SAND2016-11221C

Optimization Modeling with PYOMO

Algebraic Optimization

- **Explicit declarative (algebraic) modeling**
  - Equality or inequality constraints
  - Linear or nonlinear equations
  - Continuous or discrete decision variables
- **Solve problems to provable local or global optimality**
- **For nonlinear problems we use exact first and second derivatives (obtained using automatic differentiation)**
- **Algebraic optimization is <u>not:</u>**
  - Derivative-free (blackbox) optimization
  - Heuristic-based solution techniques (genetic algorithm, simulated annealing, etc.)

Pyomo Overview

Idea: a Pythonic framework for formulating optimization models

- Provide a natural syntax to describe mathematical models
- Formulate large models with a concise syntax
- Separate modeling and data declarations
- Enable data import and export in commonly used formats

Highlights:

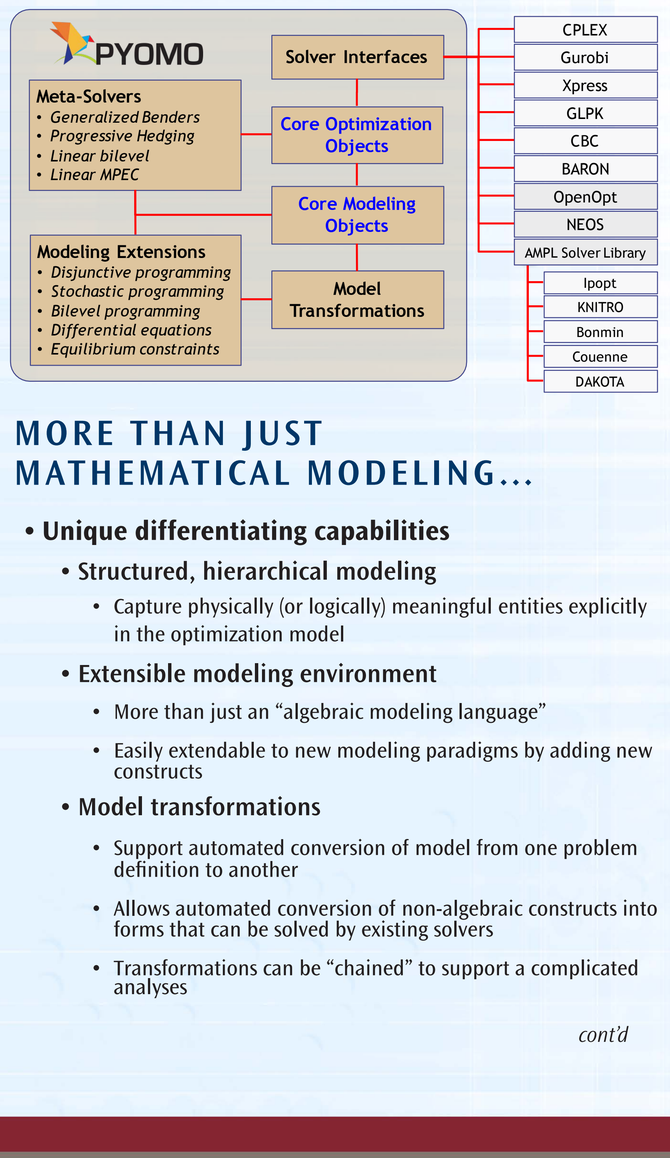
- Python provides a clean, intuitive syntax
- Python scripts provide a flexible context forexploring the structure of Pyomo models
- Explicitly represent model algebra

```
# simple.py
from pyomo.environ import *

M = ConcreteModel()
M.x1 = Var()
M.x2 = Var(bounds=(-1,1))
M.x3 = Var(bounds=(1,2))
M.o  = Objective(
          expr=M.x1**2 + (M.x2*M.x3)**4 + \
               M.x1*M.x3 + \
               M.x2*sin(M.x1+M.x3) + M.x2)

model = M
```

Pyomo at a Glance



MORE THAN JUST MATHEMATICAL MODELING...

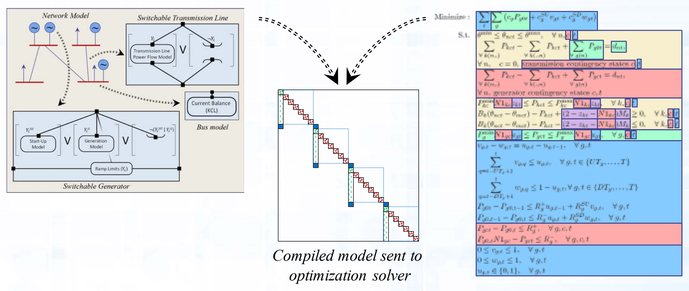
- **Unique differentiating capabilities**
  - Structured, hierarchical modeling
    - Capture physically (or logically) meaningful entities explicitly in the optimization model
  - Extensible modeling environment
    - More than just an "algebraic modeling language"
    - Easily extendable to new modeling paradigms by adding new constructs
  - Model transformations
    - Support automated conversion of model from one problem definition to another
    - Allows automated conversion of non-algebraic constructs into forms that can be solved by existing solvers
    - Transformations can be "chained" to support a complicated analyses

cont'd

- **Advanced solution strategies**
  - Because Pyomo is embedded in Python, users have the full power of a modern programming language at their fingertips
  - Powerful framework to support the rapid development of new optimization algorithms and custom solution strategies
  - Blurring the traditional distinction between "optimization modeling environment" and "optimization solver"

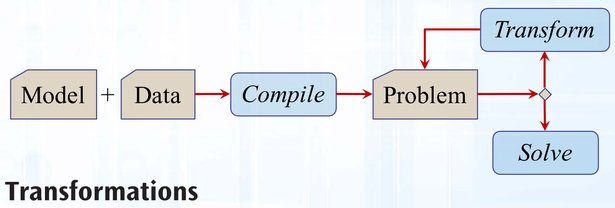
Composable, structured models

- **Object-based model**
  - Explicit structure
  - High-level constructs for more intuitive modeling
- **Standard algebraic model**
  - Flat representation
  - Implicit structure



Compiled model sent to optimization solver

Transformation-centric workflows



- **Transformations**
  - **Project from one problem space to another**
  - **Standardize common reformulations or approximations**
  - **Convert "unoptimizable" modeling constructs into equivalent optimizable forms**
    - differential equations, disjunctions, complementarities, bilevel models, etc.
  - **Separate model expression from solution approach**
  - **Reduce errors due to manual implementation**

DIFFERENT WAYS TO FORMULATE *abs()*

$$a = b + c,\quad b \le M \cdot y,\quad c \le M(1-y),\quad x - 3 = c - b,\quad b \ge 0,\quad c \ge 0,\quad y \in \{0,1\}$$

$$a = \sqrt{(x-3)^2 + \epsilon}$$

$$a = abs(x-3)$$

$$a \ge x - 3,\quad a \ge 3 - x$$

$$a = b + c,\quad x - 3 = c - b,\quad b \ge 0 \perp c \ge 0$$

$$a = \frac{2(x-3)}{1 + e^{-\frac{x-3}{h}}} - x + 3$$

If we *mean* "$a = abs(x-3)$",
why don't we *write* that in our models???

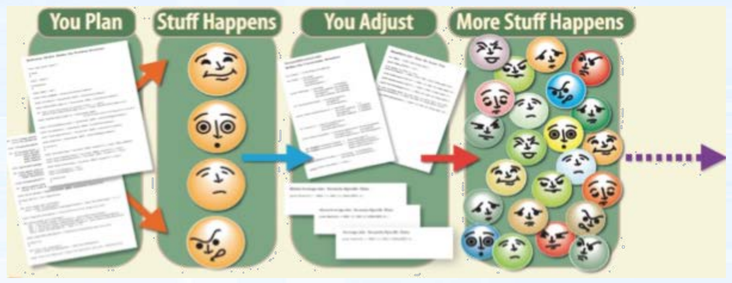
A TRANSFORMATION-CENTRIC VIEW OF *abs()*

Chaining transformations

$$f = abs(x) \Rightarrow \begin{array}{l} f = x^+ + x^- \\ x = x^+ - x^- \\ x^+ \ge 0 \perp x^- \ge 0 \end{array} \Rightarrow \begin{array}{l} f = x^+ + x^- \\ x = x^+ - x^- \\ \begin{bmatrix} Y \\ x^- = 0 \end{bmatrix} \vee \begin{bmatrix} \neg Y \\ x^+ = 0 \end{bmatrix} \\ x^+ \ge 0, x^- \ge 0 \end{array} \Rightarrow \begin{array}{l} f = x^+ + x^- \\ x = x^+ - x^- \\ x^- \le My \\ x^- \le M(1-y) \\ x^+ \ge 0, x^- \ge 0 \end{array}$$

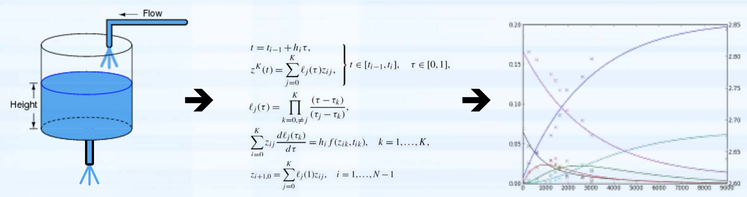
```
model = ConcreteModel()
# […]
TransformationFactory("abs.complements").apply_to(model)
TransformationFactory("mpec.disjunctive").apply_to(model)
TransformationFactory("gdp.bigm").apply_to(model)
```

Stochastic Programming



- **Multi-stage planning for uncertain environments**
  - Continuous / discrete decisions, linear / nonlinear models
  - Deterministic equivalent
  - Scenario-based and stage-based decomposition algorithms
  - Supports serial, SMP workstation, HPC cluster environments

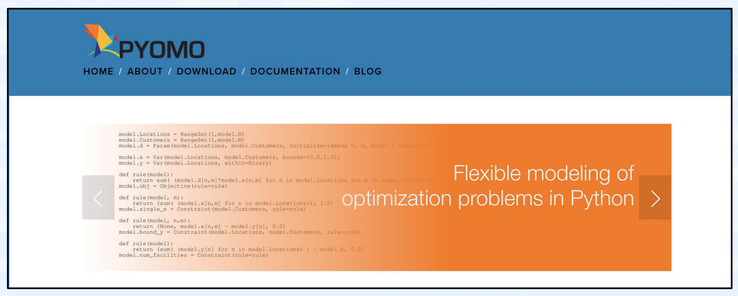
Scalable optimization of dynamic systems



- **Rapid development of dynamic models**
  - Directly express dynamics in a natural form (differential equations)
  - Automatic conversion to a large-scale nonlinear optimization problem
  - Solution using state-of-the-art serial solvers; scalable parallel decomposition strategies
  - Compatible with other Pyomo extensions (e.g., stochastic dynamic problems)

Open development model

- **Pyomo homepage (www.pyomo.org)**



- **Public development process (hosted on GitHub)**
- **Unrestrictive, commercial-friendly licensing (3-clause BSD)**
- **Large user base**
  - Compatible with other Pyomo extensions (e.g., stochastic dynamic problems)
- **Key funding for Pyomo development provided by ASCR and FE**

U.S. DEPARTMENT OF ENERGY
National Nuclear Security Administration

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.
SANDxxxx-xxxx Sandia Creative Group 284-2905

Sandia National Laboratories