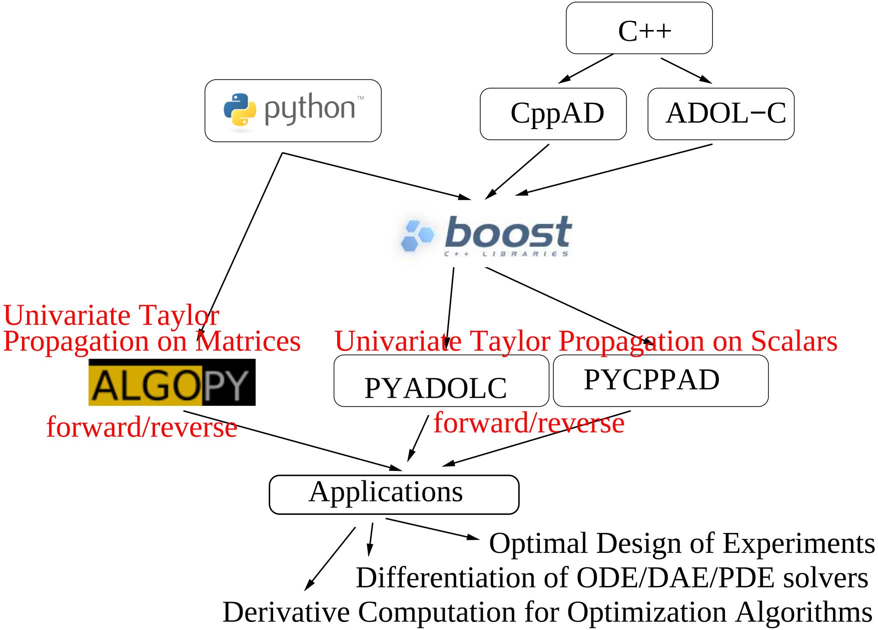
Intro to Algorithmic Differentiation

Euro SciPy 2009, Leipzig

Sebastian F. Walter

HU Berlin

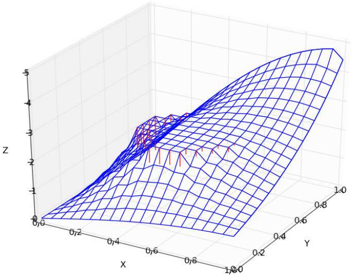
Thursday, 25'th July 2009


C++
python
CppAD
ADOL−C
boost
C++ LIBRARIES
Univariate Taylor
Propagation on Matrices
ALGOPY
forward/reverse
Univariate Taylor Propagation on Scalars
PYADOLC
PYCPPAD
forward/reverse
Applications
Optimal Design of Experiments
Differentiation of ODE/DAE/PDE solvers
Derivative Computation for Optimization Algorithms

Example: Minimal Surface Problem with PYADOLC

- Minimal Surface Problem:

$$
\begin{aligned}
u : S \subset [0,1] \times [0,1] \quad &\rightarrow \quad R \quad u \in C^1(S) \\
O(u) \quad &= \quad \int_0^1 \int_0^1 \sqrt{1 + \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2} \mathrm{d}x\mathrm{d}y \\
&\approx \quad \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} O_{ij}(u) \\
O_{ij}(u) \quad &:= \quad h^2 \left[1 + \frac{(u_{i+1,j+1} - u_{i,j})^2 + (u_{i,j+1} - u_{i+1,j})^2}{4} \right]
\end{aligned}
$$



Nonlinear Program with Inequality Box Constraints:

$$
\begin{aligned}
R^{m \times m} \ni u_* \quad &= \quad \mathrm{argmin}_u O(u) \\
\text{s.t.} \quad 0 \quad &\leq \quad u_{ij} \quad \forall (i,j) \in \text{Cylinder set}
\end{aligned}
$$

Example (Cont): Slicing, Broadcasting, Loops: works!

```
import numpy
from adolc import *
def O_tilde(u):
        M = numpy.shape(u)[0]
        h = 1./(M-1)
        return M**2*h**2 +
          numpy.sum(0.25*( (u[1:,1:] - u[0:-1,0:-1])**2
            + (u[1:,0:-1] - u[0:-1, 1:])**2))
M = 5
h = 1./M
u = numpy.zeros((M,M),dtype=float)
u[0,:]= [numpy.sin(numpy.pi*j*h/2.) for j in range(M)]
u[-1,:] = [ numpy.exp(numpy.pi/2) * numpy.sin(numpy.pi * j * h / 2.) for j
u[:,0]= 0
u[:,-1]= [ numpy.exp(i*h*numpy.pi/2.) for i in range(M)]
trace_on(1)
au = adouble(u)
independent(au)
ay = O_tilde(au)
dependent(ay)
trace_off()
ru = numpy.ravel(u)
rg = gradient(1,ru)
rH = hessian(1,ru)
```

- machine EPS $\approx 10^{-16}$ for 64bit IEEE-754 floats

$$\frac{\mathrm{d}^3 f}{\mathrm{d}x^3} = \frac{f(x+3h) - 3f(x+2h) + 3f(x+h) - f(x)}{h^3} + \mathcal{O}(h)$$

- cancellation of the addition in numerator/denominator

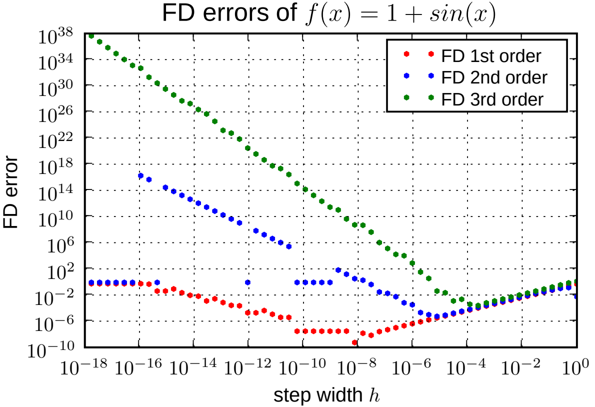


Figure: Unsuitable for Higher Order Derivatives

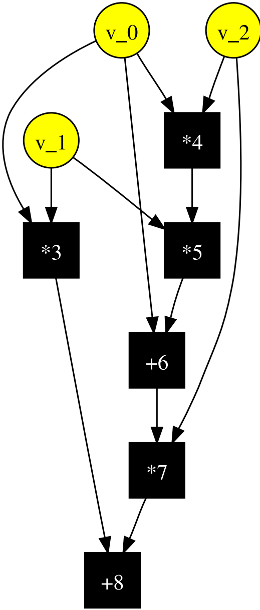
What is wrong with Symbolic Differentiation

- does not work well with deep recursions (e.g. for-loops)
- Example: Compute sensitivity of a ball on an elliptic pool table w.r.t. initial angle...
- gradient of a function $f : \mathbb{R}^N \to \mathbb{R}$ are N functions $\frac{\partial f}{\partial x_n}$ for $n = 1, \dots, N$.

```
from sympy import *
x,y,z = symbols('xyz')
f = x*y*z
g = [f.diff(x), f.diff(y), f.diff(z)]
print 'g=',g
>>>g= [y*z x*z x*y]
```

Algorithmic Differentiation: Overview

- AD is **not** Finite Differences and **not** Symbolic Differentiation
- computes derivatives at machine precision
- a.s. faster than FD or symbolic derivs
- operates on computational graph representation: `f = (x * y) + z*(x+y*(x*z))`



Reverse Mode of AD

reverse == traversing computational graph in reverse order

For functions $f : R^N \rightarrow R$ the number of operations OPS is independent of N:

$\mathrm{OPS}(\nabla f) \leq 4\mathrm{OPS}(f)$

Problem: Memory consumption

$\mathrm{MEM}(\nabla f) \approx \mathrm{OPS}(f)$

- **PYADOLC, wrapper for ADOL-C (C++)**, S.F. Walter, http://github.com/b45ch1/pyadolc
- **PYCPPAD: wrapper for CppAD (C++)**, B. Bell and S.F. Walter, http://github.com/b45ch1/pycppad
- **ALGOPY: AD on matrix valued functions** , S.F. Walter, http://github.com/b45ch1/algopy
- **Evaluating Derivatives, Second Edition** Andreas Griewank, Andrea Walther, SIAM, 2008