Mario R. Eden, Marianthi Ierapetritou and Gavin P. Towler (Editors) Proceedings of the 13th
International Symposium on Process Systems Engineering – PSE 2018
July 1-5, 2018, San Diego, California, USA © 2018 Elsevier B.V. All rights reserved.

A Flexible Framework and Model Library for Process Simulation, Optimization and Control

Andrew Lee[a*], Jaffer H. Ghouse[b], Qi Chen[c], John C. Eslick[b], John D. Siirola[d], Ignacio E. Grossman[c], David C. Miller[b]

[a] *KeyLogic Systems, 626 Cochrans Mill Road, Pittsburgh, PA 15236, USA*
[b] *National Energy Technology Laboratory, 626 Cochrans Mill Road, Pittsburgh,, PA 15236, USA*
[c] *Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA*
[d] *Sandia National Laboratories, Albuquerque, NM 87185, USA*
andrew.lee@netl.doe.gov

Abstract

A new framework for optimizing process flowsheets has been developed, which enables a greater degree of flexibility and automation to facilitate the development of hierarchical models suitable for rigorous optimization and control problems under both steady-state and dynamic operating conditions. A library of models for common process unit operations has been developed along with the framework to allow rapid development of flowsheets, with a focus on adaptability and extensibility to allow users to extend and apply these models to novel processes and configurations. This paper presents the development of the framework and model library which support conceptual design, process design/integration, and dynamic optimization/control.

Keywords: Process modeling, simulation, optimization, control, Pyomo.

1. Introduction

Advances in computational power and numerical optimization routines have enabled the possibility of applying rigorous simulation and optimization techniques to large scale problems such as those associated with the design, optimization and control of integrated chemical processes and energy systems. To facilitate the development of process models, a number of chemical process simulation packages, such as Aspen Plus®, gPROMS, ProSim, PRO/II®, have been developed which provide robust and easy to use tools, including libraries of common process unit operations, thermo-physical property models, and efficient algorithms to solve large, sparse systems of nonlinear differential algebraic equations. These tools allow engineers to solve very large process flowsheets under both steady-state and dynamic conditions. However, most process simulation packages focus primarily on solving well defined simulation problems, and often have only limited capabilities for advanced optimization, such as conceptual design or optimization under uncertainty. In contrast, there exist several platforms and modeling languages designed specifically for solving large scale optimization problems; however, these platforms lack the specialized infrastructure and model libraries necessary for easily simulating chemical processes and energy systems. In these cases, models must be laboriously assembled for specific systems, often requiring specialized initialization procedures. Developing and applying these models to chemical processes also requires expert knowledge in modeling and optimization not available to the typical process engineer. Thus, there exists a gap in the capabilities of existing software for optimizing chemical process flowsheets.

The US Department of Energy's Institute for the Design of Advanced Energy Systems (IDAES) is addressing this need by developing a next-generation process systems engineering framework (Miller at al., 2018) that is built for optimization from the ground up, enabling the use of modern optimization solvers with a framework for advanced process modeling. The IDAES framework utilizes the Pyomo (Hart et al., 2017) algebraic modeling language (AML), an open source framework for formulating large scale optimization problems based on the Python programming language, which can interface with a wide range of optimization solvers. As part of the IDAES framework, a library of models has been developed, along with a standard modular framework for linking the models in a flowsheet and solving the resulting problem. The goal of these tools and libraries is to reduce the amount of effort required to develop a process model and facilitate rapid development and screening of process configurations.

2. Framework Development

Despite the wide variety of chemical engineering process unit models, they share a number of common features, allowing for a standardized core modeling structure. The process for developing a unit model can be summarized as follows:

1. Specify the operating mode (dynamic or steady state).
2. Define calculations for the thermophysical, transport and reaction kinetic properties of the material(s) of interest.
3. Specify material and energy inlets and outlets, along with the information (state variables) transferred in each (e.g., total flowrate, pressure, temperature, and mole fractions).
4. Define material, energy, and pressure balances.
5. Define additional constraints describing the behavior of the unit (e.g., pressure-flow relations, or heat transfer correlations).

Of these elements, the material, energy and pressure balances are central to the formulation of the model, with the other elements serving mainly to inform aspects of the balance equations. This would suggest that it is possible to write the balance equations in a generic form, and to substitute expressions or variables for the generic terms as the model needed. Terms which are not needed in a specific model can be excluded by substituting a value of 0 into the balance equations, and the exact form of the expressions can be adapted to the problem at hand to improve the tractability of the problem. Doing this allows for a completely generic formulation of the balance equations that can be reused between unit models, whilst allowing for formulation which can adapt to the differing needs of each unit. Further, the generic form of the balances equations is inherently unitless, which means that by changing the units of measurement used in the individual terms, the units of the entire model can be changed.

2.1. Material, Energy and Pressure Balances

Mass, energy and pressure balances may contain many terms depending on the operation type including: generation of species via chemical reactions, pressure differentials and heat, work and mass transfer. A framework for modeling these systems needs to accommodate all of the possible terms. At a more fundamental level, there are many ways that material and energy balances can be expressed – whether by mass or moles, by element or component, by component flows or total flow and composition, etc. The preferred expression of the balance equations is situation dependent, and often one form is more convenient or provides for a more tractable problem formulation.

Further, the choice of units of measurement needs to be considered, and different applications and industries may prefer different sets of units. Despite the variations that may occur, the general form of the balance equations is constant, consisting of terms that may or may not be present in each specific case. These can be summarized as follows:

1. inlet and outlet terms,
2. accumulation terms, which are needed only in models which consider dynamic behavior (it is generally assumed that pressure changes occur instantaneously, so pressure accumulation terms are generally neglected as well),
3. generation of species due to kinetic and equilibrium based reactions (including phase equilibria), which are primarily functions of the material state, and,
4. transfer terms which are dependent on the behavior of the unit as well as the material state.

For unit models where consideration of spatial variations are not required (0-dimensional models), this results in the following general forms for the material, energy and pressure balance equations (Eqs. (1)–(3));

$$\frac{\partial M_{p,j}}{\partial t} = F_{p,j,in} - F_{p,j,out} + R_{p,j,kin} + R_{p,j,eq} + K_{p,j} \tag{1}$$

$$\frac{\partial E}{\partial t} = \sum_p H_{p,in} - \sum_p H_{p,in} + W + Q \tag{2}$$

$$0 = P_{in} - P_{out} + \Delta P \tag{3}$$

Here $\frac{\partial M_{p,j}}{\partial t}$ is the accumulation of species j in phase p within the unit, $F_{p,j,in}$ and $F_{p,j,out}$ are the flows of species j in phase p in and out, $R_{p,j,kin}$ and $R_{p,j,eq}$ are the generation of species j in phase p by kinetic and equilibrium controlled reactions respectively. Similarly $\frac{\partial E}{\partial t}$ is the accumulation of energy in the unit, $H_{p,in}$ and $H_{p,out}$ are the flows in and out of energy in phase p, and P_{in} and P_{out} are the pressure of the material at the inlet and outlet respectively. $K_{p,j}$, W, Q and ΔP represent the mass, work, heat and pressure transfer terms within the unit respectively.

The traditional approach to expressing these equations within an AML would be to either write custom equations for each model which contain only the necessary terms, or to write the balance equations using generic terms and to then provide additional constraints to describe each term. The first alternative results in a more compact problem with fewer variables and constraints, whilst the second alternative allows for greater flexibility and the reuse of code for the balance equations. By exploiting the object oriented nature of Python, it is possible to write the balance equations dynamically, using procedural code to automatically substitute the terms of the balance equations as they are constructed. This can be used to completely automate the construction of the balance equations, using a small set of instructions to inform the code which terms should be constructed for each unit model.

2.2. Property Calculations

Most available software packages and correlations use intensive forms for the property calculations, as these are dependent only on the state variables of the system, and not the flowrates of material. However, there are situations where using extensive forms yields a more tractable formulation for the overall problem (e.g. adjusting when and where bilinear terms appear in the problem structure). Thus, the framework needs to allow for both extensive and intensive formulations in order to provide the flexibility necessary to achieve the most tractable problem structure. Further, in many cases the developer of a process model is not the developer of thermophysical property models, and may not know the details of the property model formulation. Thus, there needs to be a way for the property calculations to integrate with the unit models without the need for the modeler to be aware of the full details of the property models they are using.

In the IDAES modeling framework, the property module defines the extensive terms in the balance so property models are specific to the application whilst the balance equations are generic. Further, the property calculations become the only part of the model structure where units of measurement are inherently present, which means that the units of measurement for the entire model stem from the property calculations. However, this requires that the units be internally consistent and that property model contain information on the extensive flows of material, which has traditionally been kept separate from the otherwise intensive property calculations.

2.3. Inlets and Outlets

The final consideration when developing a model for a unit operation is what information needs to be passed between units within a flowsheet – i.e. state information for the inlet and outlet streams. The necessary state information and its specific form it may take, is dependent on the specific problem at hand is governed by the system at hand. Considering that any two unit models connected together in a flowsheet are by necessity using the same material, and in most cases the same set of property calculations, it follows that the best form for the definition of the inlet and outlet streams of each unit should be the same. Rather than defining a single standard to which inlets and outlets must conform, the IDAES framework allows each property module to define the best set of state variables to pass between units. In cases where the state variables required by two connected units do not match (e.g. a change in property modules between units), a translator block can be used.

2.4. Models with Spatial Variations

So far this discussion has focused on models where spatial variations can be neglected; however, there are many cases where this is not possible (e.g. plug flow reactors). In these cases it is necessary to include spatial domains and partial differential equations describing these variations within the balance equations. The framework discussed above can easily be expanded to include these derivative terms, and Pyomo includes support for partial differential equations through the Pyomo DAE toolbox (Nicholson et al., 2017), which allows for automatic discretization of domains and numerical approximation of the derivatives for an arbitrary number of domains. All that is necessary to extend the framework is to replace the extensive material, energy and pressure terms in the balance equations with differential forms and to use the inlet (and/or outlet) flows as boundary conditions for the differential equations. An example for a 1-dimensional system is shown below, where x is the spatial domain:

$$\frac{\partial M_j}{\partial t} = -\frac{\partial F_j}{\partial x} + R_{j,kin} + R_{j,eq} + K_j \tag{4}$$

$$\frac{\partial E}{\partial t} = -\frac{\partial H}{\partial x} + W + Q \tag{5}$$

$$0 = -\frac{\partial P}{\partial x} + \Delta P \tag{6}$$

2.5. Model Library

Using the framework developed above, a library of models for the most common process unit operations has been developed as part of the IDAES PSE Framework. Models have been developed for operations such as mixers and splitters, compressors, equilibrium separators, and ideal reactors. These models can be applied to a wide range of problems with varying demands for rigor and tractability, all of these models have been developed with the aim of providing a basic representation for these types of operations that can be easily extended and adapted by users to suit their specific applications. The models in the library have been developed to be the simplest representations of these pieces of equipment possible, with the intent that users will build upon these and add additional correlations as needed to obtain the necessary level of detail and rigor for their particular application.

3. Framework Example

In order to demonstrate the application of the framework discussed above, consider the workflow for generating a steady-state model for an isentropic compressor unit. When creating the model, the user must specify a property package containing the necessary property calculations and a set of instructions on how to build the balance equations. For a compressor, it is clear that the pressure drop and work transfer terms will be required, whilst reactions, heat and mass transfer are generally neglected. Each unit model contains a default set of instructions for the terms most commonly used, however the option is provided for the user to override these if their application demands it.

From there, the automatic framework first retrieves the time domain for the problem from the flowsheet (due to limitation in specifying indexing domains, a time domain is required for steady-state models, however it consists of s single point). Then, based on the instructions provided, the framework automatically generates the mass, energy and pressure balances. Taking the mass balances as an example, the framework first eliminates all unneeded terms by substituting a value of zero (such as the accumulation and reaction terms). The framework then checks with the provided property package to determine the best form to use for the extensive flow terms in the equations. For this example, assuming that the property package uses total flowrate, component mole fractions, temperature and pressure as state variables, the expression $F_p y_{pj}$ would be returned, where F_p is the total flowrate of phase p and $y_{p,j}$ is the mole fraction of component *j* in phase *p*. These are then substituted into the mass balance equation, and an equation automatically written for each component and phase, as shown in Eq. (7).

$$0 = F_{p,in} y_{p,j,in} - F_{p,out} y_{p,j,out} + 0 + 0 + 0 \tag{7}$$

Similarly, the pressure and energy balances are written as shown in Eqs. (8) and (9), where h_p is the specific enthalpy for phase *p*, as calculated by the property package.

$$0 = \sum_p \left(F_{p,in} h_{p,in}\right) - \sum_p \left(F_{p,out} h_{p,out}\right) + W + 0 \quad (8)$$

$$0 = P_{in} - P_{out} + \Delta P \quad (9)$$

The framework then checks the property package again to determine the state variables to include in the inlet and outlet from the unit (in this case F_p, $y_{p,j}$, P and temperature). The user may then provide the constraints necessary to calculate the remaining terms in the balance equation, notably the isentropic constraints relating ΔP and W.

4. Conclusions

A flexible and hierarchical framework for developing and optimization process flowsheet has been developed, which automates much of the development of the mass, energy and pressure balances. The framework has been designed from the ground up to be suitable for advanced optimization and control applications, under both dynamic and steady-state operation. The framework was developed based on the Pyomo AML and leverages the capabilities of modern optimization solvers, using an equation oriented approach with access to complete first and second derivatives. A library of unit models has been developed as part of the IDAES PSE Framework which includes models for many of the common unit operations used in chemical processes. The model library was developed to be adaptable and extensible, in order to allow users to customize the models to suit the needs of their specific problem.

Disclaimer

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

KeyLogic Systems, Inc.'s contributions to this work were funded by the National Energy Technology Laboratory under the Mission Execution and Strategic Analysis contract (DE-FE0025912) for support services. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

References

W.E. Hart, C.D. Laird, J.-P. Watson, D.L. Woodruff, G.A. Hackebeil, B.L. Nicholson, and J.D. Siirola, 2017, Pyomo – Optimization Modeling in Python. Second Edition. Vol. 67. Springer.

W.E. Hart, J.-P. Watson, and D.L. Woodruff, 2011, Pyomo: modeling and solving mathematical programs in Python, Mathematical Programming Computation 3, 3, 219-260.

D.C. Miller, J.D. Sirrola, D. Agarwal, A.P. Burgard, A. Lee, J.C. Eslick, B. Nicholson, C. Laird, L.T. Biegler, D. Bhattacharyya, N.V. Sahinidis, I.E. Grossmann, C.E. Gounaris, and D. Gunter, 2018, Next Generation Multi-Scale Process Systems Engineering Framework, PSE 2018, San Diego.

B.L. Nicholson, J.D. Siirola, J.-P. Watson, V.M. Zavala, L.T. Biegler. 2017, pyomo.dae: A Modeling and Automatic Discretization Framework for Optimization with Differential and Algebraic Equations, Accepted to Math Programming Computation.