Computational Differentiation in Global Optimization Software

This talk will present:

1. The general problem framework in optimization software.

2. An example of effectiveness of computational differentiation in optimization packages.

3. Particular importance of computational differentiation verified global optimization.

4. A simple example of use of our package.

5. Outline of the present structure.


7. Relationships with other work presented here.
The General Problem Framework

*Traditional Optimization Packages*

- Traditional non-verified global optimization algorithms require *objective function, gradient, and Hessian matrix*.

- Supplying these without computational differentiation was been done with *user-supplied derivatives, finite-difference approximations, or symbolic manipulation packages*.
Traditional Optimization

*Disadvantages without Computational Differentiation*

**User-supplied derivatives** can take excessive amounts of labor to get right.

**Finite-difference approximations** can be both numerically inaccurate and take more computer resources than necessary.

**Symbolic manipulation packages** are subject to *expression swell* that makes the results impractical to use.

Although only order-2 derivatives are used (in contrast to other problems addressed here), complicated functions and many variables make the above disadvantages real.
Computational Differentiation in Traditional Optimization

A Successful Example

• Computational differentiation can numerically more accurate, in addition to being more labor and computation efficient.

• An example is *Network Enabled Optimization System* (NEOS) server at [www.mcs.anl.gov/otc/Server/](http://www.mcs.anl.gov/otc/Server/)

• Users need only submit subroutines in Fortran or C, and the package uses the computational differentiators ADIFOR or ADOLC to compute derivatives.

• The system runs the solver, and the answer to the problem is sent back through the WWW.