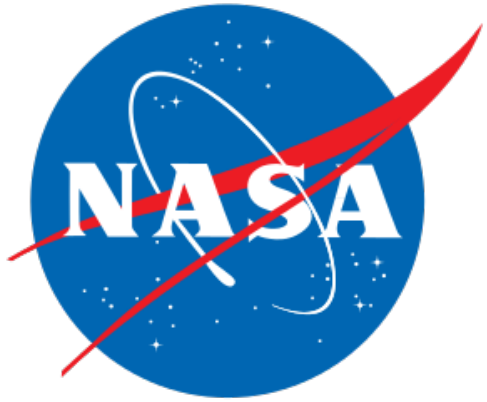
OpenMDAO

Efficient Multidisciplinary Optimization

Justin Gray

October 12th, 2017



Our Goal: make optimization fast and cheap so you can run more of it!

- Specialize in gradient-based optimization using analytic derivatives
- Automatic computation of total-derivatives across multidisciplinary models
- MPI based distributed memory data storage with support for high-fidelity analyses

Open-source project with code contributions from US and Europe

- Primarily developed by NASA Glenn Research Center
- Released under permissive open-source license: Apache v2.0
- **USA Contributions:** National Renewable Energy Laboratory, Purdue University, University of Michigan
  Rensselaer Polytechnic Institute, Metamorph Inc.
- **Europe Contributions:** DTU, ONERA

A long time ago ... in 2008

(2008) NASA realized that designing unconventional aircraft would need something "different"

- Leverage state-of-the-art MDO methods
- Build fully coupled models
- Integrate high-fidelity analyses into those models

(2010) OpenMDAO 0.1: Split "workflow" and "dataflow"

The good:

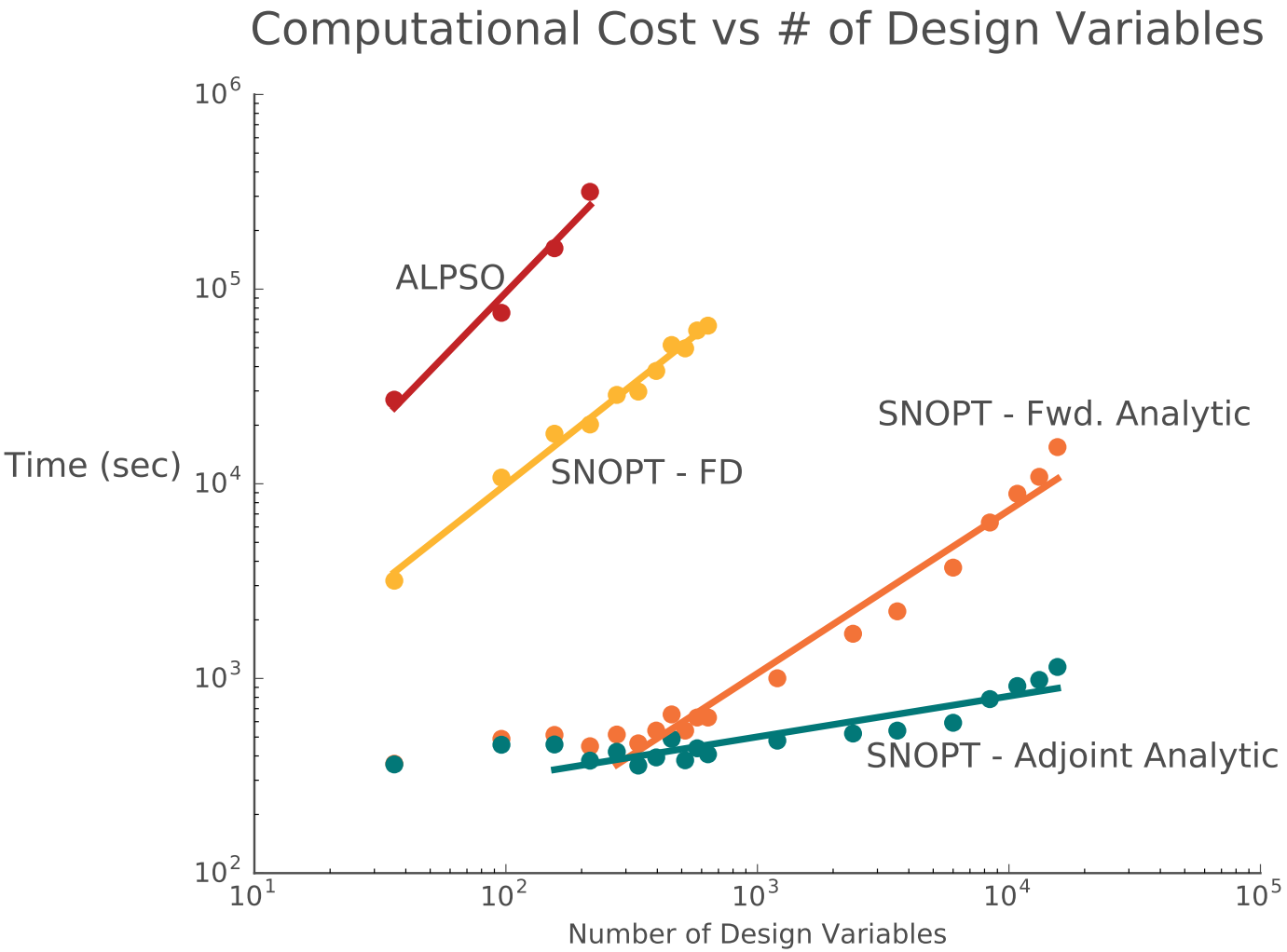
- Support for MDAO architectures: IDF, MDF, CO, BLISS
- Open-source MDAO framework

The bad:

- Serial execution
- Difficult installation process

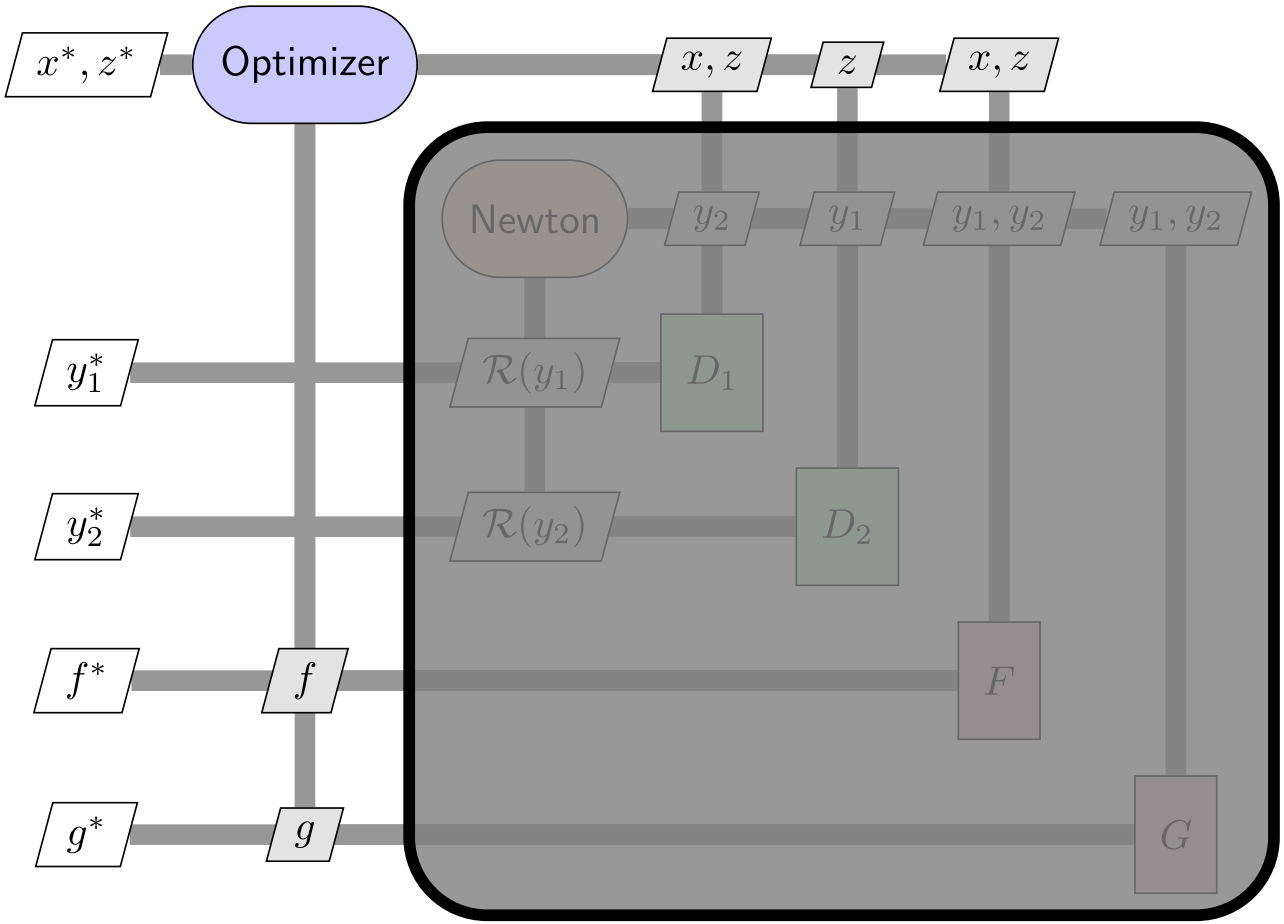
But what about high-fidelity??

We needed analytic derivatives to run efficient optimizations with high fidelity



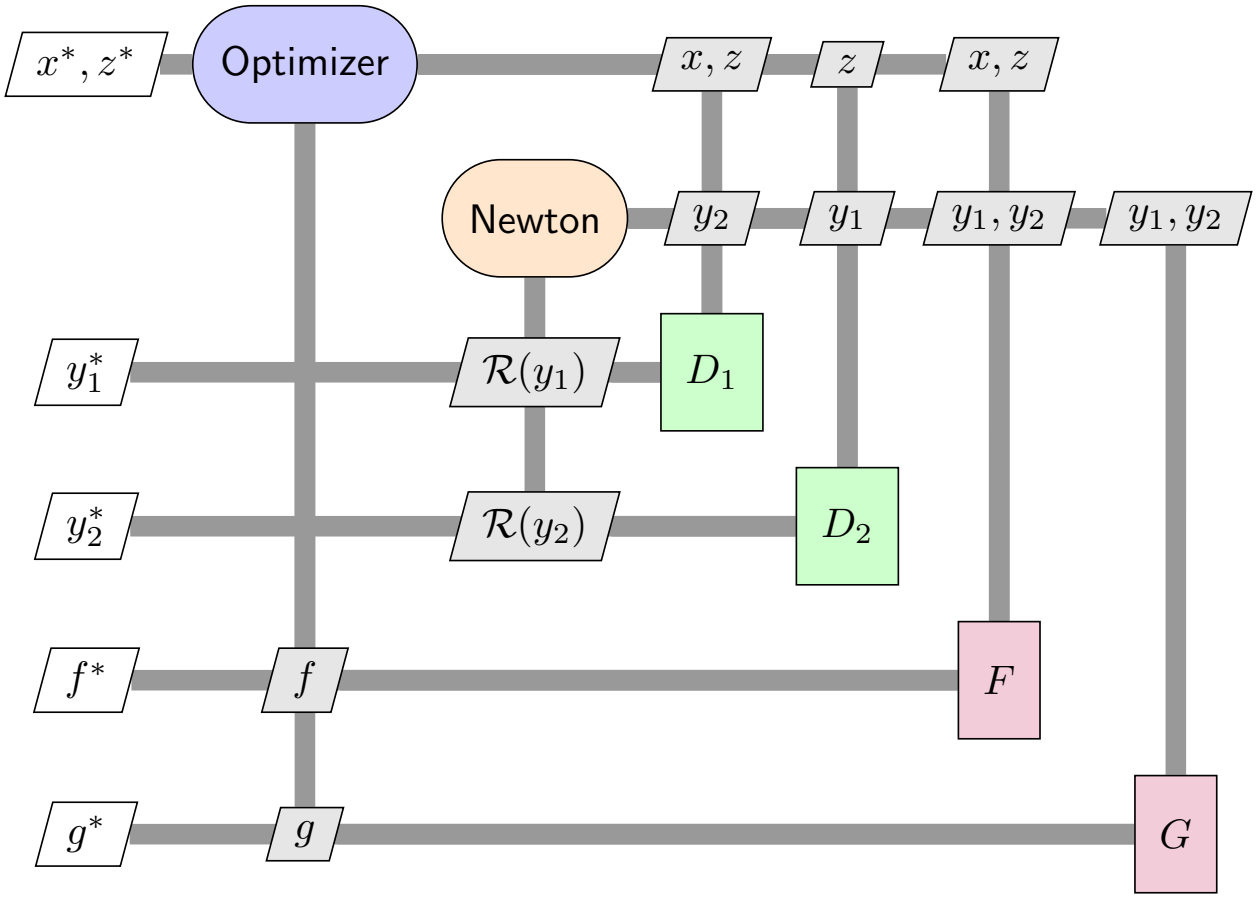
Computing total derivatives across a multidisciplinary model can be challenging

Need to compute $\frac{df}{dx}$, $\frac{df}{dz}$, etc.,



Computing total derivatives across a multidisciplinary model can be challenging

but you only know $\frac{\partial \mathcal{R}(y_1)}{\partial y_2}$, $\frac{\partial \mathcal{R}(y_2)}{\partial y_1}$, $\frac{\partial f}{\partial y_2}$, $\frac{\partial f}{\partial y_1}$, etc.



(2012) John Hwang and Joaquim Martins developed the Unified Derivative Equations

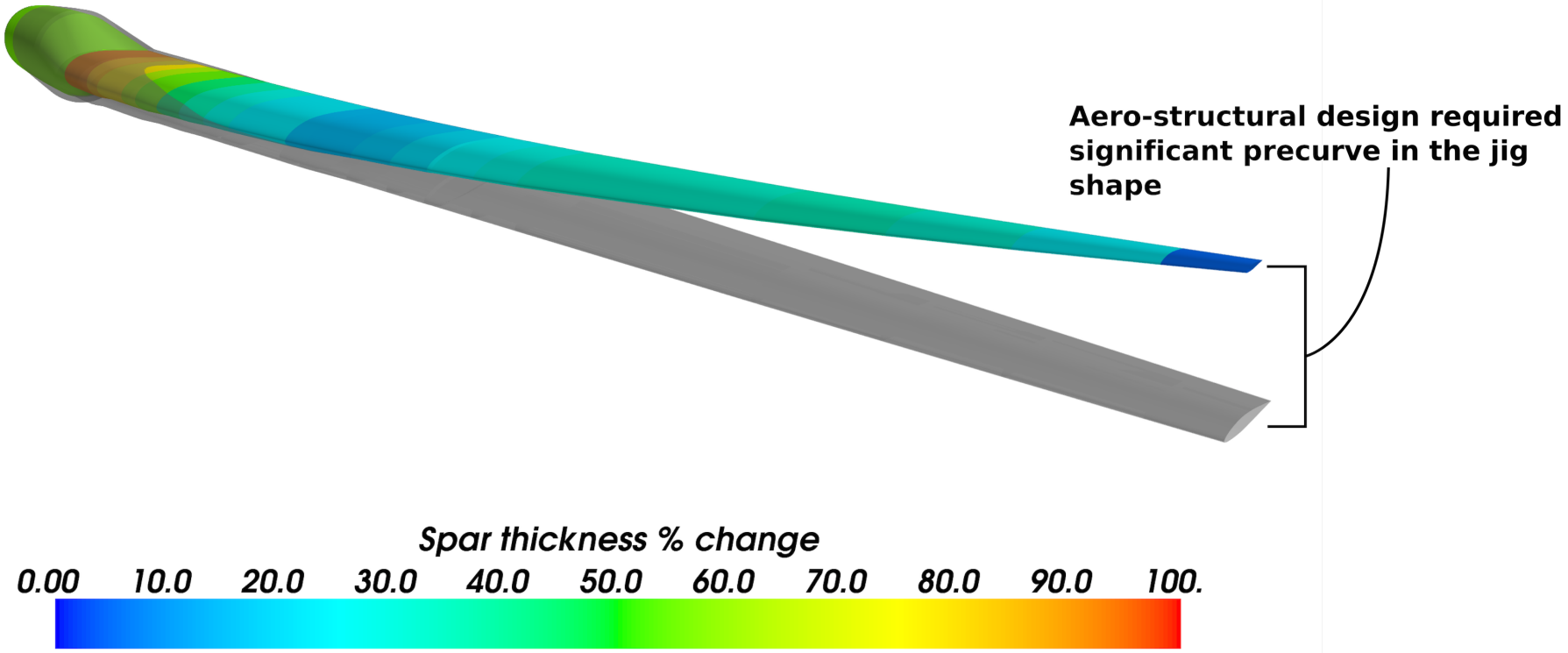
$$\underbrace{\left[\frac{\partial R}{\partial u}\right]}_{partials}\overbrace{\left[\frac{du}{dr}\right]}^{totals} = [\mathcal{I}]$$

We know the partials, so we can solve this linear system to compute total derivatives across any model

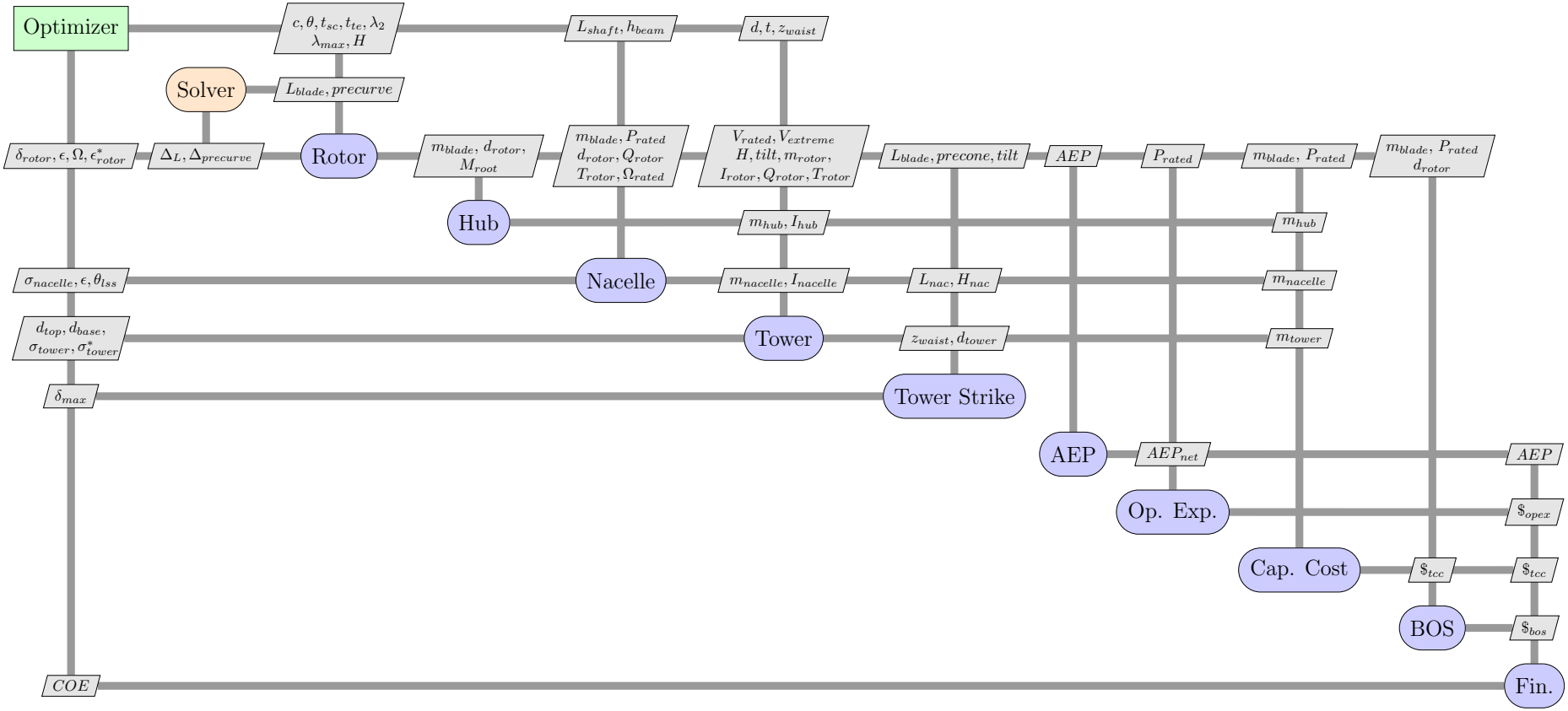
(2012) OpenMDAO 0.2, now with analytic derivatives!

- Still serial
- Same user API as 0.1
- A dense matrix design and direct factorization to solve $\left[\frac{\partial R}{\partial u}\right]\left[\frac{du}{dr}\right] = [\mathcal{I}]$

Application: Low fidelity aerostructural wind turbine blade design

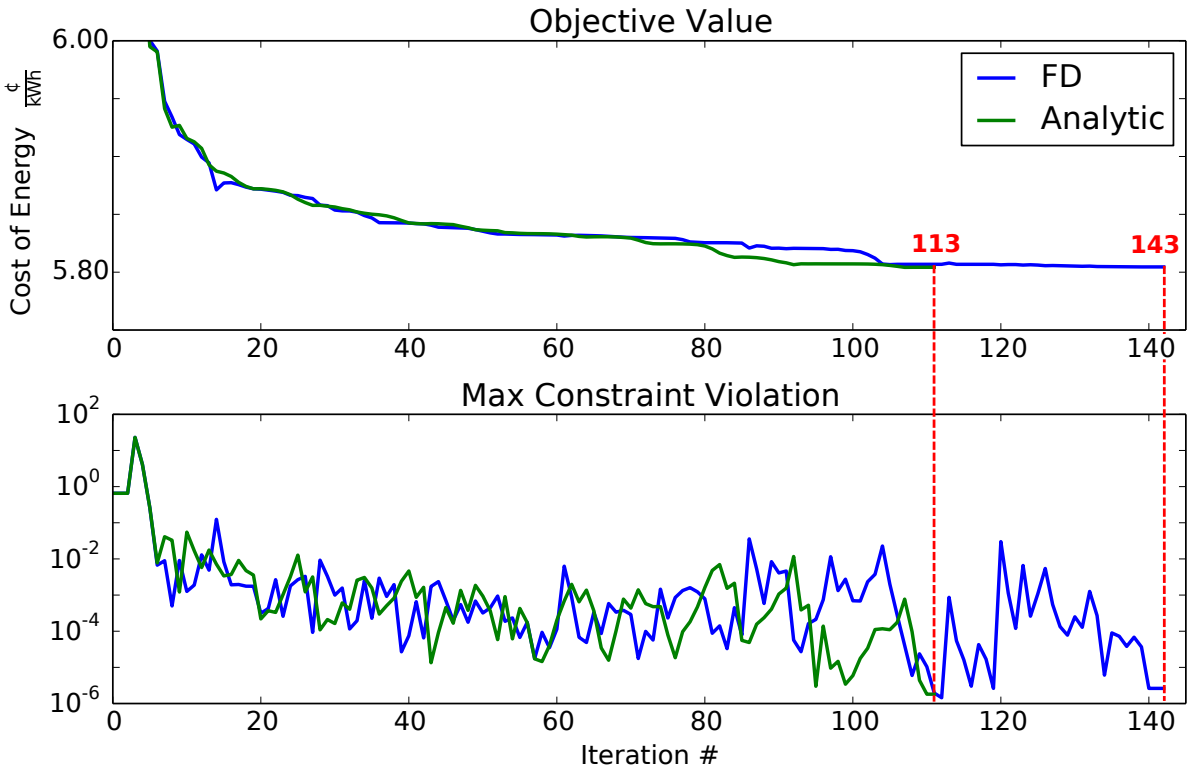


10 engineering disciplines with aerostructural coupling in the rotor design



Optimization was 5x faster with analytic derivatives

	Finite-Difference	Analytic
Objective (COE in ¢/kWh)	5.8045	5.8042
Max Constraint Violation	2.62×10^{-6}	1.81×10^{-6}
# Major Iterations	143	113
Time Per Major Iteration (minutes)	2.27	0.59
Total Run Time (hours)	**5.43**	**1.11**



But what about high-fidelity?????

OpenMDAO 0.2 was not able to scale up to support high-fidelity

- Serial execution, no MPI support
- Simplistic linear solver and dense matrix implementation

2013-2014 John Hwang developed a Modular Analysis and Unified Derivatives (MAUD)

collection of new algorithms and data structures
to efficiently solve $\left[\frac{\partial R}{\partial u}\right]\left[\frac{du}{dr}\right] = [\mathcal{I}]$ in parallel

Several important applications

- Small satellite design with over 25000 design variables
- Optimizing aircraft trajectories with adjoint derivatives
- Coupled aircraft allocation-mission-design
  including **high-fidelity aerodynamics**

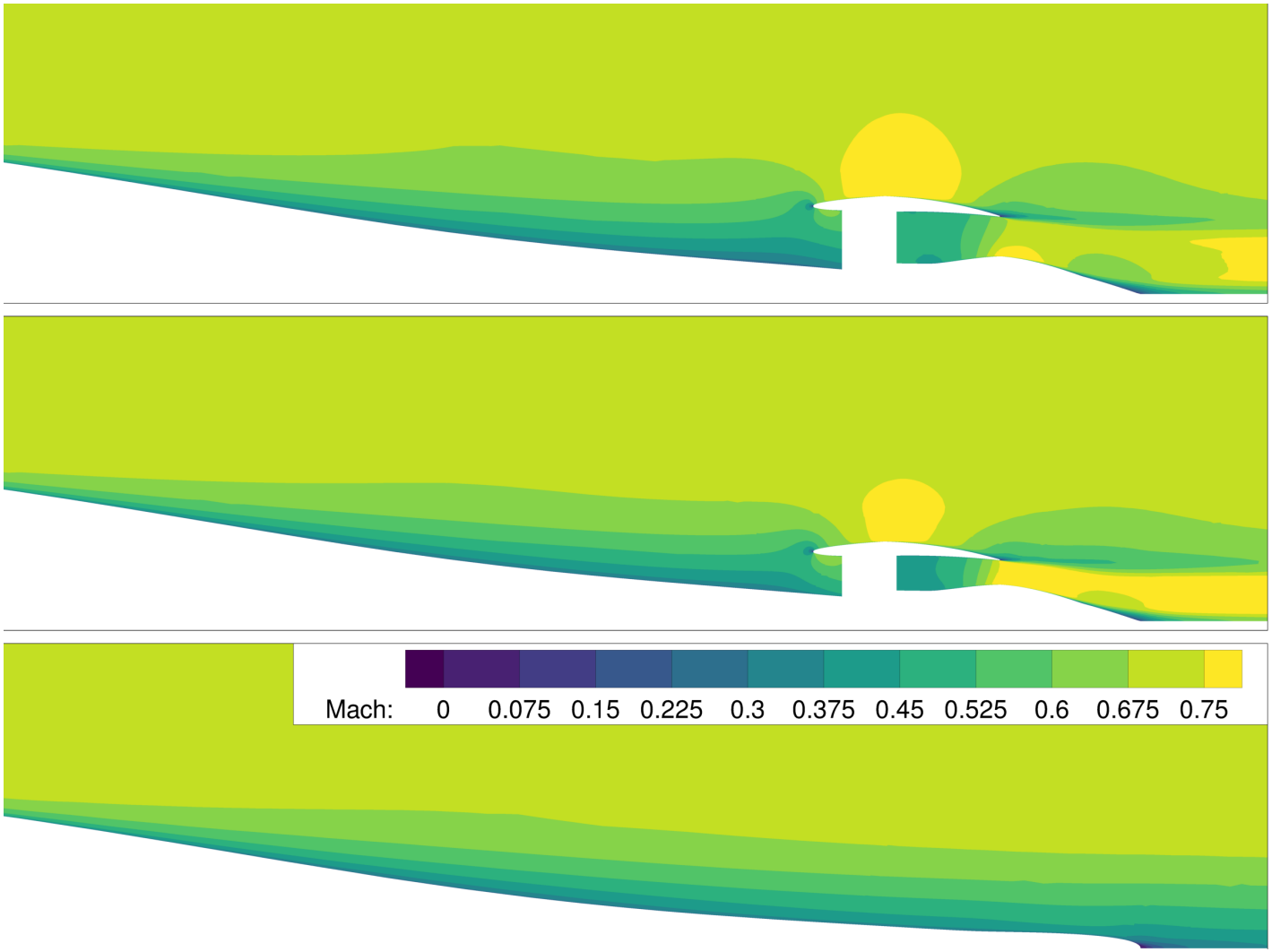
high-fidelity!!

(2015) OpenMDAO 1.0, now with high fidelity!

- ground-up rewrite of the framework
- MPI parallelism
- Distributed sparse data storage
- Sophisticated nonlinear and linear solver design
- Whole new user API

Application: Coupled aero-propulsive analysis of a tail-cone thruster

The fully coupled model demonstrates strong aero-propulsive coupling



FPR = 1.2

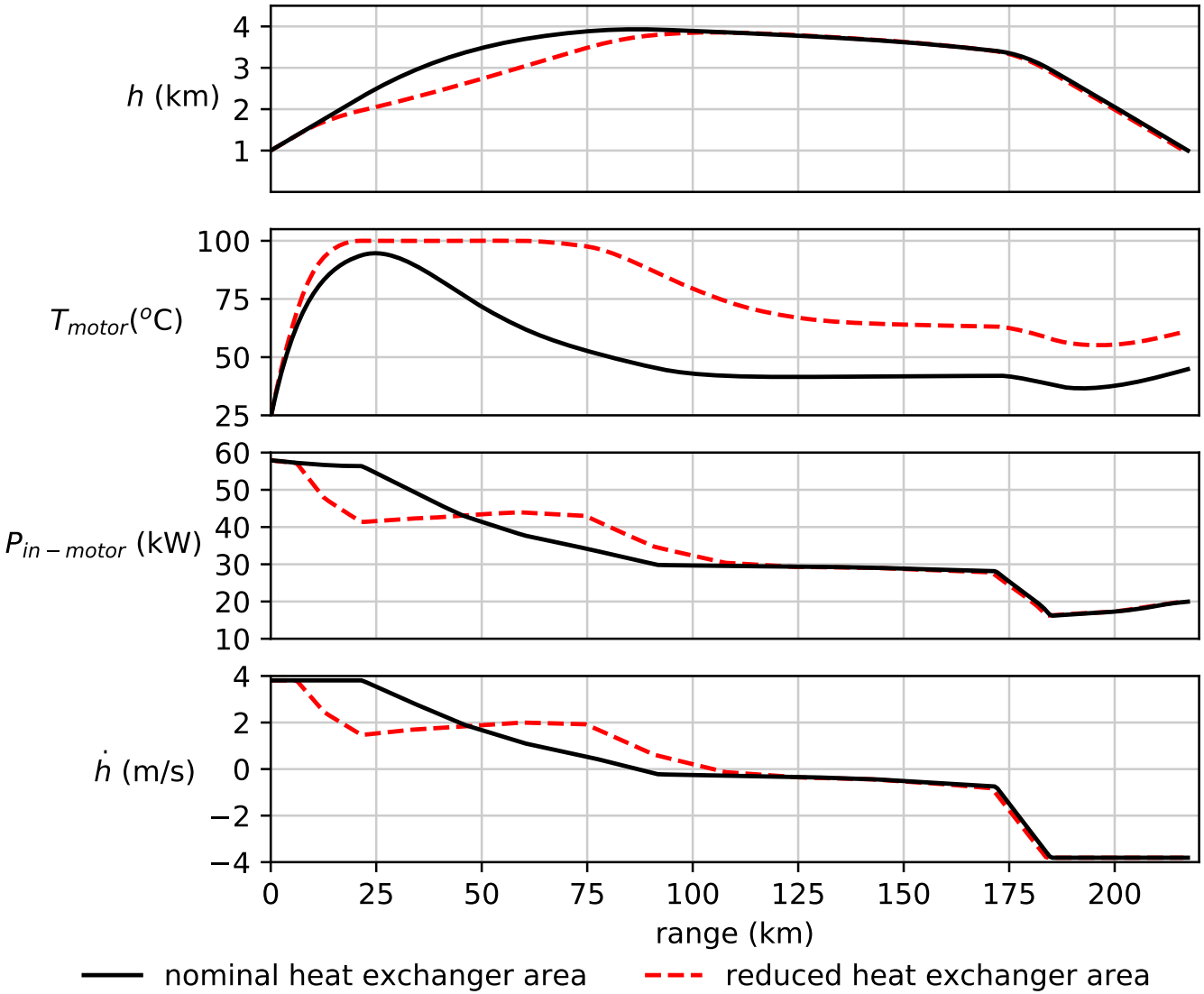
FPR = 1.35

baseline

Application: Optimal trajectory with integrated thermal modeling for X-57 Maxwell

- All-electric, distributed propulsion experimental testbed.
- Reduce power required to cruise via high-aspect ratio wings sized for cruise
- Use the wing motors to generate high-lift needed for take/off landing

Trajectory optimization retains performance while respecting thermal limits



1.0 was a major improvement, but it still had some weaknesses

- The sparse implementation was great for high-fidelity codes, but slow for low fidelity models
- TCT model worked well
- Trajectory optimization was too slow!

We want it all:
High and low fidelity

(2017) OpenMDAO 2.0, now good for low and high fidelity uses

- Total rewrite of the framework again!
- Minor updates to the API
- Now support both sparse and dense linear solvers
- Much more efficient for mixed-fidelity work

Analytic derivatives are the
key to integrating high-fidelity codes

Problem scale deeply impacts
the framework design!

Don't be afraid to throw away
code and start over