 2 + 6*3
20

>>> autoderiv("sin(exp(x^2))*ln(x)", 3.24)
254566.1653152972
>>> cos(exp(3.24**2))*exp(3.24**2)*2*3.24*ln(3.24) + sin(exp(3.24**2))/3.24
254566.16531529723
More about dual numbers

- Quotient of $\mathbb{R}[x]$ by $(x^2)$.
- Show up in algebraic geometry
- Show up in modern physics
More about dual numbers

- Quotient of $\mathbb{R}[x]$ by $(x^2)$.
- Show up in algebraic geometry
- Show up in modern physics

Other uses for automatic differentiation

- Also useful for functions defined by computer programs
- Can be applied to higher derivatives
- Can be applied to functions from $\mathbb{R}^n$ to $\mathbb{R}^m$