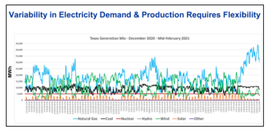
SAND2022-12851D

Dynamic Model Convergence, Reliability, and Diagnostics

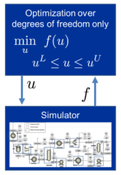
Bethany Nicholson[a], John Siirola[a], Robert Parker[b], Lorenz Biegler[b], John Eslick[cd], Douglas Allan[cd], Stephen Zitney[ce]
[a] Sandia National Laboratories, [b] Carnegie Mellon University, [c] National Energy Technology Laboratory, [d] NETL Support Contractor, [e] West Virginia University

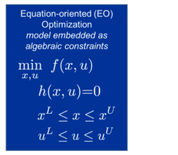
Motivation

Current and future energy systems must be able to operate more dynamically



Derivative-free ("black-box") optimization (DFO)
~ 100-1000 simulations



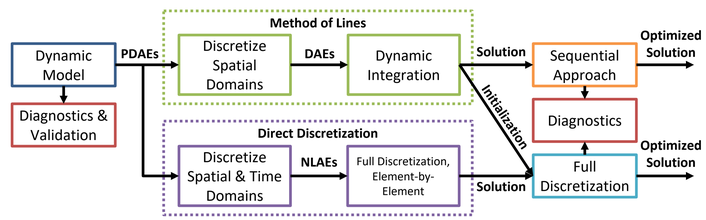


Glass-box optimization
~ 1-5 "Simulation Time Equivalents"
Leverage exact derivatives, sparse structure
[Adapted from Biegler, 2017]

Dynamic modeling and optimization is crucial for:
- Understanding complex trade-offs
- Capturing coupling across time and spatial scales/domains
- Operating dynamic energy systems designed for flexibility

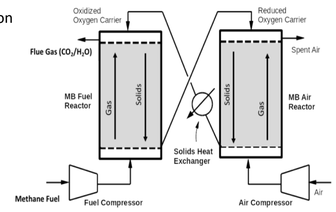
BUT… dynamic optimization problems can be difficult to formulate and solve

IDAES Dynamic Workflow



Chemical Looping Combustion (CLC)

- Gas-solid hydrocarbon reactors conducive to carbon capture
  - **Differential equations:** mass & energy balances, Ergun equation
  - **Algebraic equations:** thermodynamic properties, transport correlations, and many more equations
  - 272 differential eqs, 2437 algebraic eqs, 2 control inputs
- Lessons learned:
  - IDAES streamlines the construction of complex DAE models
  - Modeling mistakes can still happen and can be difficult to manually diagnose
  - A nonsingular DAE model does not guarantee optimization convergence



Modeling & Diagnostics

Requirement for a well-posed (index-1) differential algebraic equation (DAE) model

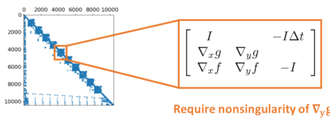
Discretized time-indexed system

$$\left.\begin{aligned} \frac{dx}{dt}\Big|_t &= f(x_t, y_t) \\ \Delta t \left.\frac{dx}{dt}\right|_t &= (x_t - x_{t-1}) \end{aligned}\right\} \quad \forall\, t \neq 0$$

$$0 = g(x_t, y_t) \qquad \forall\, t \neq 0$$

$$x_0 = \bar{x}_0$$

Jacobian



Subsystem

$$\begin{bmatrix} I & -I\Delta t \\ \nabla_x g & \nabla_y g \\ \nabla_x f & \nabla_y f & -I \end{bmatrix}$$

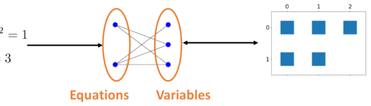
Require nonsingularity of $\nabla_y g$

Every algebraic model has an associated bipartite graph of variables and equations

This graph can tell us a lot about our process model:
- Whether any equations are redundant
- Which variables are *not* valid degrees of freedom
- Whether the model is structurally singular
- Where in the model a singularity is coming from

And can lead us towards decompositions that help us initialize, simulate, or optimize our process model

$$x^2 + y^2 + z^2 = 1$$
$$x + 2y = 3$$

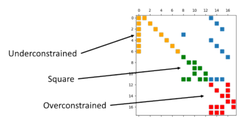


Maximum Matching

- Identifies when submatrix (algebraic Jacobian) is structurally singular
- **Matching**: A set of disjoint pairs of adjacent nodes
- **Maximum Matching**: The largest possible matching (most pairs of nodes) in a given graph
- **Perfect Matching**: A matching that contains every node (variable and equation)

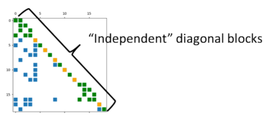
Dulmage-Mendelsohn

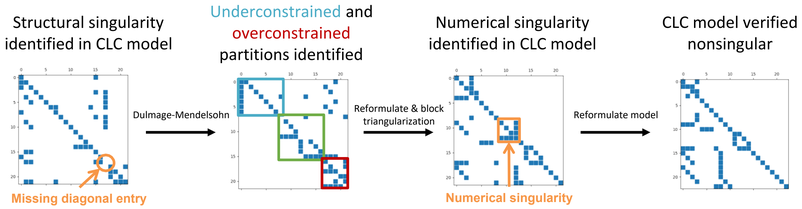
- Identifies source of structural singularity
- Partitions rows and columns into underconstrained, square, and overconstrained subsets
- Determines which variables and equations can possibly be unmatched



Block Triangularization:

- Identifies source of numerically singularity (or poor conditioning)
- Solving diagonal subblocks in order is equivalent to solving entire system
- Nonsingularity is equivalent to nonsingularity of all diagonal blocks





These diagnostic tools now available in pyomo.contrib.incidence_analysis module

Simulation

Dynamic model simulation is crucial for:
- Validating & testing dynamic model during development
- Running simulation cases without optimization
- Initializing fully discretized model
- Verifying solutions obtained using full discretization approach
- Solving dynamic optimization problems using sequential approach

IDAES offers two approaches for simulating dynamic models

PETSc Interface
- Interfaces directly with PETSc solver suite
- Included integrators: implicit Euler, Crank-Nicolson, generalized alpha method
- IDAES distributes a pre-compiled PETSc binary for several platforms
- Includes utilities for saving and loading trajectory data

```
from idaes.core.solvers import petsc

# Build Dynamic IDAES model...

# Simulate with PETSc
petsc.petsc_dae_by_time_element(m, m.time, between=[m.time.first(), m.time.last()])
```

Element-by-element
- Works directly on fully discretized model
- Works with any nonlinear optimization solver
- Solves sequence of square problems over subset of time horizon

```
from idaes.core.util.initialization import initialize_by_time_element
from pyomo.environ import SolverFactory

# Build Dynamic IDAES model...

# Simulate with "element-by-element"
solver = SolverFactory('ipopt')
initialize_by_time_element(m, m.time, solver=solver)
```

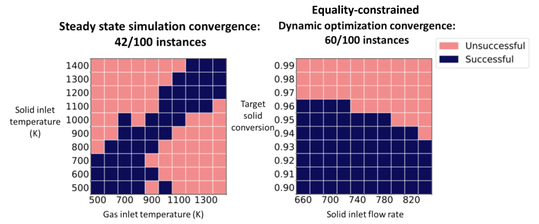
Dynamic simulation tools now available in idaes.core.solvers.petsc and idaes.core.util.initialization

Contact: Bethany Nicholson, blnicho@sandia.gov

Optimization & Convergence Reliability

Nonsingular CLC model did not converge reliably

Sensitive to initialization, initial conditions, and model parameters



Hypothesis: Model is hard to converge because of the algebraic equations

Solution: Remove algebraic equations from NLP and solve them separately exploiting the index-1 property of a well-posed DAE model

Implicit Function Formulation

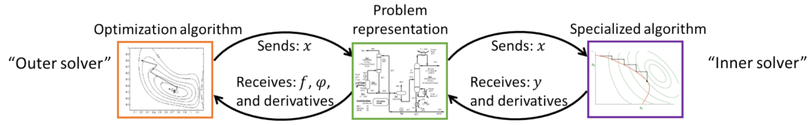
$$\begin{aligned} \min_{x,y} \quad & \varphi(x,y) \\ \text{subject to} \quad & f(x,y) = 0 \\ & g(x,y) = 0 \\ & x \ge 0 \end{aligned}$$

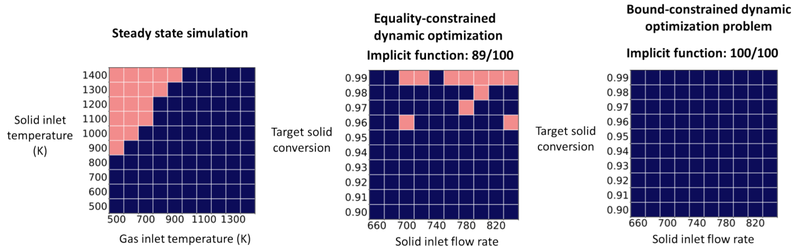
Algebraic equations

$$\begin{aligned} \min_{x} \quad & \bar{\varphi}(x) = \varphi(x, y(x)) \\ \text{subject to} \quad & \bar{f}(x) = f(x, y(x)) = 0 \\ & x \ge 0 \end{aligned}$$

Given x, use $g(x,y) = 0$ to solve for y

"External function"





Significant improvement in convergence reliability

Implicit function interface now available in pyomo.contrib.pynumero module