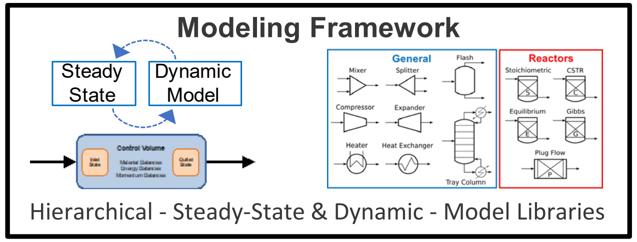
# IDAES for Advanced Users

Alejandro Garciadiego, Doug Allan, Miguel Zamarripa, Andrew Lee,
Dan Gunter

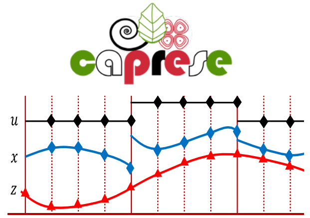
Stakeholder Summit 2023

# IDAES for Advanced Users - Outline

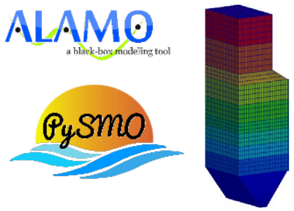
- Model, Flowsheet, Costing Libraries
- Dynamic and Controllable Models
- Surrogate Modeling
- Diagnostics & Visualization



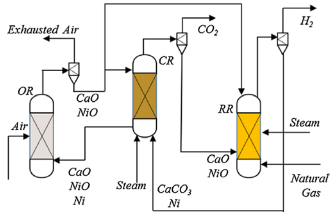

Process Operations Dynamics & Control

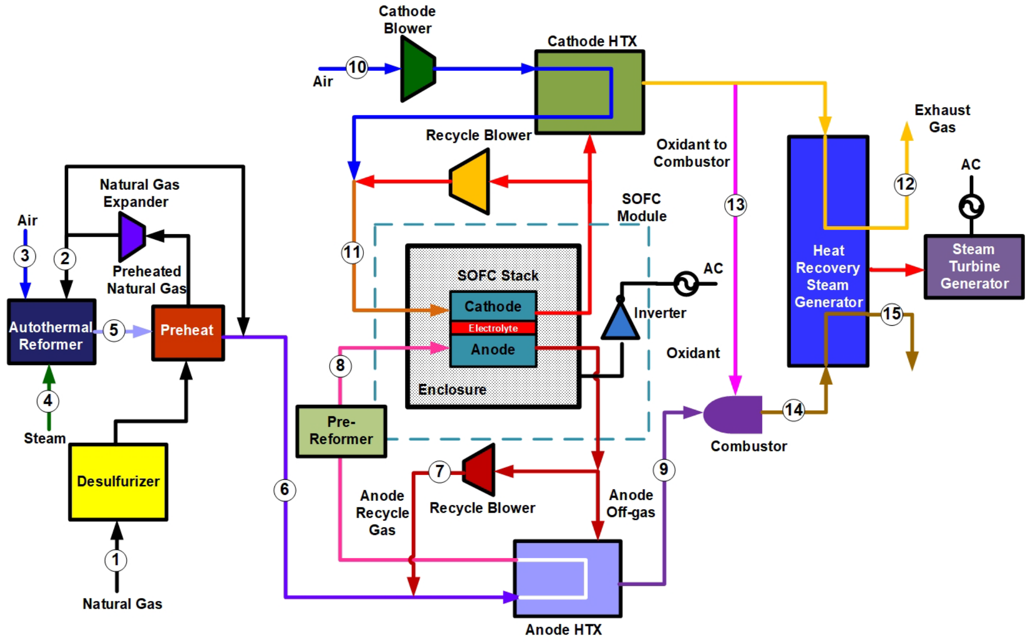


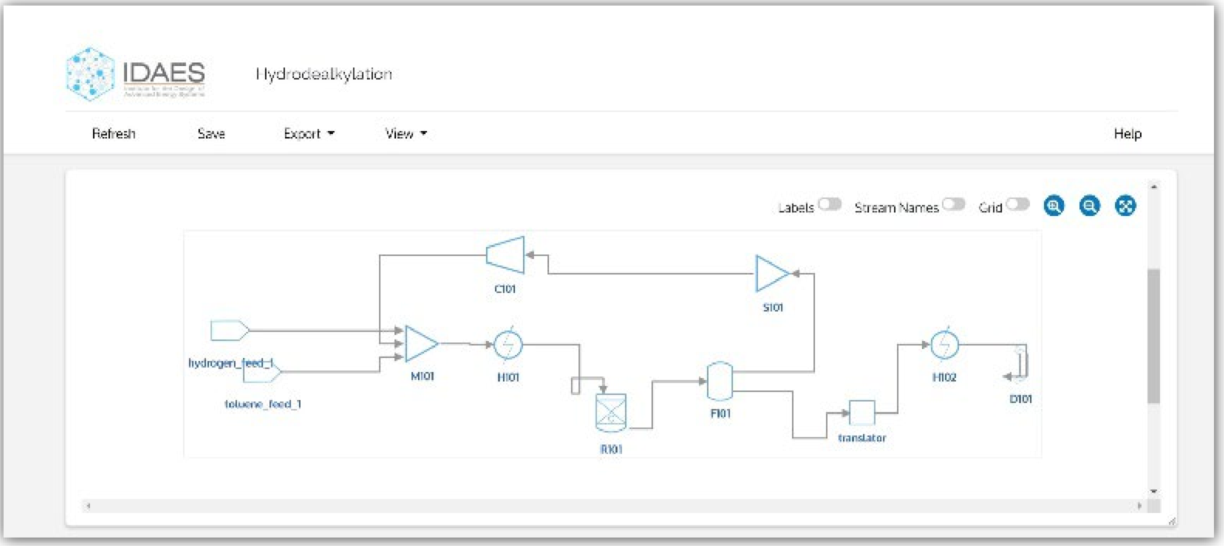

AI/ML Surrogate Modeling




Plant Design Process Optimization



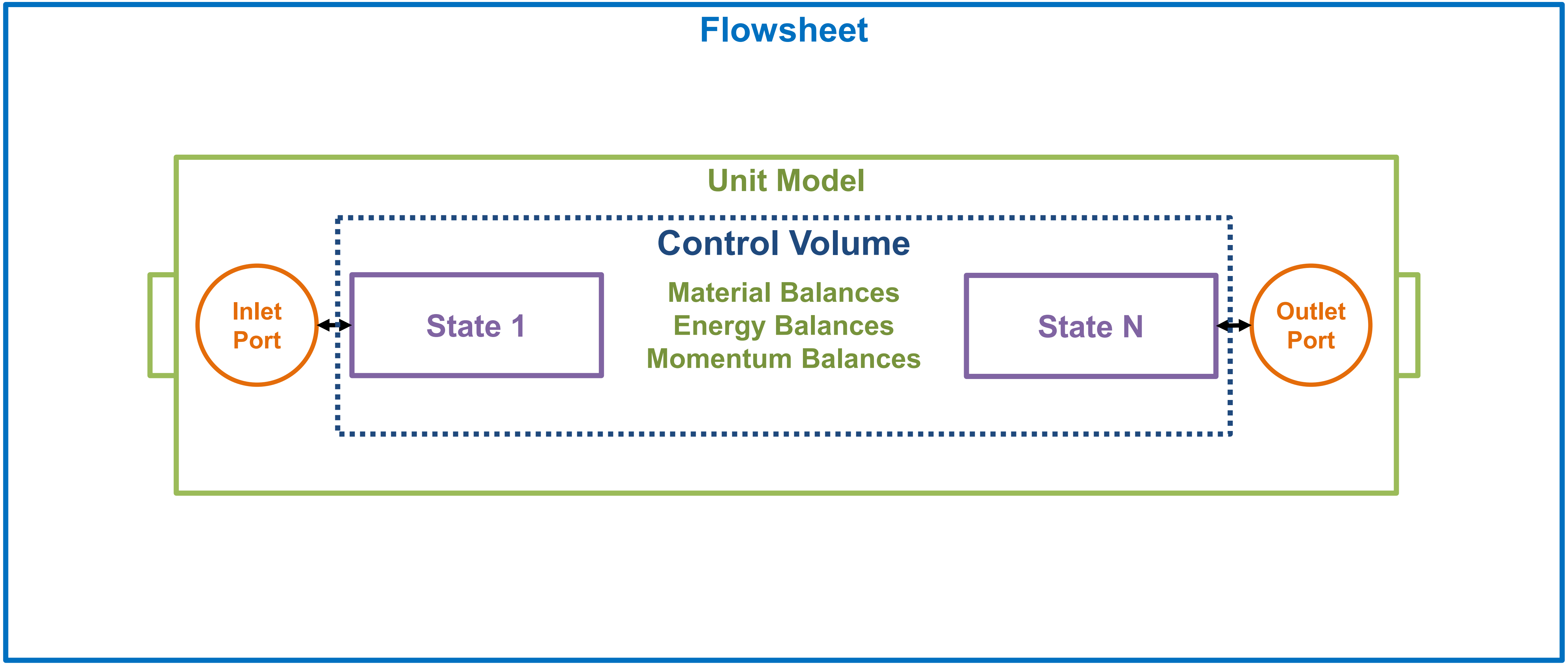



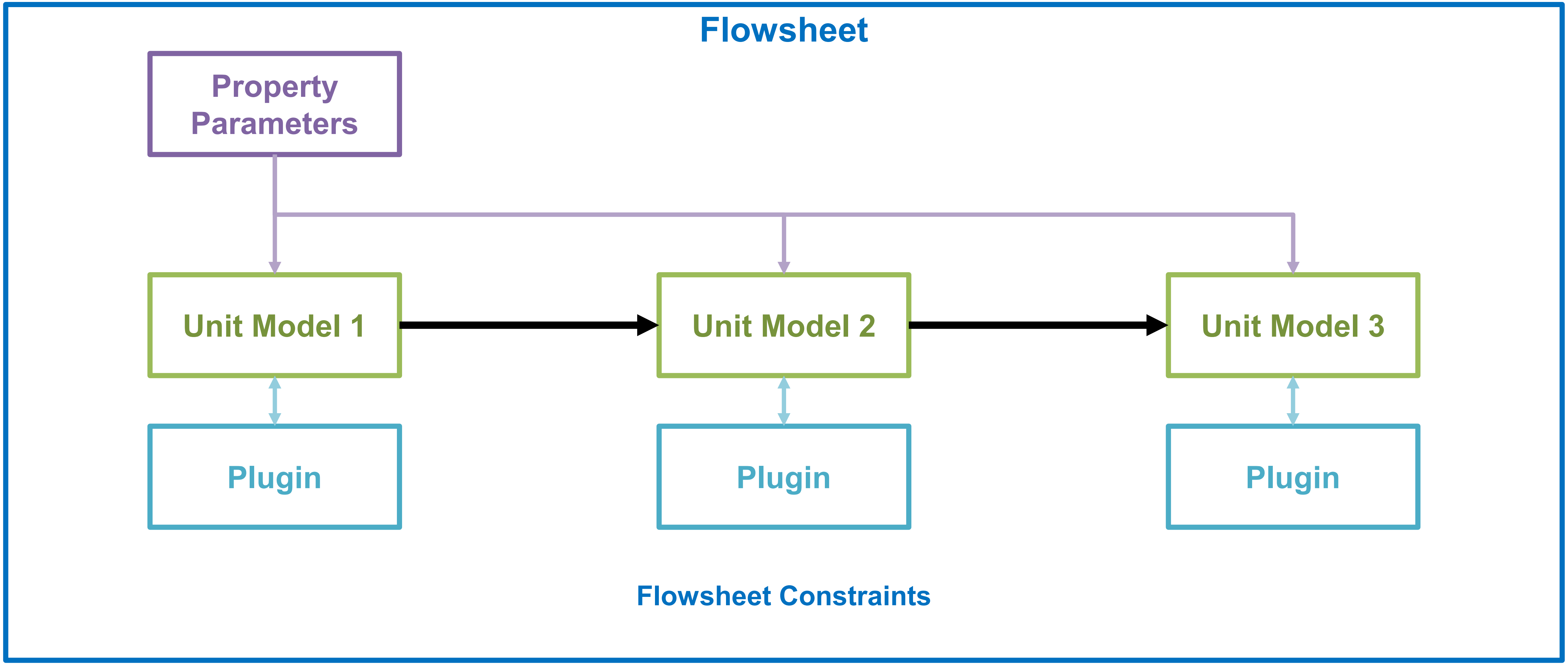

**Case Study – Solid Oxide Fuel Cell Flowsheet in IDAES**

IDAES Institute for the Design of Advanced Energy Systems

# Block Hierarchical Structure



# Block Hierarchical Structure



# IDAES Core Capabilities for Advanced Process Models

- Unit Model Library
- Property Blocks
- Ports
- Control Volumes
- Initialization

IDAES Unit **Model Library** provides basic unit models (i.e., mixers, splitters, flash drums) that can be used as building **blocks** for more **complex models**.

**Properties:**

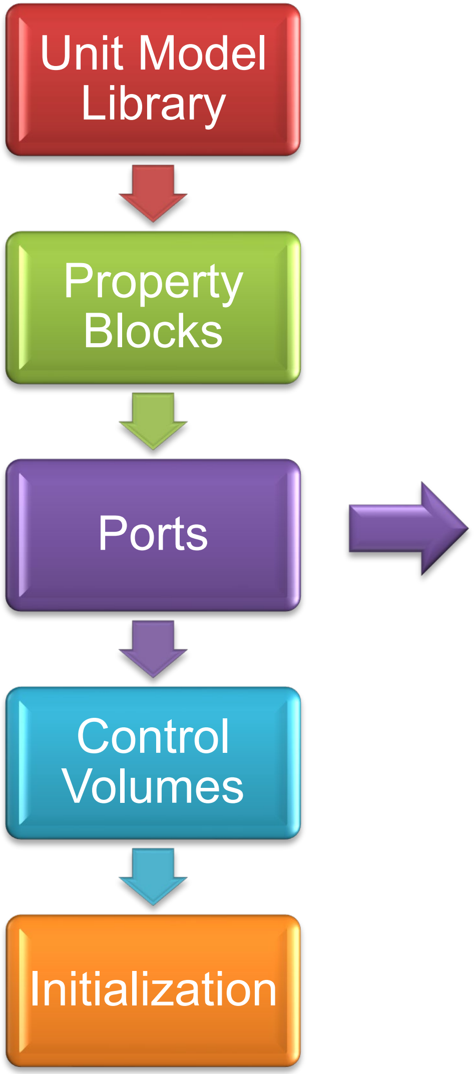
- Implemented as modular blocks
- Flexibility to choose state variables
- Flexibility to use different property packages in same flowsheet
- Flexible formulation of equations

$$V = f(T, P, x) \qquad h = f(T, x) \qquad P_{sat} = f(T)$$

$$Cp = f(T) \qquad y_i = K_i x_i \qquad r = f(T, P, x)$$

IDAES
Institute for the Design of Advanced Energy Systems

# IDAES Core Capabilities for Advanced Process Models

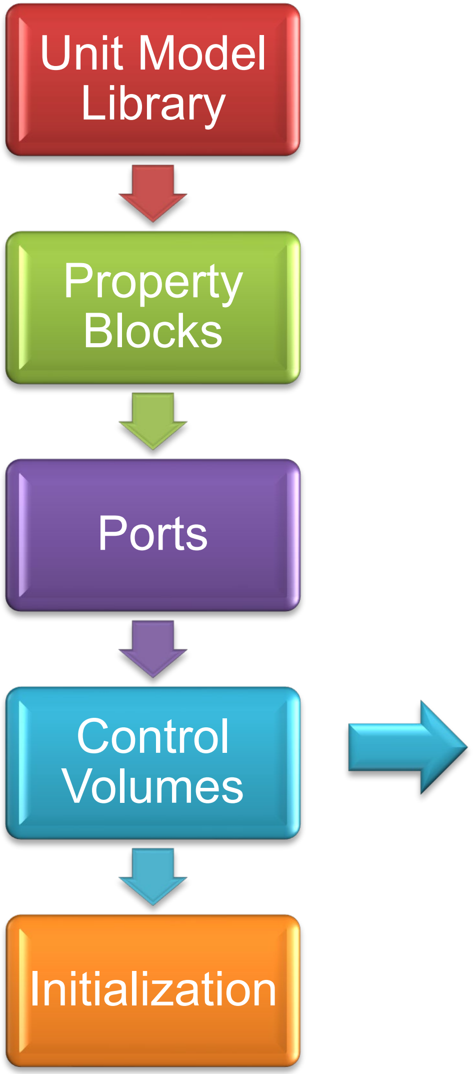



Ports:

- Defined via the property package
- Can connect to any other Port with the same members automatically
- Translator blocks to connect Ports with different state variables

IDAES Institute for the Design of Advanced Energy Systems

# IDAES Core Capabilities for Advanced Process Models




**Control Volumes:**

- Three types of Control Volumes available for different applications
  - 0-D (Inlet-Outlet type)
  - 1-D (Pipes, PFRs)
- Control Volumes can be connected together to develop complex multi-scale models
- Prebuilt **methods** for common forms of **balance equations**:

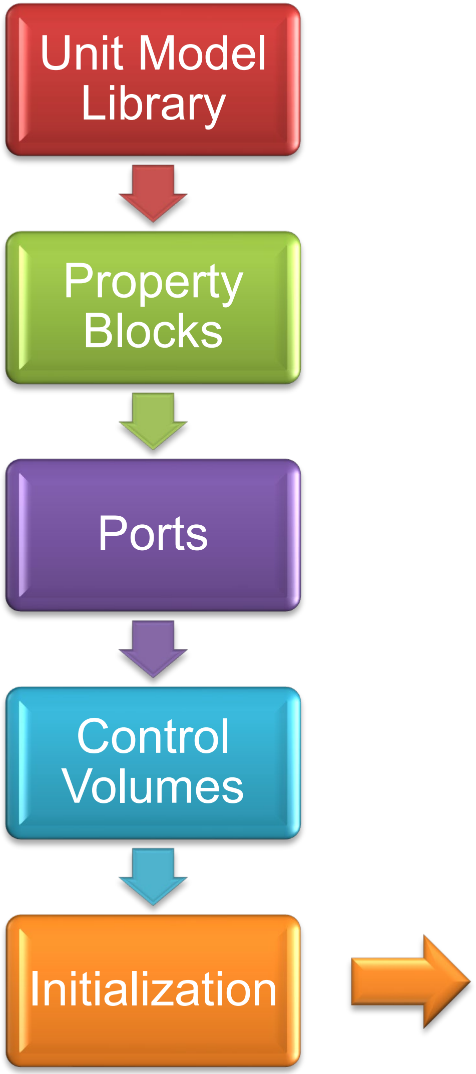
$$\frac{\partial M_j}{\partial t} = F_{in,j} - F_{out,j} + N_{rxn,j} + N_{eq,j} + N_{phase,j}$$

$$\frac{\partial E}{\partial t} = H_{in} - H_{out} + Q + W$$

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

IDAES Institute for the Design of Advanced Energy Systems

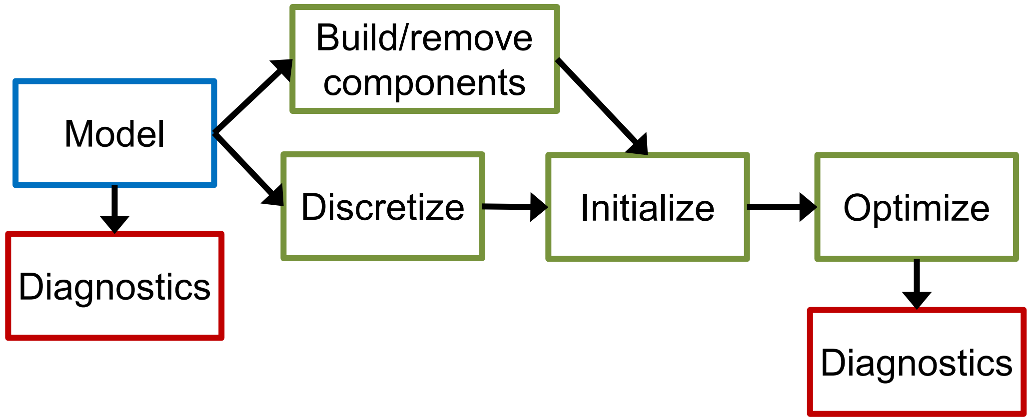
# IDAES Core Capabilities for Advanced Process Models




Initialization:

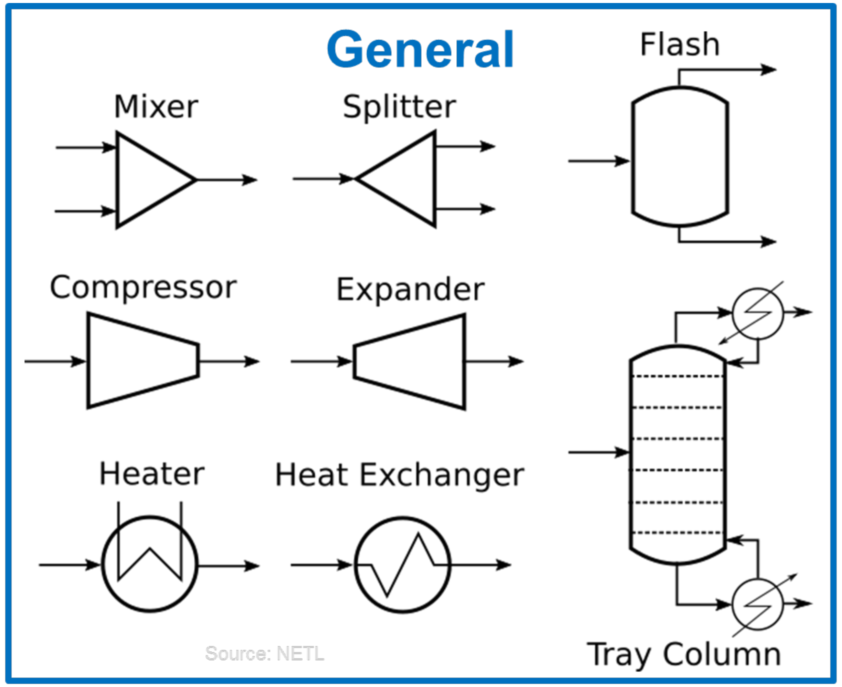
- Custom initialization routines
- IDAES developed tools for solving complex models
    - Decomposition solvers
    - Reliability and convergence tools
- IDAES diagnostics toolset (structural analysis, …)

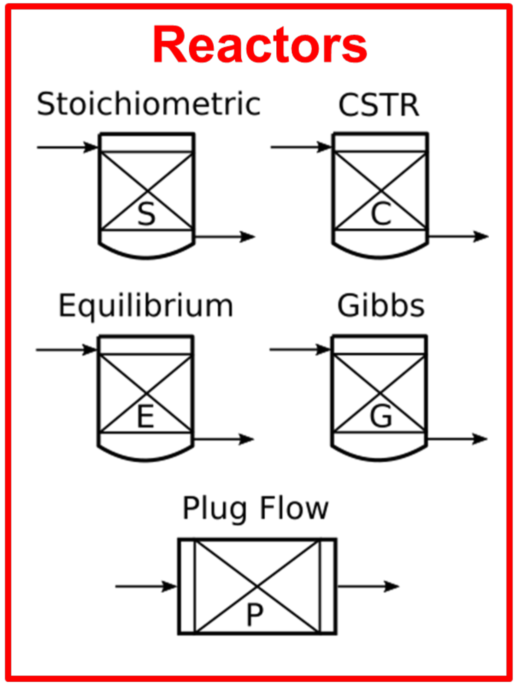
<u>Modular Framework</u>




IDAES Institute for the Design of Advanced Energy Systems

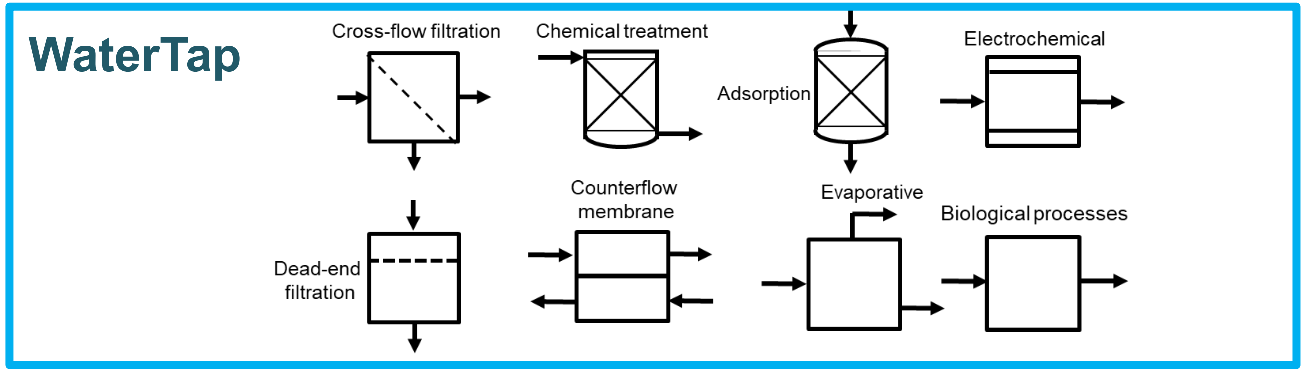
# Unit models developed

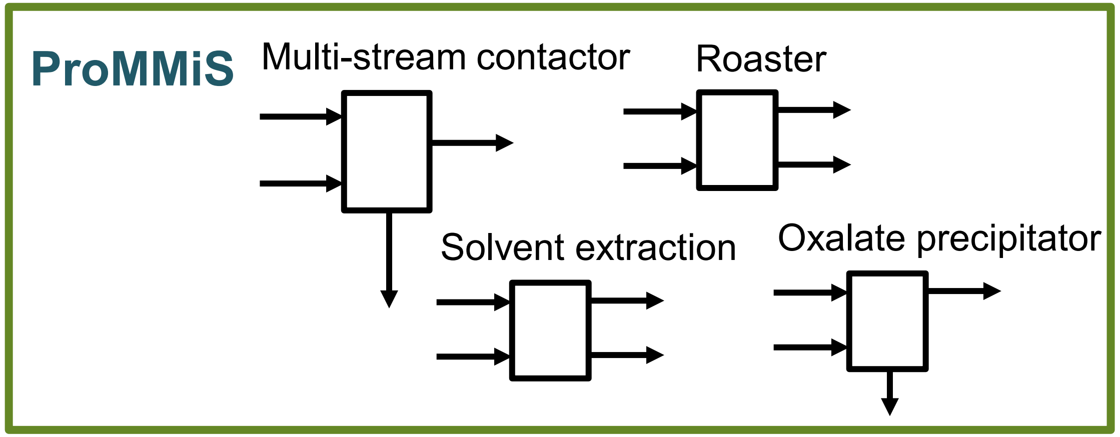


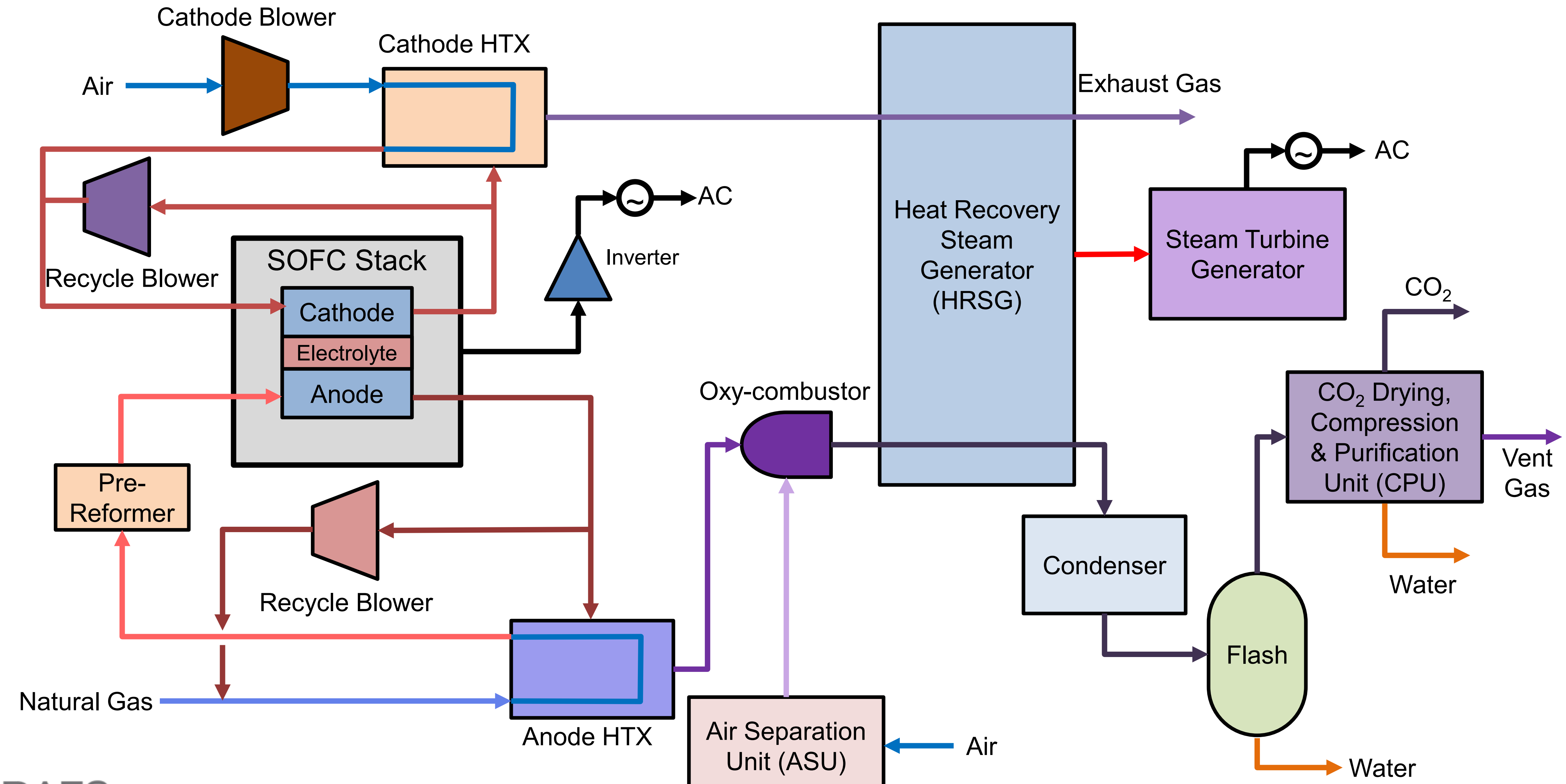

- Library of common thermodynamic models
  - Ideal
  - Cubic Equations of State
  - Helmholtz Equation of State
  - Electrolytes
- Extensive libraries for common unit operations
  - Mixers, Splitters, Heaters, Heat Exchangers, Pressure Changers, Reactors
  - Dynamics and Control
- Specialized Libraries for other equipment
  - Distillation, SOFC/SOEC, Power Generation, Gas-Solids Contactors, Gas-Liquid Absorption

## Project specific models developed leveraging IDAES:



# Solid oxide electrolysis cell (SOFC) Flowsheet building



# Cost Optimization in IDAES using NETL's QGESS[1] Methodology

## Data Sources

NETL QGESS Reports

FEED Studies

Legacy Models

Public References

## General Costing Library

| Unit | Cost Correlation | Basis |
|---|---|---|
| Heat Exchanger | $C_B = f(area)$<br>$CP = (CE/500)F_D F_M C_B$ | Area, Material, and pressure |
| Reactors, Flash, Vessel | $C_V = f(W)$<br>$C_{PL} = f(D, L)$<br>$CP = (CE/500)C_{PL} F_M C_V$ | Material, Dimensions (length, diameter) |
| Compressor | $C_B = f(Pc)$<br>$CP = (CE/500)F_D F_M C_B$ | $P_C$ - Theoretical reversible adiabatic power (horsepower) |
| Distillation column | $C_V = f(W)$<br>$C_{PL} = f(D, L)$<br>$C_{BT} = f(D)$<br>$CP = (CE/500)C_{PL} F_M C_V$ | Material, Dimensions (length, diameter), number and type of trays. |

…

Scale costs as appropriate:

$$SC = RC\left(\frac{SP}{RP}\right)^{EXP}$$

SC: Scaled Cost
RC: Reference Cost
SP: Scaled Parameter
RP: Reference Parameter

## Implementation

Excel/CSV

Python Dictionary

IDAES Library

**IDAES:** General unit model library, power plant library (turbines, heaters, fired boilers, electric boilers, condensers, pumps, compressors, etc.)

**WaterTap:** desalination and water treatment

**PARETO:** Produced water desalination and evaporation

[1] QGESS: Capital Cost Scaling Methodology Revision 4
https://github.com/IDAES/idaes-pse/tree/main/idaes/models_extra/power_generation/costing

IDAES Institute for the Design of Advanced Energy Systems

# Summary

1. Developed unit models and property models in the IDAES framework

2. Contributed unit model and flowsheet examples to the IDAES open-source GitHub repository

3. Developed costing in IDAES based on NETL's QGESS[1] Methodology

4. Modeled and simulated an initial NGFC flowsheet without CCS in IDAES Core Modeling Framework (CMF)

5. Developed a natural gas property package in IDAES CMF

6. Compared initial NGFC flowsheet with state-of-the-art commercial simulators

7. Contributed NGFC example to the IDAES open-source GitHub repository

# Motivation: Dynamic Modeling, Simulation, and Optimization

- No plant is truly at steady state
- When optimization is used for design, resulting plant may not be **controllable**
  - Setpoint transition
  - Start up/shut down
- **Dynamic operation** of integrated energy systems (IES) can take advantage of dynamic electricity pricing
- **Equipment degradation** occurs over thousands of hours

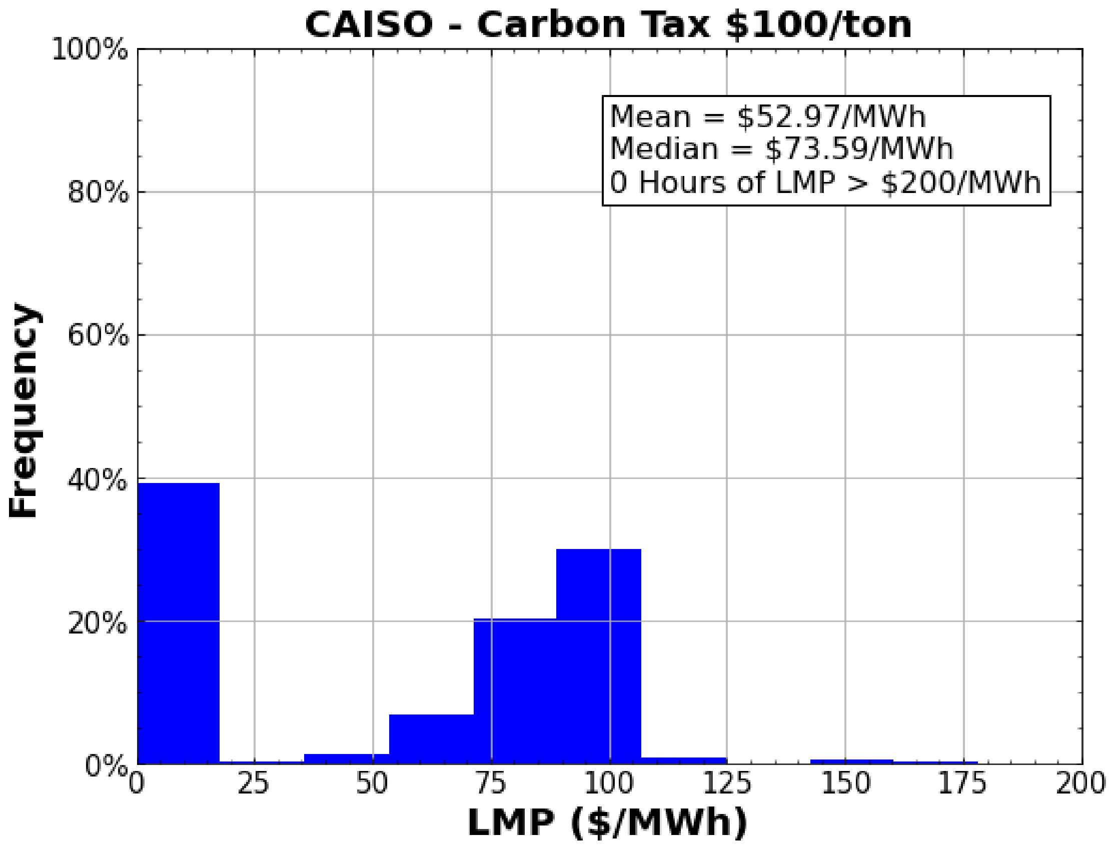
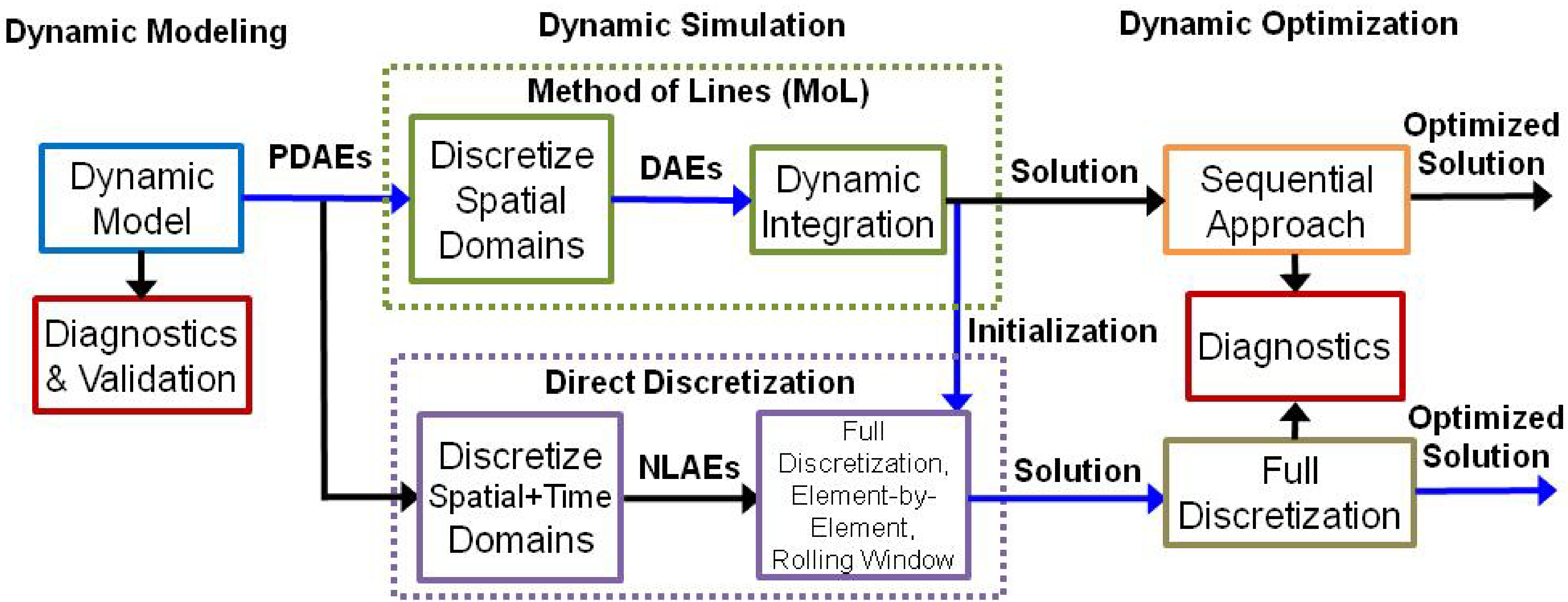



Figure from a presentation by Cortes et al.

(Price Signal Data): Cohen, Stuart; Durvasulu, Venkat (2021): NREL Price Series Developed for the ARPA-E FLECCS Program. National Renewable Energy Laboratory.

# IDAES Dynamic Modeling/Simulation/Optimization Workflow

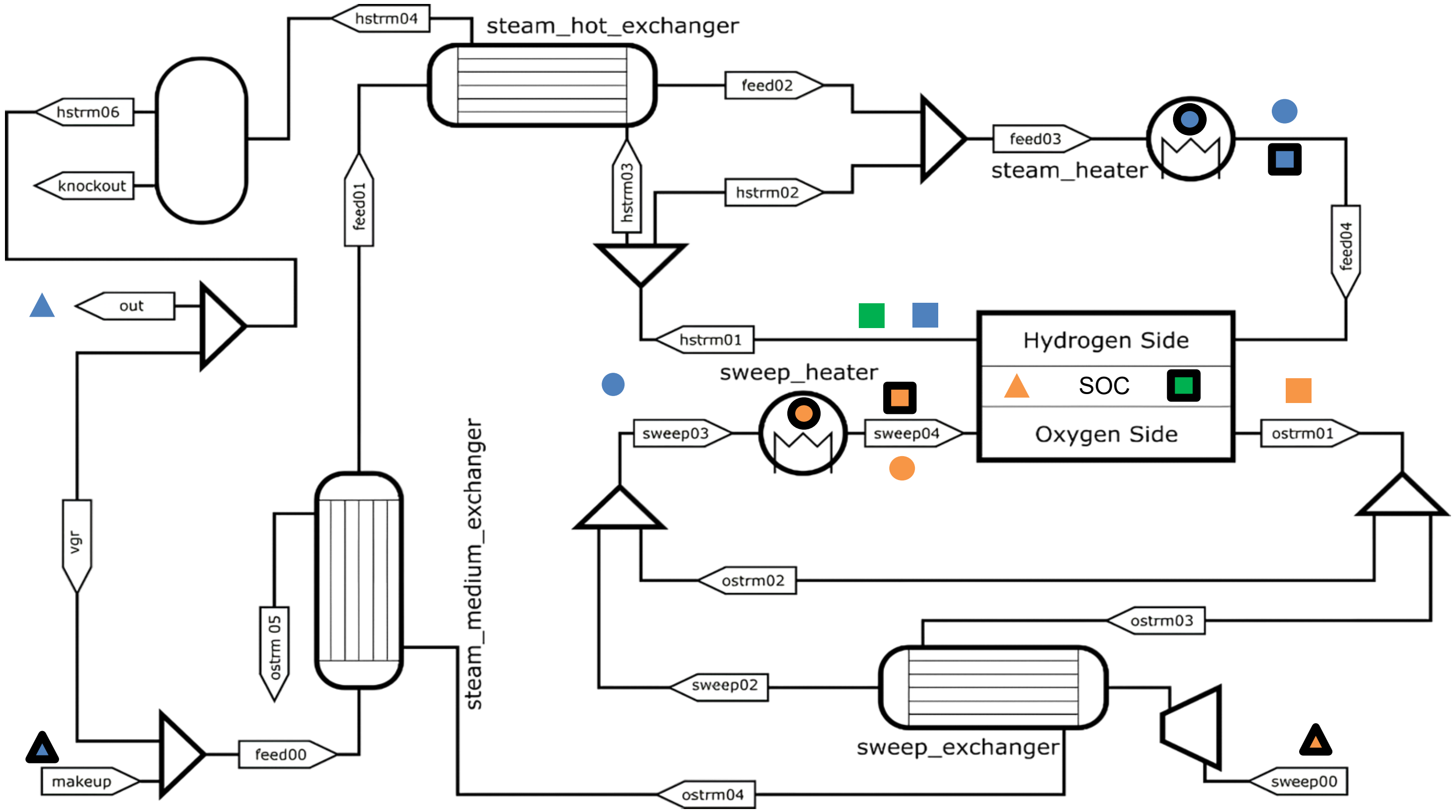


# IDAES Dynamic Modeling/Simulation/Optimization Workflow

- **PYOMO DAE**
  - Develop differential algebraic equation (DAE) model
  - Apply diagnostic tools for structural/numerical analysis
- **IDAES**
  - Set some/all unit models to be dynamic
  - PID Controller implemented with anti-windup
- **Dynamic Simulation**
  - Method-of-Lines (MoL) (AMPL/PETSc/TS)
  - Direct Discretization
- **Dynamic Optimization**
  - Full Discretization
  - Can implement nonlinear model predictive control (NMPC)

# Case Study: Control of Solid Oxide Cell (SOC)-IES

| Controller Type | Manipulated Variable (MV) | Controlled Variable (CV) |
| --- | --- | --- |
| PI | Cell potential | SOC fuel outlet H2 mole fraction |
| P | Makeup feed rate | Hydrogen production rate |
| P | Sweep feed rate | SOC stack core temperature |
| PI (C1I) | Steam heater duty | Steam heater outlet temperature |
| PI (C2I) | Sweep heater duty | Sweep heater outlet temperature |
| P (C1O) | Steam heater outlet temperature setpoint* | SOC feed outlet temperature |
| P (C2O) | Sweep heater outlet temperature setpoint* | SOC sweep outlet temperature |

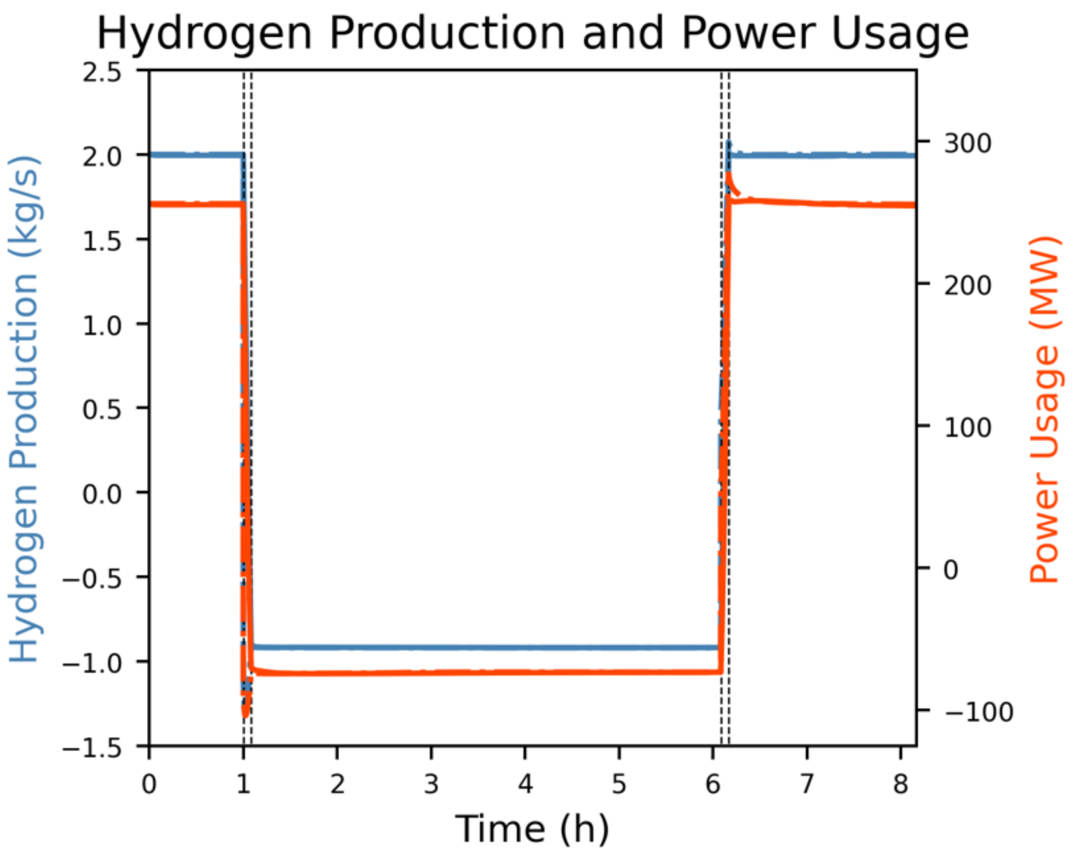



- Plantwide PI control setup with cascade control
- Compare to NMPC, which is better able to handle variable interactions and constraints

IDAES Institute for the Design of Advanced Energy Systems

# SOC-IES Case Study: Comparison of PI control to NMPC

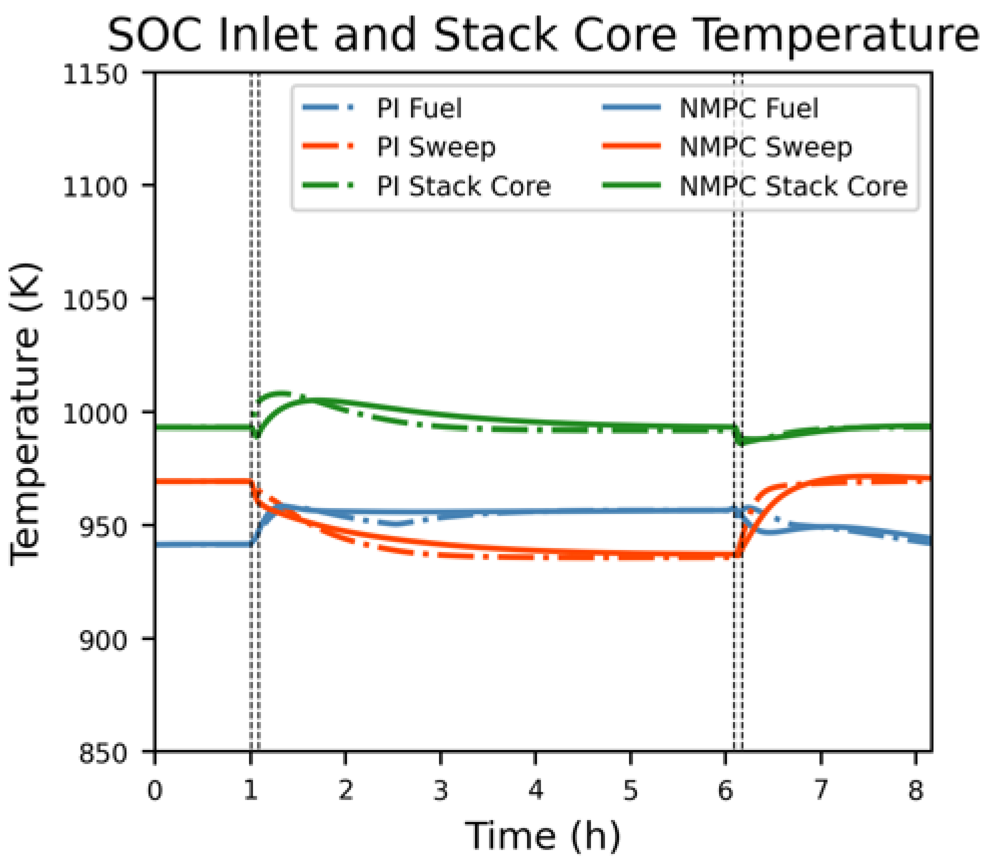
- Both PI and NMPC quickly transition between setpoints for maximum $H_2$ production to power generation (within 5-10 minutes) and back
- PI overshoots on power usage, whereas NMPC does not
  - Possibly able to be smoothed over by a local battery
- Both control methods are able to achieve a rapid transition

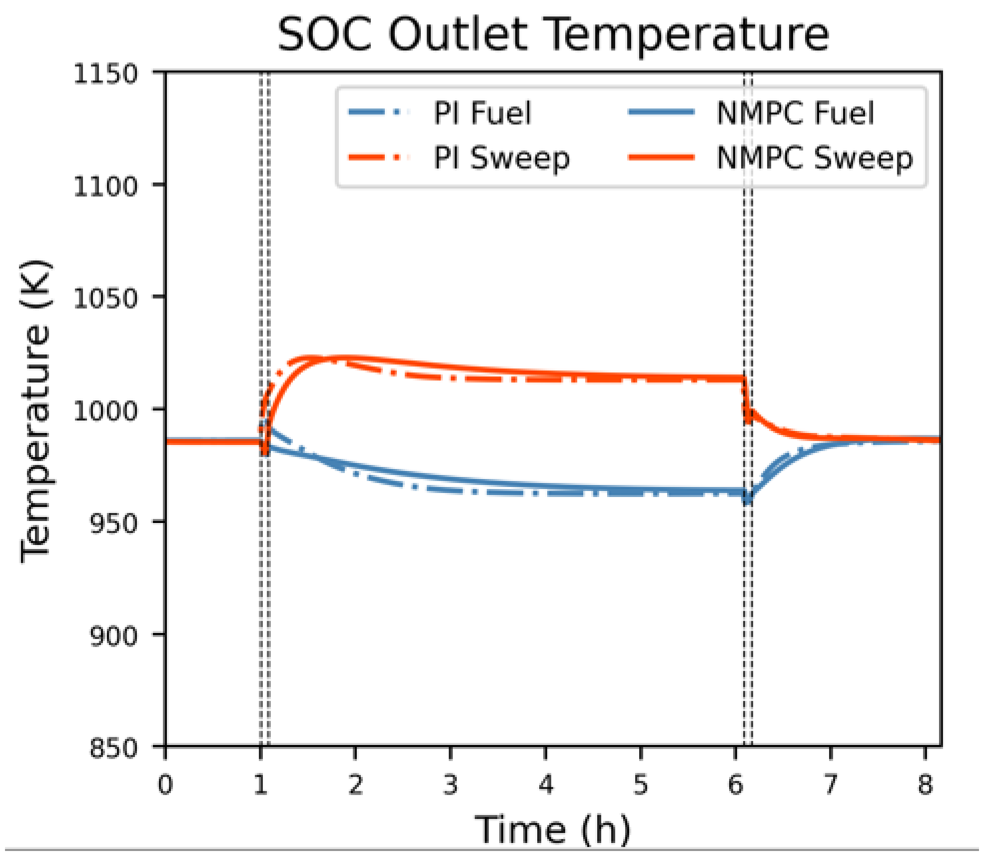



NMPC work by Mingrui (Michael) Li, Vibhav Dabadghao, and Lorenz (Larry) Biegler from CMU, more details on poster

NMPC --- solid line
PI --- dashed line

# SOC-IES Case Study: Slow thermal dynamics

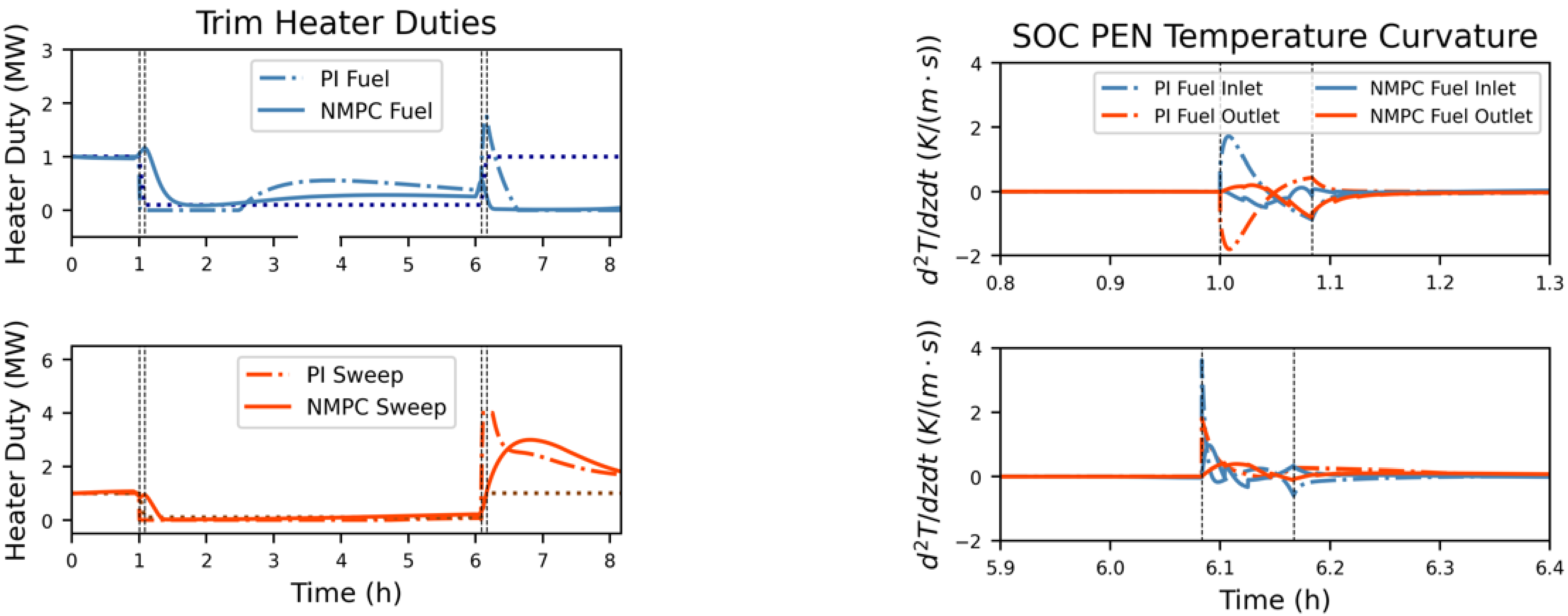



- $H_2$ production responds immediately, but temperature takes hours to settle

IDAES Institute for the Design of Advanced Energy Systems

# SOC-IES Case Study: Trim heaters and mixed partials



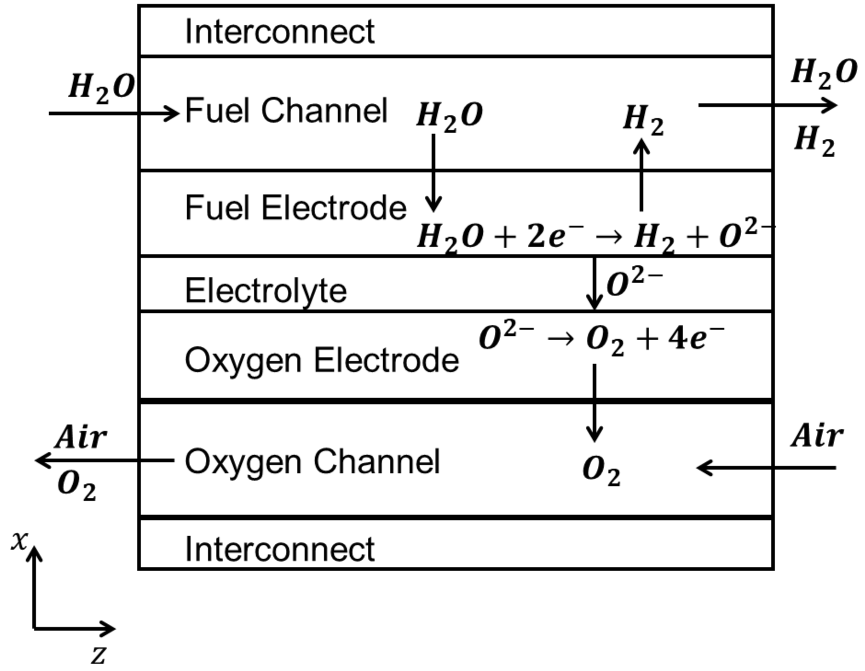

- Trim heaters are still engaged at end of time period, indicating the entire system is not at steady state yet
- NMPC can minimize magnitude of time derivative of cell temperature gradient (mixed partial) in order to reduce thermal failure probability

IDAES Institute for the Design of Advanced Energy Systems

# Case Study: SOEC Degradation Modeling

- Due to degradation, **SOC performance is time-varying**.
- Optimize SOC system performance over **cell's entire lifespan** to maximize net present value (NPV)
- Cell experiences both **physical** and **chemical degradation.**
- Due to mass and heat integration, it is desired that the entire SOC-IES be considered.
- Results presented are for **electrolysis mode only.**
- Long timescales (up to 20,000 hours) mean that **everything in IES besides cell is at steady-state**

SOC degradation work by Nishant Giridhar, Quang Minh Li, and Debangsu Bhattacharyya at WVU, more details on poster




**Planar fuel electrode supported SOC**

**Materials**
- Fuel electrode – Ni-YSZ
- Oxygen electrode – LSM-YSZ
- Electrolyte – YSZ

**Modeling details**
- First principles dynamic 2-D, non-isothermal equation-oriented model
- Capable of SOFC and SOEC operation
- This work - SOEC

IDAES Institute for the Design of Advanced Energy Systems

# SOEC Microstructure Degradation Modeling

- Degradation occurs over thousands of hours

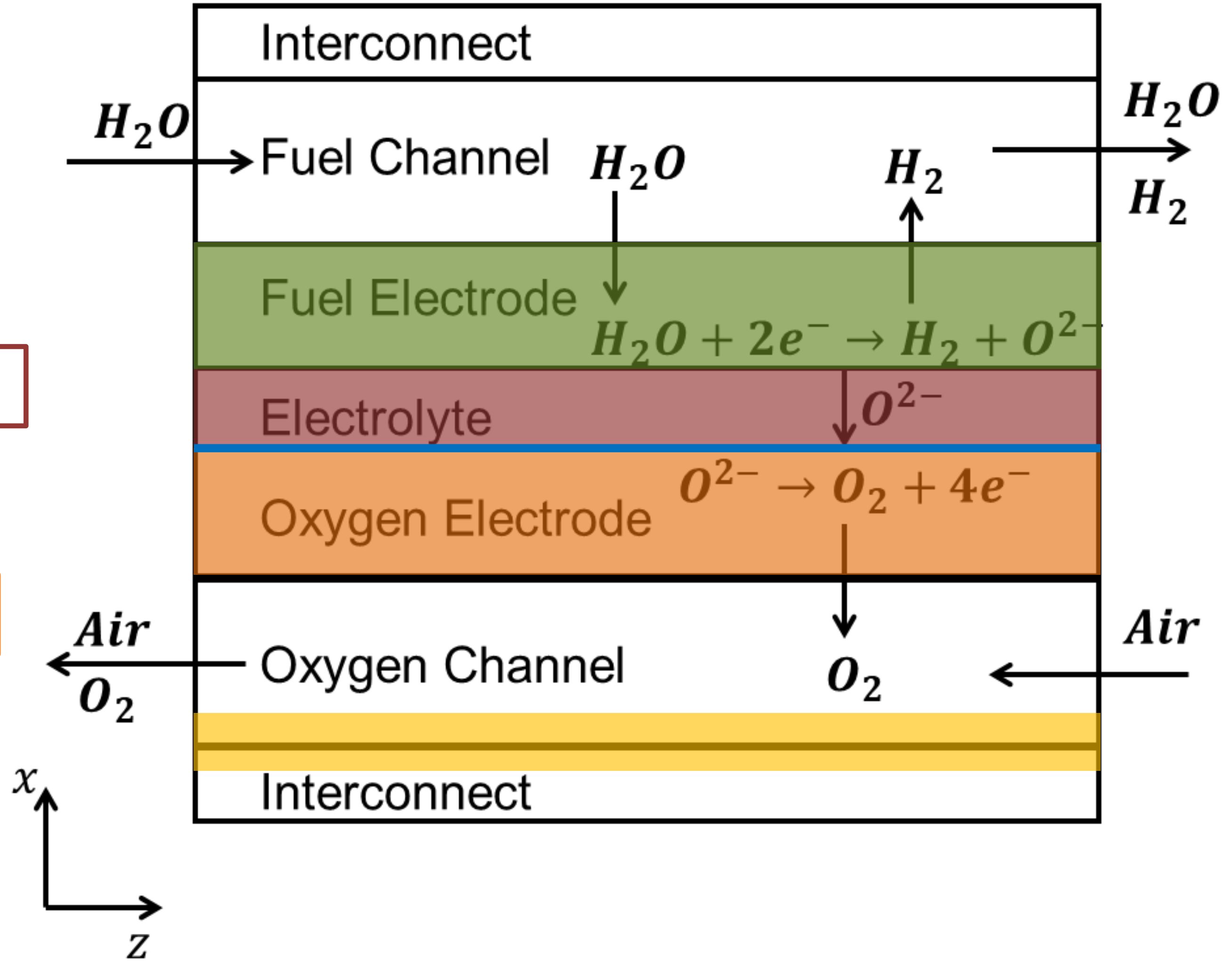
**Fuel electrode Ni agglomeration**

**YSZ electrolyte phase transformation**

**LSM-YSZ phase coarsening**

**Lanthanum zirconate scale growth**

**Chromium oxide scale growth**



# Optimizing Long-Term SOEC System Operation

**Case 0: Constant Temperature**

- Constant $H_2$ production rate at 1.5 kg/s
- Constant temperature at SOC inlets

Decision variables at each time point:

1. Feed heater duties
2. Sweep heater duties
3. Sweep blower flowrate
4. Feed exchanger flowrate
5. Feed recycle ratio
6. Sweep recycle ratio

**Case 1: Maximize Integral Efficiency**

- Constant $H_2$ production rate at 1.5 kg/s
- Objective: Maximize integral efficiency for the flowsheet.

$$\max \frac{1}{t_f} \sum_{t=0}^{t_f} \eta_{th,t} \Delta t_t$$

$$st.$$

$$h(x) = 0$$

$$\frac{dR}{dt} = f_R(x,t)$$

$$j_t = j_0 \qquad \forall\, t$$

$$\eta_{th,t} = \frac{HHV(H_2,t)}{P_{in,total,t}}$$

**Case 2: Minimize Final Degradation**

- Constant $H_2$ production rate at 1.5 kg/s
- Objective: Minimize resistance growth for oxygen electrode

$$\min \frac{R_{tf}}{\Delta t}$$

$$st.$$

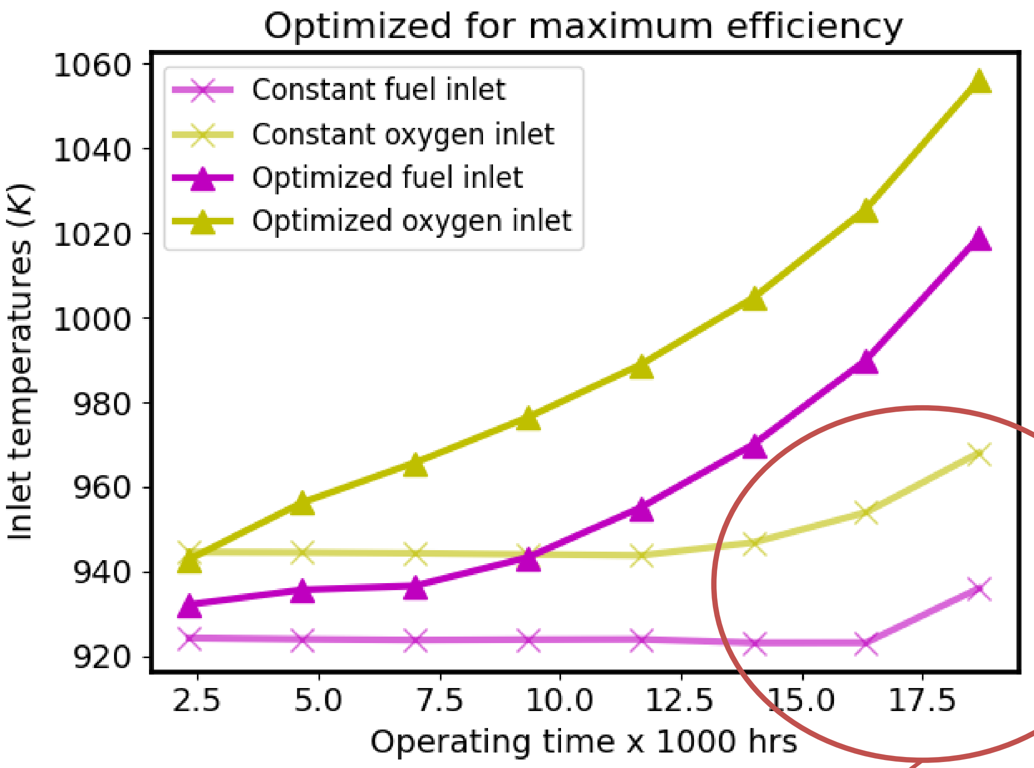
$$h(x) = 0$$

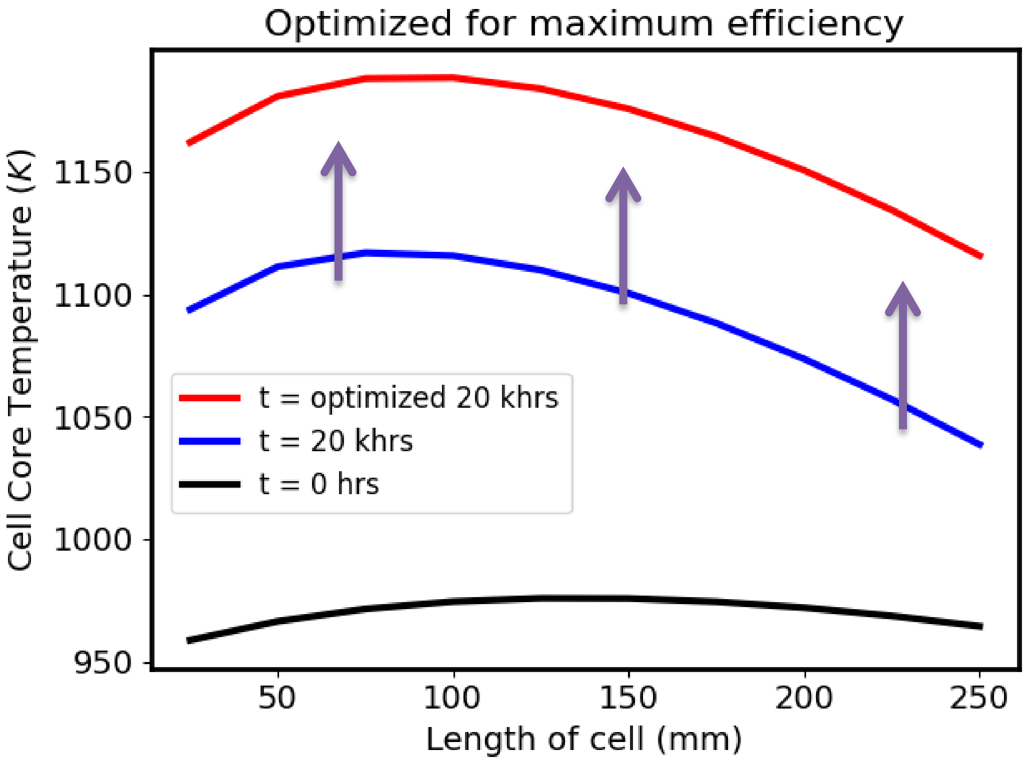
$$\frac{dR}{dt} = f_R(x,t)$$

$$j_t = j_0 \qquad \forall\, t$$

IDAES Institute for the Design of Advanced Energy Systems

# Case 1. Maximize SOEC Efficiency over 20,000 hrs of Operation



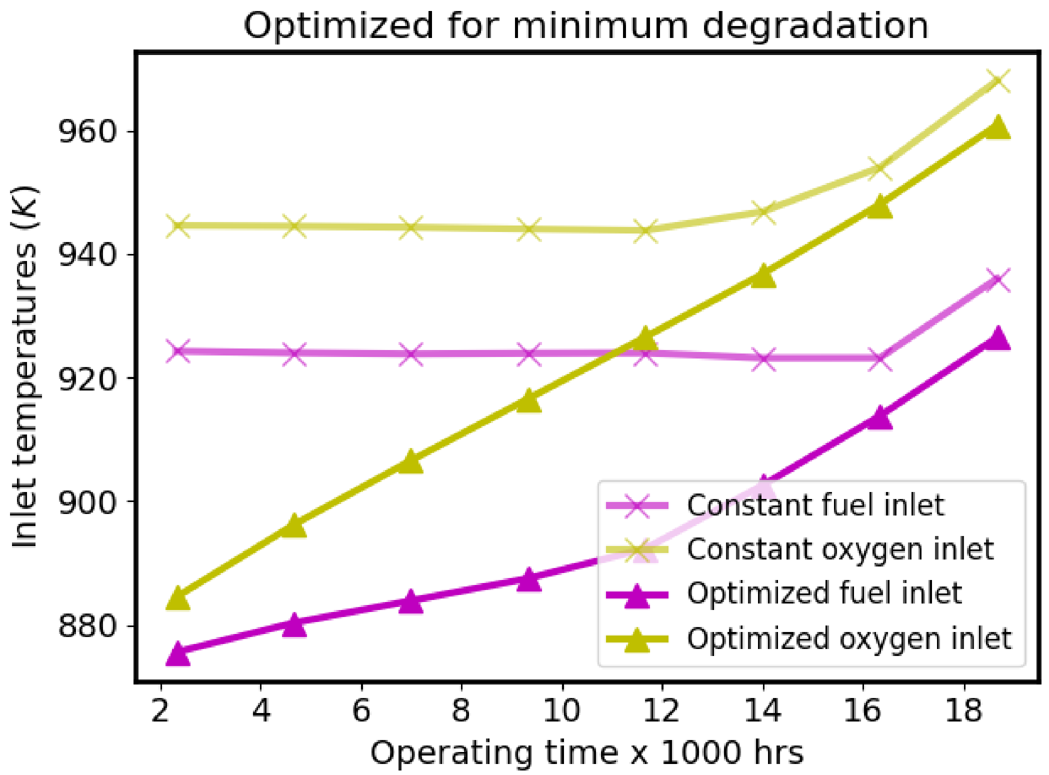

**Inlet temperatures for maximizing efficiency:**

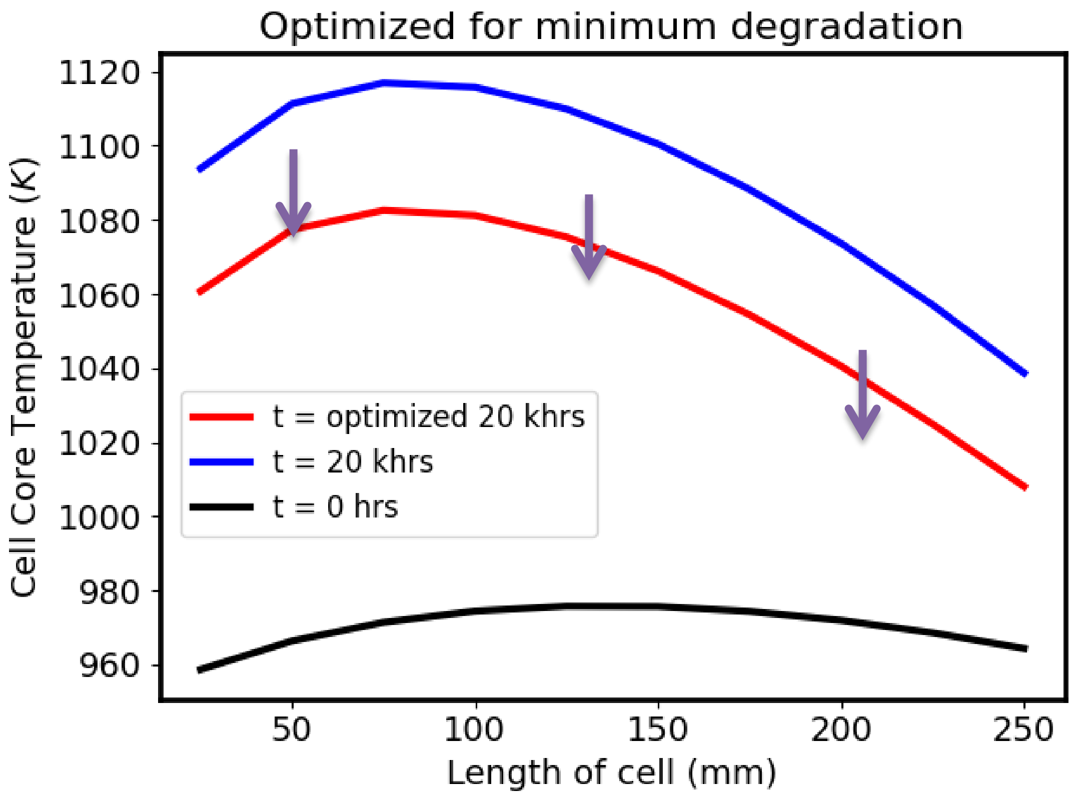
- Inlet temperature increased to **reduce resistive losses**
- Increased temperature comes at a price of **increased chemical degradation**

**Constant inlet temperature:**

Increased Joule heating in cell causes **air blower to hit a bound**, so temperature must increase.

# Case 2. Minimize SOEC Degradation over 20,000 hrs of Operation



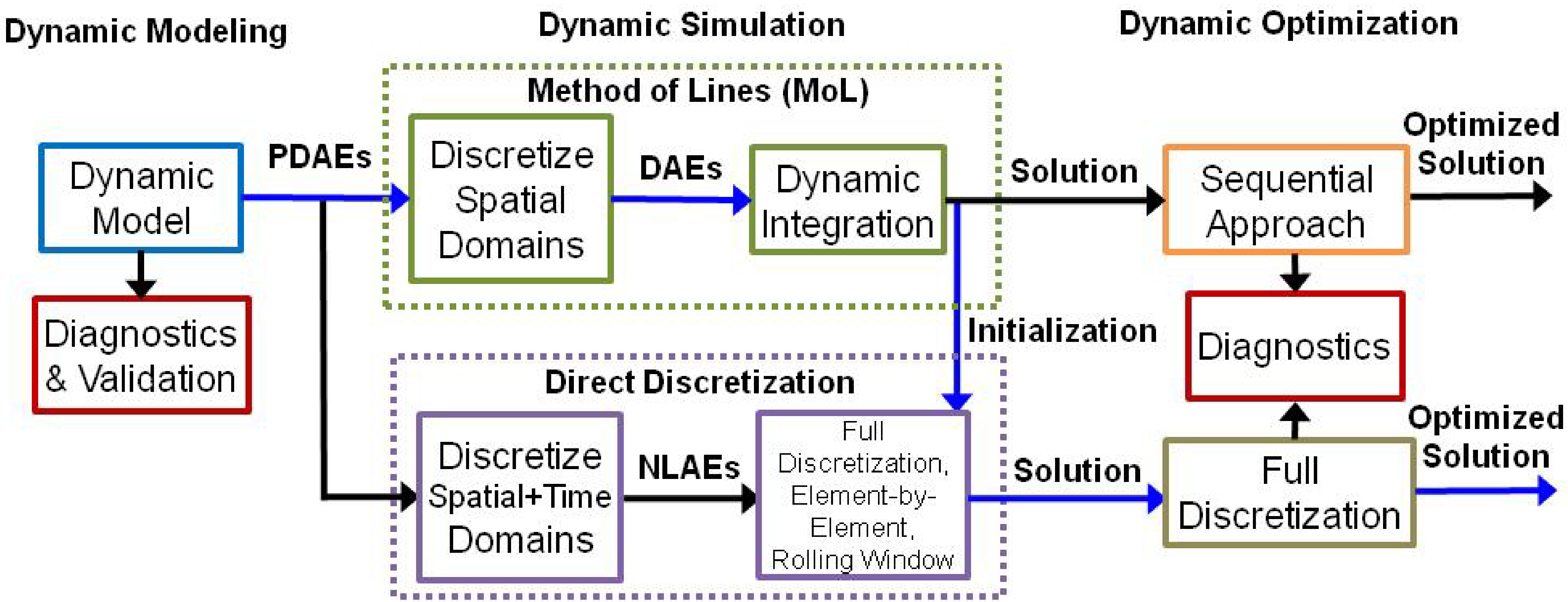




**Inlet temperatures for minimizing degradation:**

- **Inlet temperatures remain low** throughout the operating period.
- **Cell core temperatures** about 60 K **lower** than the constant temperature case.

IDAES Institute for the Design of Advanced Energy Systems

# Dynamics Questions




Questions about Pyomo/IDAES dynamic capabilities?

IDAES Institute for the Design of Advanced Energy Systems

## Surrogate Modeling in IDAES

**What it is new?**
Development: Trust region framework, examples, notebooks,
Application: market integration, SOFC examples, etc.

# Machine Learning and AI for Optimization

Automated Learning of Algebraic Models (ALAMO)

Integrating ML Models in IDAES Flowsheet Optimization

Trust-region Methods for Optimization of Surrogates and Reduced-order Models

Unified API to Support use of Surrogates across IDAES

Reinforcement Learning for Enhanced Control of IDAES Applications

Machine Learning Approaches for Solver Performance Improvements

Fundamental Tools Enabling Workflows Across IDAES Team

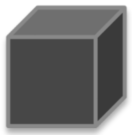
**“Consistent Framework to Validate/Verify ML/AI Tools for Optimization”**

# IDAES Framework: Surrogates for Optimization

**Data Source**

Experimental Data

Black-Box Models (e.g., ASPEN, Prescient, CFD)



IDAES Models or Properties

**Sampling**

Common Sampling Tools

idaes.
surrogate.sampling

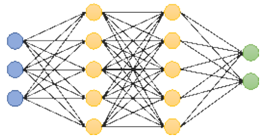
PySMO

ALAMO a black-box modeling tool

**Model Selection and Training**

ALAMO a black-box modeling tool

PySMO



IDAES ML Surrogates

RIPE

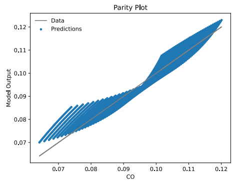
HELMET

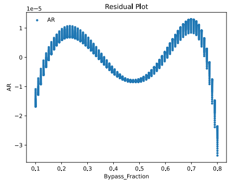
**Visualization and Validation**

Standard Metrics

RMSE = 0.291E-1
R2 = 0.988

Plotting and Visualization Tools




**Integrated IDAES Model**

EO Model: Correlation, Property Package, Unit Model

Direct Evaluation

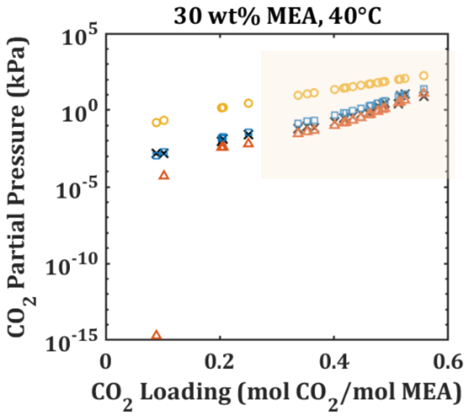
z[0.75, 1.0] =
3.02965425e-01

Validate solution against rigorous model or data

# 1.- Data Source “Best Practices”


Ensure **quantity and quality**

- Experiments: visualize data to guarantee sampling distribution
- Simulation: ensure model robustness




80% of data is between 0.4 and 0.6 $CO_2$ loading

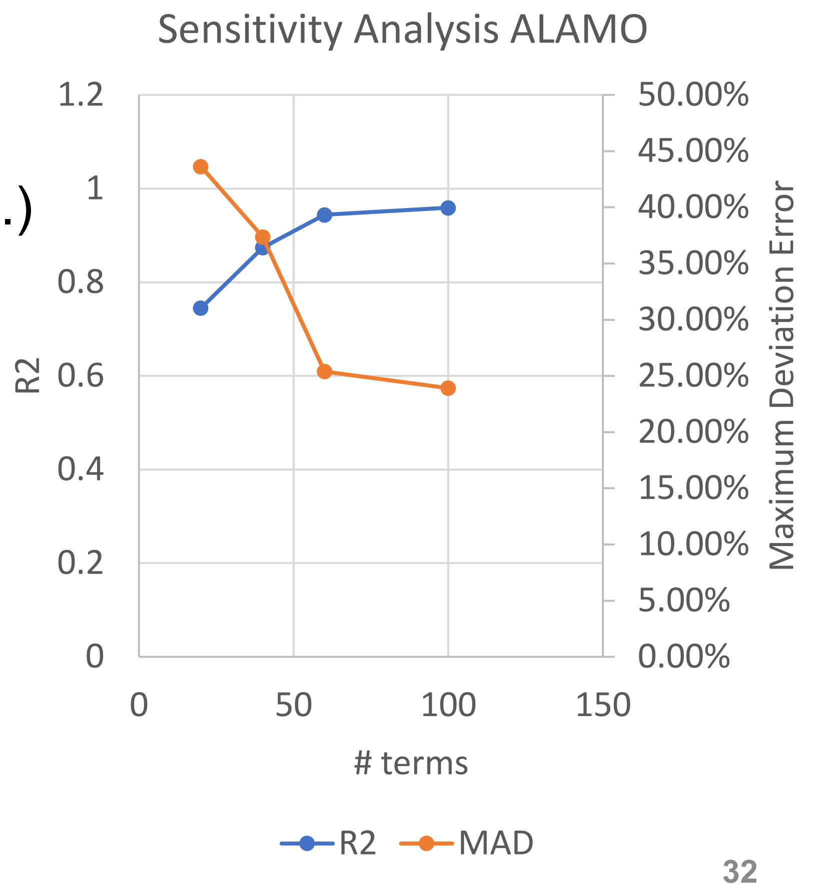
# 2.- Sampling “Best Practices”


- Identify samples within the Population
  - Remove “bad data”
  - Remove infeasible operating conditions
- Training data and Validation data

```
df = pd.read_excel(fname, sheet_name='data_set.xlsx')
n_data = df[input_labels[0]].size
df_training, df_validation = split_training_validation(df, 0.8, seed=n_data)
```

# 3.- Model Selection “Best Practices”


- ALAMO provides flexible models
  - Mallows CP, Bayesian Information Criterion (BIC), …
  - Flexible models (monomial powers, multi, ratio combinations, etc.)
  - Constrained regression and custom basis functions
  - Adaptive sampling (systematic approach to interrogate models)
- PySMO
  - Flexible models (polynomials, radial basis functions, kriging)
  - Native Pyomo tool (open-source)
- Neural Networks (open source tensorflow.Keras)
  - Correlated output variables
  - Flexible NN (Sigmoid, Relu, etc.)



# 4.- Model Verification/Validation


- The IDAES framework developed several tools for validating and verifying the surrogate models.
  - Plotting tools provide visual verification (parity, scatter errors, …)
  - Model statistics provide a good idea of model fitting and errors (R2, RMSE, MAD, etc.)

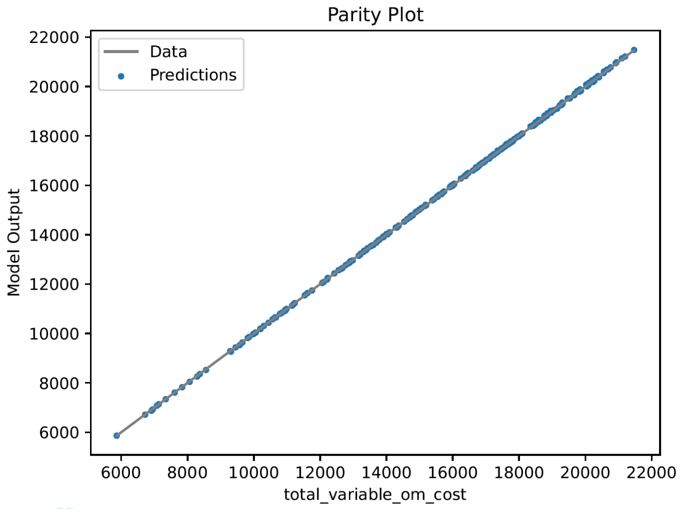
IDAES Institute for the Design of Advanced Energy Systems

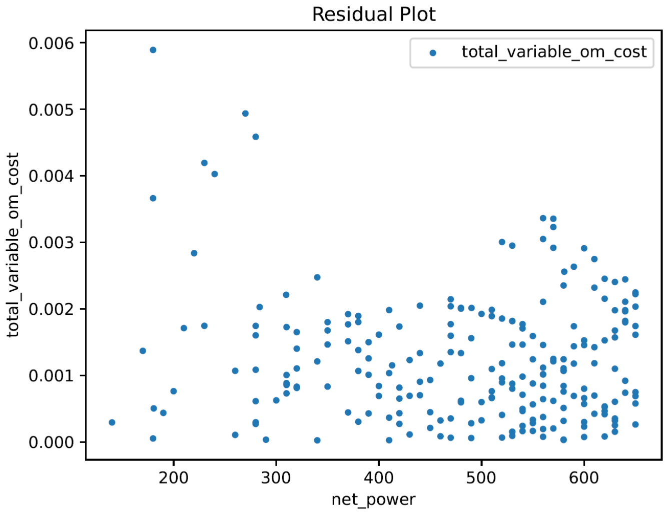
# 4.- Model Verification/Validation

```
df_training, df_validation = split_training_validation(df, 0.8, seed=n_data)
# Train surrogate (this will call to Alamo through AlamoPy)
success, alamo_surrogate, msg = trainer.train_surrogate()

# Display the metrics of fit
metrics = compute_fit_metrics(alamo_surrogate, df_validation)
# Generate residual and parity plots
surrogate_residual(alamo_surrogate, df_validation, filename=fname, relative_error=True)
surrogate_parity(alamo_surrogate, df_validation, filename=fname)
surrogate_residual(alamo_surrogate, df_training, filename=fname, relative_error=True)
surrogate_parity(alamo_surrogate, df_training, filename=fname)
```

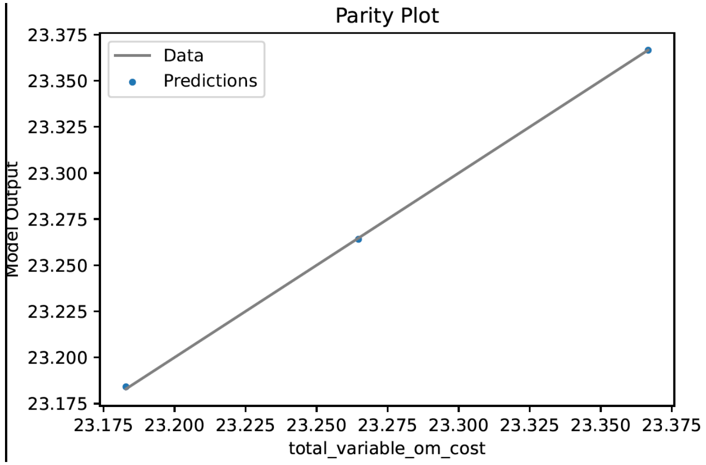
80% of data for training
20% for validation

## Training Data Set




## Independent Data Set



# 5.- Surrogates for Optimization


- Two main aspects: 1.- integrating the surrogate models in the mathematical model, and 2.- robustness of the final mathematical model.

Automated surrogate modeling implementation

```
# create the IDAES model
m = ConcreteModel()
m.fs = FlowsheetBlock(default={"dynamic": False})
m.variable_OM_costs = Var(initialize=20, doc='variable OM costs in $/MW')
m.net_power = Var(initialize=650, doc='net power or rated capacity in MW')
m.h2_production = Var(initialize=5, doc='hydrogen production in kg/s')
m.ng_flowrate = Var(initialize=27, doc="natural gas flowrate in kg/s")
m.surrogate = SurrogateBlock()
m.var_cost_d_hrs = Var(doc='variable OM costs in $/hr')
inputs = [m.net_power, m.h2_production]  # MW, kg/s
outputs = [m.var_cost_d_hrs, m.ng_flowrate]  # $/hr, kg/s
m.surrogate.build_model(alamo_surrogate,
                        input_vars=inputs,
                        output_vars=outputs)
m.power_con_min = Constraint(expr= m.net_power >= Pmin)
m.power_con_max = Constraint(expr= m.net_power <= Pmax)
```

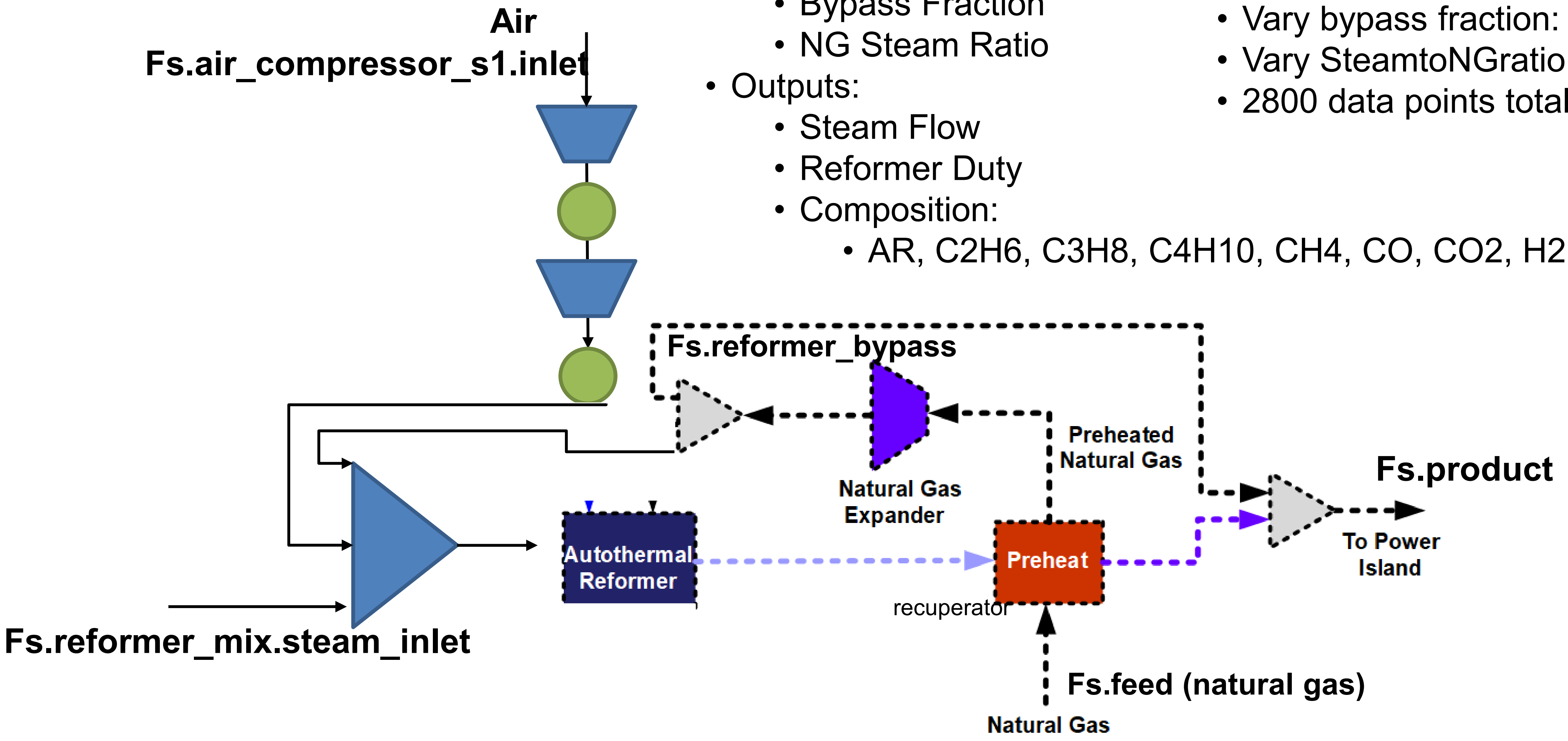
Conventional surrogate model implementation

```
import SRD

@self.Constraint(self.flowsheet().config.time,
                 doc="Specific reboiler duty in GJ/t CO2")
def SRD_eqn(b, t):
    A2 = b.lean_loading[t]
    B2 = b.Pz_mol[t]
    return b.SRD[t] == (1297.92749665023 * A2 + 124.190464480941 * B2
                        - 1144.97804719883 * A2**2 - 14.8595802759414
                        * B2**2 + 553.845056558601 * A2**3
                        + 0.847572796795006 * B2**3 - 626.741015730156
                        * (A2*B2)**0.5 + 128.236265690663 * A2*B2
                        - 6.92413056110739 * (A2*B2)**2 + 0.3261654983218
                        * (A2*B2)**3 - 643.204744175051 * A2/B2
                        - 1.69063104541704 * B2/A2 + 698.322562008984
                        * (A2/B2)**2 + 0.00474692417757652 * (B2/A2)**2)
```

# End to End Demonstration: Autothermal Reformer Example

- Inputs:
  - Bypass Fraction
  - NG Steam Ratio
- Outputs:
  - Steam Flow
  - Reformer Duty
  - Composition:
    - AR, C2H6, C3H8, C4H10, CH4, CO, CO2, H2, H2O, N2, O2

Data generated:
- Vary bypass fraction: 0.1-0.8
- Vary SteamtoNGratio: 0.8-1.2
- 2800 data points total



# Sampling “Best Practices”

Data Source → Sampling → Model Selection and Training → Visualization and Validation → Integrated IDAES model

IDAES Models and Properties

2800 samples (grid sampling method)

idaes.
surrogate.sampling

## ALAMO Python Wrapper

- Highly tractable models (flexible, minimizes model size)
- Allows model validation, as well as multi-input/output

## PySMO (Polynomial, RBF, Kriging)

- Restrictive model forms, but highly accurate
- Allows model validation, as well as multi-input (single output)

## Tensorflow Keras (Sequential)

- Option-dependent accuracy, *a priori* knowledge is helpful
- Free of expression-driven modeling – models are complex
- Sequential limited to single input/output layer (see Functional API)

# Sampling “Best Practices”

Data Source → Sampling → Model Selection and Training → Visualization and Validation → Integrated IDAES model

IDAES Models and Properties

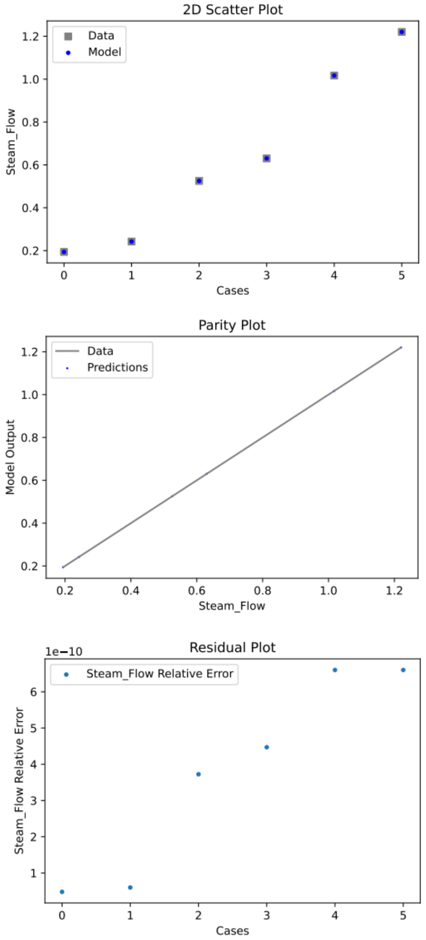
2800 samples (grid sampling method)

idaes.
surrogate.sampling

ALAMO Python Wrapper

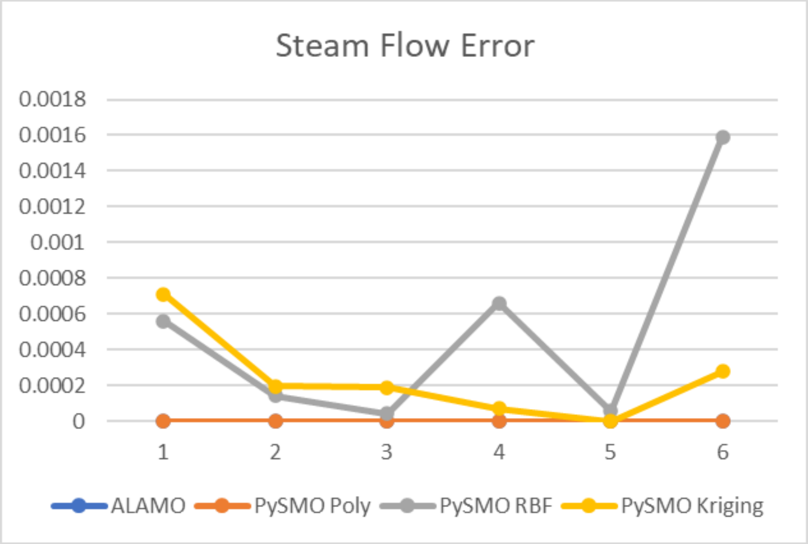
PySMO (Polynomial, RBF, Kriging)

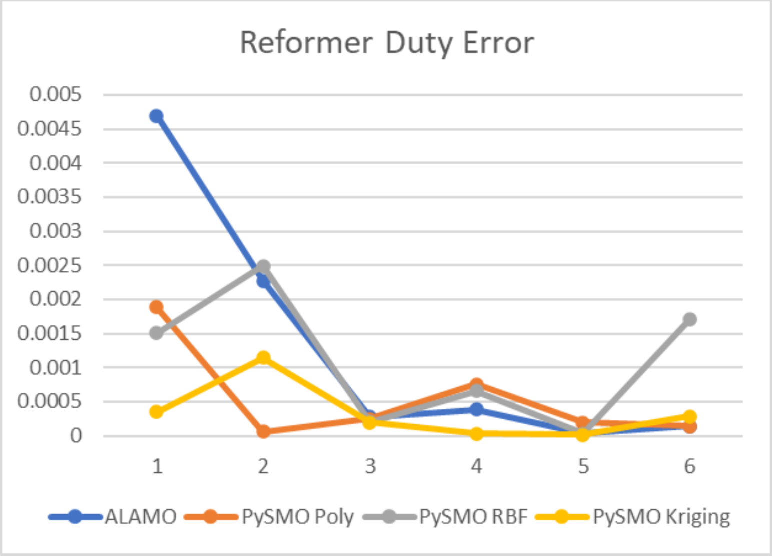
Tensorflow Keras (Sequential)

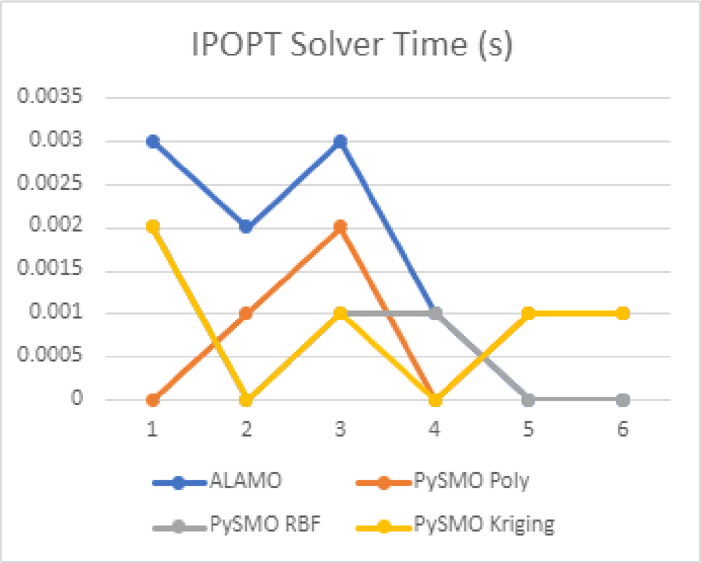



```python
# Import statements
import numpy as np
from AR_surrogate_trainer import ARtrainer
from idaes.surrogate.surrogate_block import SurrogateBlock
from idaes.core import FlowsheetBlock
from idaes.core.util.convergence.convergence_base import _run_ipopt_with_stats

# Import Pyomo libraries
from pyomo.environ import ConcreteModel, SolverFactory, value, Var

def build_flowsheet(case='case_0', out_vars='Steam_Flow'):

    # train and call the surrogate(s) we need from external module
    surrogate = ARtrainer(trainer='ALAMO', out_vars=out_vars, case=case)

    # create the IDAES model and flowsheet
    m = ConcreteModel()
    m.fs = FlowsheetBlock(default={"dynamic": False})

    # create and fix flowsheet input variables
    m.bypass_frac = Var(initialize=0.80, doc='natural gas bypass fraction')
    m.ng_steam_ratio = Var(initialize=0.80, doc='natural gas to steam ratio')

    # create flowsheet output variables
    m.steam_flowrate = Var(initialize=0.2, doc="steam flowrate")
    m.reformer_duty = Var(initialize=10000, doc="reformer heat duty")

    all_output_data = build_vars(m, case)  # fixes inputs and retrieves
    # output data (this is for all outputs from pre-defined dictionaries)
    output_data = dict(zip(out_vars, [] * len(out_vars)))  # pre-allocate
    for var in out_vars:  # we only need to return data for outputs of interest
        output_data[var] = all_output_data[var]

    inputs = [m.bypass_frac, m.ng_steam_ratio]
    outputs = [m.steam_flowrate, m.reformer_duty]

    # create the Pyomo/IDAES block that corresponds to the surrogate
    m.surrogate = SurrogateBlock()
    m.surrogate.build_model(surrogate, input_vars=inputs, output_vars=outputs)

    return m, output_data

if __name__ == "__main__":
    out_vars = ['Steam_Flow', 'Reformer_Duty']
    cases = np.transpose(np.array([i for i in range(6)], ndmin=2))
    zdata = np.empty((len(cases), len(out_vars)))
    zfit = np.empty((len(cases), len(out_vars)))
    ipopt_status, ipopt_solved, ipopt_iter, ipopt_time = [], [], [], []
    for i in range(len(cases)):  # there are cases 0 through 5
        case = 'case_' + str(i)
        m, output_data = build_flowsheet(case, trainer, basistype, out_vars)
        solver = SolverFactory('ipopt')
        [status_obj, solved, iters, time] = _run_ipopt_with_stats(m, solver)
        ipopt_time.append(time)
        zfit[i, 0] = value(m.steam_flowrate)
        zfit[i, 1] = value(m.reformer_duty)
        zdata[i, 0] = output_data['Steam_Flow']
        zdata[i, 1] = output_data['Reformer_Duty']
```

# Sampling “Best Practices”

# Examples of when and why to use surrogates?

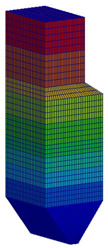
- Reducing model size, improve performance
  - Model more general than needed by a particular application (e.g., SOEC market analysis, auto-thermal reformer)
  - Opportunity to reduce model size and complexity
- Replacing “external” models (e.g., compiled code)
  - Convert external model to an EO model (e.g., fire side boiler, grid market surrogates)
  - Opportunity to obtain native EO model and improve performance
- Data-driven models
  - Data to replace missing first-principles model
- Improve convergence reliability
  - Push convergence challenges “offline” (controlled initialization)
  - Produce data from rigorous model – surrogate for application (e.g., auto-thermal reformer)
- Global optimization and MINLP (discrete + nonlinear)
  - Improve performance by replacing detailed nonlinear model
  - Surrogates may be lower order or piecewise linear

Tools for constructing and integrating surrogates in IDAES

ALAMO
a black-box modeling tool

HELMET

RIPE



Trust-region Approaches

PySMO

IDAES ML Surrogates

IDAES
Institute for the Design of Advanced Energy Systems

# IDAES Diagnostics Toolbox

Andrew Lee, Stakeholder Summit 2023

# Bad models are easy to write

- A model can only be valuable if you can solve it
- Good models take time and experience
  - We spend a significant amount of time debugging models
- One of the biggest limitations on the adoption of EO tools

# A Simple Example…

8 Variables:

- v1: [m]
- v2: [m]
- v3: $\in [0, 5]$
- v4:
- v5: $\in [0, 1]$
- v6:
- v7: [m], $\in [0, 1]$
- v8:

4 Constraints:

- c1: $v_1 + v_2 = 10$
- c2: $v_3 = v_4 + v_5$
- c3: $2v_3 = 3v_4 + 4v_5 + v_6$
- c4: $v_7 = 1 \times 10^{-8}\ v_1$

```
import pyomo.environ as pyo

m = pyo.ConcreteModel()

m.v1 = pyo.Var(units=pyo.units.m)
...bounds=(0, 1))
m.v8 = pyo.Var()

m.c1 = pyo.Constraint(expr=m.v1 + m.v2 == 10)
m.c2 = pyo.Constraint(expr=m.v3 == m.v4 + m.v5)
m.c3 = pyo.Constraint(expr=2*m.v3 == 3*m.v4 + 4*m.v5 + m.v6)
m.c4 = pyo.Constraint(expr=m.v7 == 1e-8*m.v1)
```

**Fix 3 Degrees of Freedom**

$v_4 = 2$ : `m.v4.fix(2)`
$v_5 = 2$ : `m.v5.fix(2)`
$v_6 = 0$ : `m.v6.fix(0)`

IDAES Institute for the Design of Advanced Energy Systems

# …and a Common Outcome

```
EXIT: Converged to a point of local infeasibility. Problem may be infeasible.
WARNING: Loading a SolverResults object with a warning status into
model.name="unknown";
    - termination condition: infeasible
    - message from solver: Ipopt 3.13.2\x3a Converged to a locally infeasible
      point. Problem may be infeasible.
```

# If Only There Was A Tool To Help…

```
from idaes.core.util import DiagnosticsToolbox

dt = DiagnosticsToolbox(m)

dt.report_structural_issues()
```

**Now Available**

**v2.2.0**

```
====================================================================================
Model Statistics

        Activated Blocks: 1 (Deactivated: 0)
        Free Variables in Activated Constraints: 4 (External: 0)
```


```
====================================================================================
Model Statistics

        Activated Blocks: 1 (Deactivated: 0)
        Free Variables in Activated Constraints: 4 (External: 0)
```


```
            Free Variables with only lower bounds: 0
            Free Variables with only upper bounds: 0
            Free Variables with upper and lower bounds: 2
        Fixed Variables in Activated Constraints: 3 (External: 0)
        Activated Equality Constraints: 4 (Deactivated: 0)
        Activated Inequality Constraints: 0 (Deactivated: 0)
        Activated Objectives: 0 (Deactivated: 0)

------------------------------------------------------------------------------------
```

```
Suggested next steps:

    display_components_with_inconsistent_units()
    display_underconstrained_set()
    display_overconstrained_set()

====================================================================================
```

```
====================================================================================
Model Statistics

        Activated Blocks: 1 (Deactivated: 0)
        Free Variables in Activated Constraints: 4 (External: 0)
            Free Variables with only lower bounds: 0
            Free Variables with only upper bounds: 0
            Free Variables with upper and lower bounds: 2
        Fixed Variables in Activated Constraints: 3 (External: 0)
        Activated Equality Constraints: 4 (Deactivated: 0)

------------------------------------------------------------------------------------
2 WARNINGS

    WARNING: 1 Component with inconsistent units
    WARNING: Structural singularity found
        Under-Constrained Set: 3 variables, 2 constraints
        Over-Constrained Set: 1 variables, 2 constraints

------------------------------------------------------------------------------------
    Caution: 1 variable fixed to 0
    Caution: 1 unused variable (0 fixed)

------------------------------------------------------------------------------------
Suggested next steps:

    display_components_with_inconsistent_units()
    display_underconstrained_set()
    display_overconstrained_set()

====================================================================================
```

```
====================================================================================
Model Statistics

        Activated Blocks: 1 (Deactivated: 0)
        Free Variables in Activated Constraints: 4 (External: 0)
            Free Variables with only lower bounds: 0
            Free Variables with only upper bounds: 0
            Free Variables with upper and lower bounds: 2
        Fixed Variables in Activated Constraints: 3 (External: 0)
        Activated Equality Constraints: 4 (Deactivated: 0)
        Activated Inequality Constraints: 0 (Deactivated: 0)
```

```
------------------------------------------------------------------------------------
2 Cautions

    Caution: 1 variable fixed to 0
    Caution: 1 unused variable (0 fixed)

------------------------------------------------------------------------------------
```


    Caution: 1 variable fixed to 0
    Caution: 1 unused variable (0 fixed)

------------------------------------------------------------------------------------


```
Suggested next steps:

    display_components_with_inconsistent_units()
    display_underconstrained_set()
    display_overconstrained_set()

====================================================================================
```

```
====================================================================================
Model Statistics

        Activated Blocks: 1 (Deactivated: 0)
        Free Variables in Activated Constraints: 4 (External: 0)
            Free Variables with only lower bounds: 0
            Free Variables with only upper bounds: 0
            Free Variables with upper and lower bounds: 2
        Fixed Variables in Activated Constraints: 3 (External: 0)
        Activated Equality Constraints: 4 (Deactivated: 0)
```

```
------------------------------------------------------------------------------------
Suggested next steps:

    display_components_with_inconsistent_units()
    display_underconstrained_set()
    display_overconstrained_set()

====================================================================================
```

```
    Caution: 1 variable fixed to 0
    Caution: 1 unused variable (0 fixed)
```


```
------------------------------------------------------------------------------------
Suggested next steps:

    display_components_with_inconsistent_units()
    display_underconstrained_set()
    display_overconstrained_set()
```

# Inconsistent Units

```
dt.display_components_with_inconsistent_units()

====================================================================================
The following component(s) have unit consistency issues:

    c1

For more details on unit inconsistencies, import the assert_units_consistent method
from pyomo.util.check_units
====================================================================================
```

c1: $v_1 + v_2 = 10$

**Constant has no units!**

$v_1$ - [m]

$v_2$ - [m]

# Structural Singularities

```
dt.display_overconstrained_set()

====================================================================================
Dulmage-Mendelsohn Over-Constrained Set

    Independent Block 0:

        Variables:

            v3

        Constraints:

            c2
            c3

====================================================================================
```

**2 Constraints, 1 Variable**
**-1 Degrees of Freedom!**

IDAES
Institute for the Design of Advanced Energy Systems

# Structural Singularities

```
dt.display_overconstrained_set()

================================================================================
Dulmage-Mendelsohn Over-Constrained Set

    Independent Block 0:

        Variables:

            v3

        Constraints:

            c2
            c3

================================================================================
```

c2: $v_3 = v_4 + v_5$

c3: $2v_3 = 3v_4 + 4v_5 + v_6$

$v_4 = 2, \; v_5 = 2, \; v_6 = 0$

**Fixed 1 Variable I shouldn't have:** m.v4.unfix()

IDAES Institute for the Design of Advanced Energy Systems

# Structural Singularities

```
dt.display_underconstrained_set()

================================================================================
Dulmage-Mendelsohn Under-Constrained Set

    Independent Block 0:

        Variables:

            v2
            v1
            v7

        Constraints:

            c1
            c4
================================================================================
```

**3 Variable, 2 Constraints**
**1 Degrees of Freedom!**

**Shows which variables I can fix**

**Fix 1 of these Variables:** $v_2 = 5$ : m.v2.fix(5)

# Lets Try Again…

```
====================================================================================
Model Statistics

        Activated Blocks: 1 (Deactivated: 0)
        Free Variables in Activated Constraints: 4 (External: 0)
            Free Variables with only lower bounds: 0
            Free Variables with only upper bounds: 0
            Free Variables with upper and lower bounds: 2
        Fixed Variables in Activated Constraints: 3 (External: 0)
```

```
------------------------------------------------------------------------------------
Suggested next steps:

    Try to initialize/solve your model and then call report_numerical_issues()

====================================================================================
```

```
2 Cautions

    Caution: 1 variable fixed to 0
    Caution: 1 unused variable (0 fixed)
```


```
------------------------------------------------------------------------------------
Suggested next steps:

    Try to initialize/solve your model and then call report_numerical_issues()

====================================================================================
```

# Still a Common Outcome…

```
EXIT: Converged to a point of local infeasibility. Problem may be infeasible.
WARNING: Loading a SolverResults object with a warning status into
model.name="unknown";
    - termination condition: infeasible
    - message from solver: Ipopt 3.13.2\x3a Converged to a locally infeasible
      point. Problem may be infeasible.
```

IDAES
Institute for the Design of Advanced Energy Systems

# Numerical Issues

```
====================================================================================
Model Statistics

    Jacobian Condition Number: 1.700E+01

------------------------------------------------------------------------------------
2 WARNINGS

    WARNING: 1 Constraint with large residuals
    WARNING: 2 Variables at or outside bounds

------------------------------------------------------------------------------------
    Caution: 1 extreme Jacobian Entry

------------------------------------------------------------------------------------
Suggested next steps:

    display_constraints_with_large_residuals()
    display_variables_at_or_outside_bounds()

====================================================================================
```

# Numerical Issues

```
====================================================================================
Model Statistics

    Jacobian Condition Number: 1.700E+01

------------------------------------------------------------------------------------
```

```
------------------------------------------------------------------------------------
Suggested next steps:

    display_constraints_with_large_residuals()
    display_variables_at_or_outside_bounds()

====================================================================================
```

```
    Caution: 1 extreme Jacobian Entry
```


```
------------------------------------------------------------------------------------
Suggested next steps:

    display_constraints_with_large_residuals()
    display_variables_at_or_outside_bounds()

====================================================================================
```

# Possible Bounds Violations

```
dt.display_variables_at_or_outside_bounds()

====================================================================================
The following variable(s) have values at or outside their bounds:

    v3 (free): value=0.0 bounds=(0, 5)
    v5 (fixed): value=2 bounds=(0, 1)

====================================================================================
```

**Two Issues**

- v3 : might be limited by lower bound, possibly infeasible
- v5 : value and bounds were set to incompatible values

**Relax Bounds**

- m.v3.setlb(-5)
- m.v5.setub(10)

IDAES
Institute for the Design of Advanced Energy Systems

# Try to Solve Again…

```
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr  ls
   0  0.0000000e+00 6.67e-01 0.00e+00  -1.0 0.00e+00    -  0.00e+00 0.00e+00   0
   1  0.0000000e+00 6.66e-03 2.97e+00  -1.0 2.00e+00    -  7.17e-01 9.90e-01h  1
   2  0.0000000e+00 6.27e-05 9.38e+00  -1.0 2.00e-02    -  1.00e+00 9.91e-01h  1
   3  0.0000000e+00 8.88e-16 1.13e-12  -1.0 1.88e-04    -  1.00e+00 1.00e+00h  1
```

```
Number of Iterations....: 3
```

```
EXIT: Optimal Solution Found.
```

An Optimal Solution **does not** mean the model is robust!

# Numerical Issues Again

```
====================================================================================
Model Statistics

    Jacobian Condition Number: 1.700E+01

------------------------------------------------------------------------------------
0 WARNINGS

    No warnings found!

------------------------------------------------------------------------------------
5 Cautions

    Caution: 1 Variable with value close to their bounds
    Caution: 1 Variable with value close to zero
    Caution: 1 Variable with extreme value
    Caution: 1 Variable with None value
    Caution: 1 extreme Jacobian Entry

------------------------------------------------------------------------------------
Suggested next steps:

    If you still have issues converging your model consider:
        svd_analysis(TBA)
        degeneracy_hunter (TBA)

====================================================================================
```

No Warnings

# Numerical Issues Again

```
====================================================================================
Model Statistics

    Jacobian Condition Number: 1.700E+01

------------------------------------------------------------------------------------
5 Cautions

    Caution: 1 Variable with value close to their bounds
    Caution: 1 Variable with value close to zero
    Caution: 1 Variable with extreme value
    Caution: 1 Variable with None value
    Caution: 1 extreme Jacobian Entry

------------------------------------------------------------------------------------
Suggested next steps:

    If you still have issues converging your model consider:
        svd_analysis(TBA)
        degeneracy_hunter (TBA)

====================================================================================
```

IDAES Institute for the Design of Advanced Energy Systems

# Variables with Extreme Values

```
dt.display_variables_with_extreme_values()

====================================================================================
The following variable(s) have extreme values:

    v7: 4.9999999999999945e-08

====================================================================================
```

**Scaling Issue:** v7 has very small value and is close to lower bound of 0

**Solution:** scale v7 and associated constraint c4

# Try to Solve Again…

```
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr  ls
   0  0.0000000e+00 1.40e+01 0.00e+00  -1.0 0.00e+00    -  0.00e+00 0.00e+00   0
   1  0.0000000e+00 3.55e-15 4.88e+02  -1.0 6.00e+00    -  2.02e-03 1.00e+00f  1
```

```
Number of Iterations....: 1


EXIT: Optimal Solution Found.
```

IDAES Institute for the Design of Advanced Energy Systems

# Conclusions

- Writing good models is still hard, but…
- Diagnostics Toolbox helps you find issues
  - Expert help at your fingertips
  - Reduces time, effort and expertise required to debug models
- For more information, see the poster

# All You Need to Remember…

```
from idaes.core.util import DiagnosticsToolbox

dt = DiagnosticsToolbox(m)

dt.report_structural_issues()
```

https://idaes-pse.readthedocs.io/en/stable/explanations/model_diagnostics/index.html

# IDAES Visualization

**Dan Gunter**, Sheng Pang, Sarah Poon, Cody O'Donnell

Stakeholder Summit 2023

# Visualization is important

IDAES models are built in Python code, but..

Complex models are routinely diagrammed

Validation: are the connections correct

Communication to others (and yourself)

Diagrams provide context for model properties

Stream values, unit values, constraints, structure

Visualization is much more than diagrams (of course)

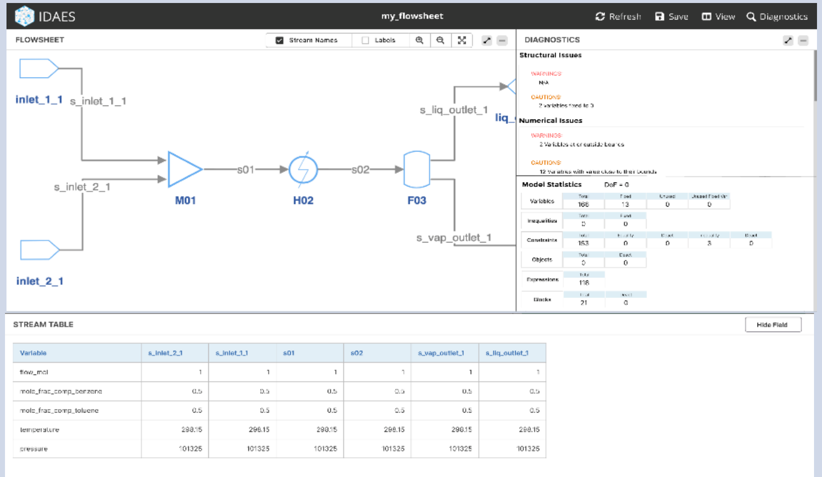
Plots of results, visual diagnostics, etc.

# IDAES has a visualization tool "built in"

With the addition of one line of code to your Jupyter Notebook or script..

```
model = build_model()
model.visualize("my_flowsheet")
```

..you get a web-based UI that automatically displays a model diagram and a stream table..

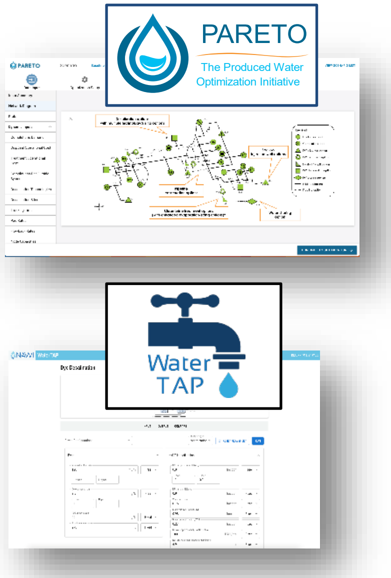



..and retains a connection to the model so you can interact with it.

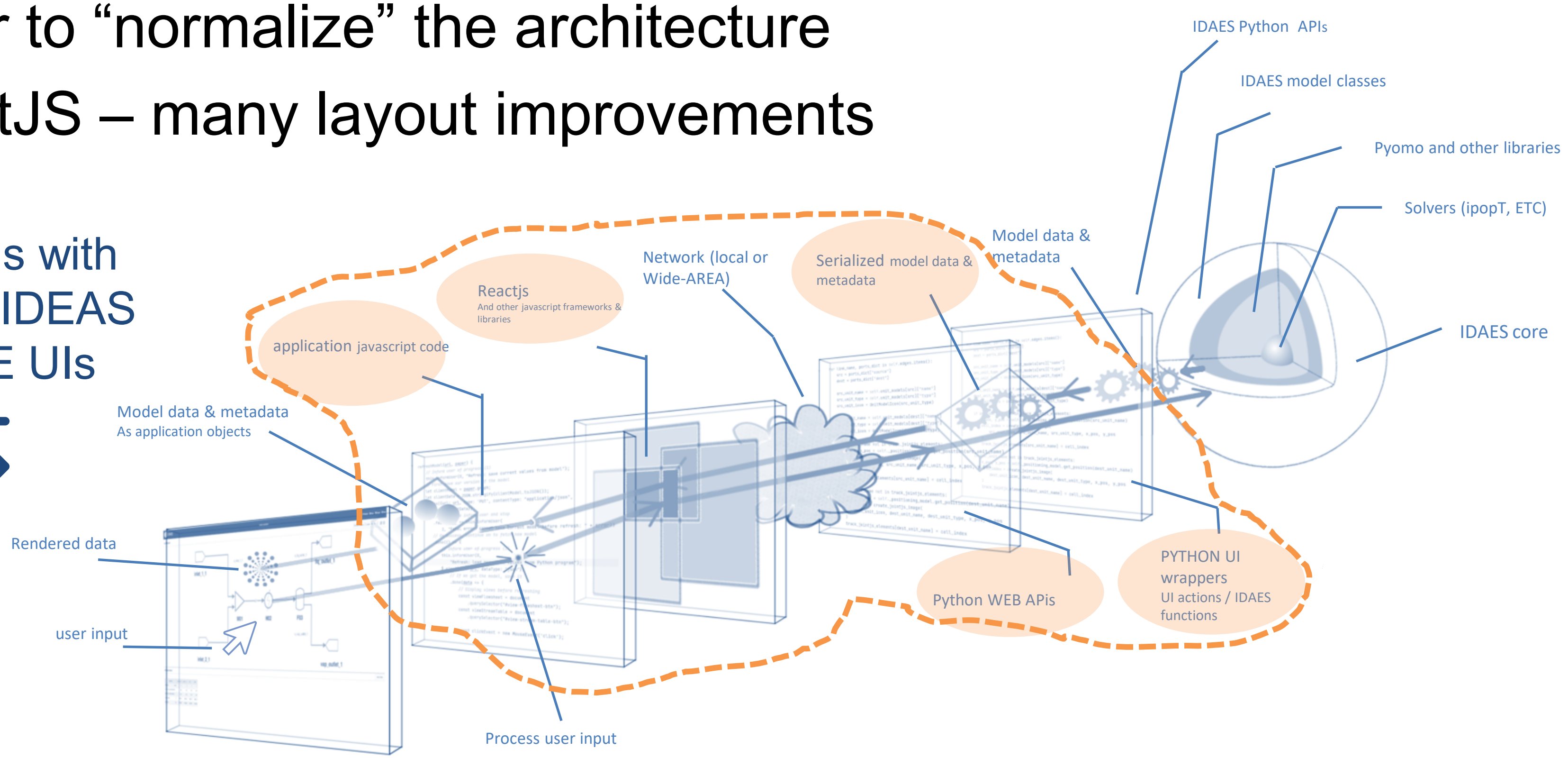
We call this tool the **Flowsheet Visualizer (FV)**

# Latest changes for the FV

- Important refactor to “normalize” the architecture
- Re-do UI in ReactJS – many layout improvements

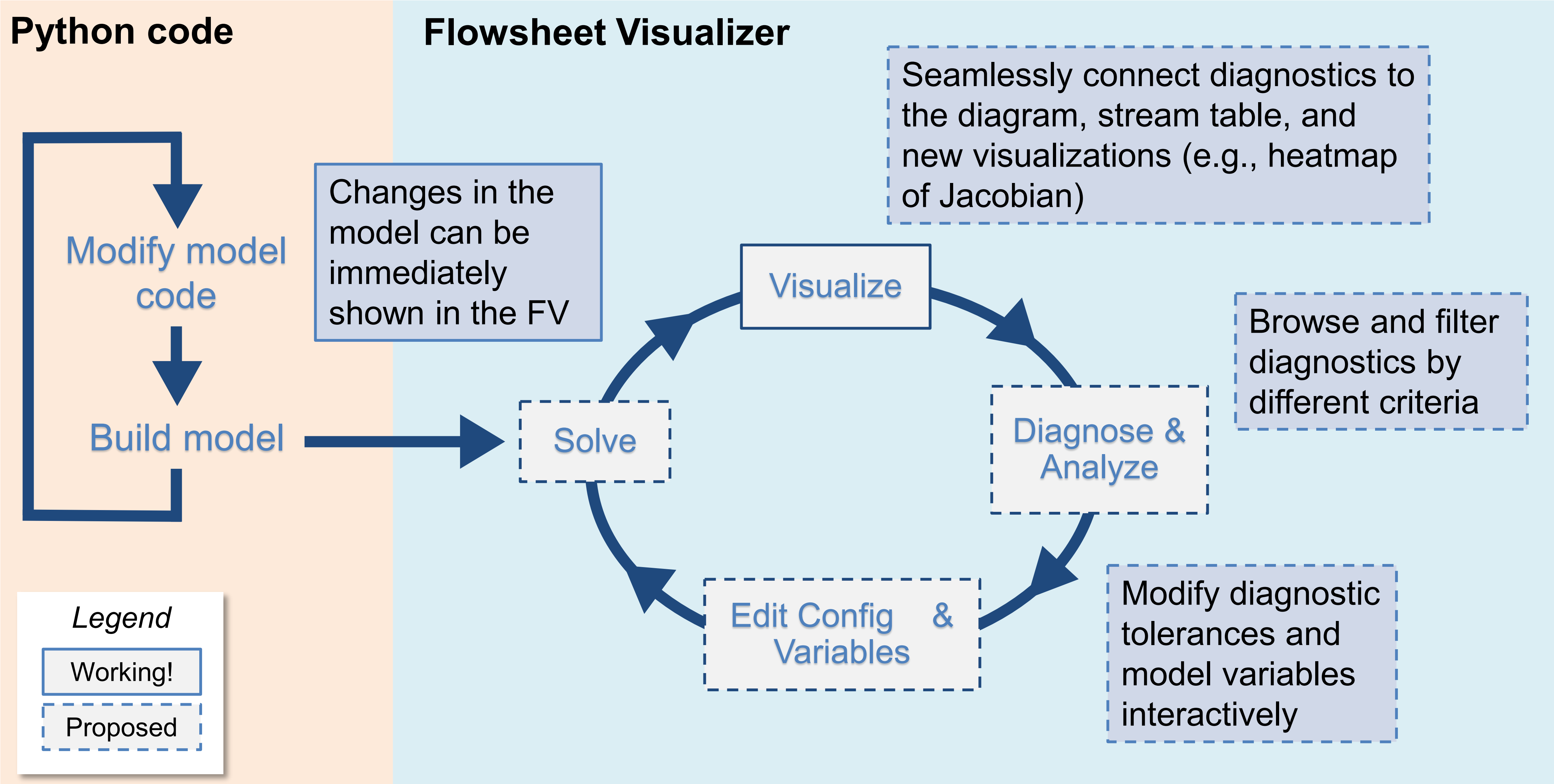



Aligns with other IDEAS PSE UIs




We are preparing the way for incorporating new, interactive elements, starting with: IDAES Diagnostics

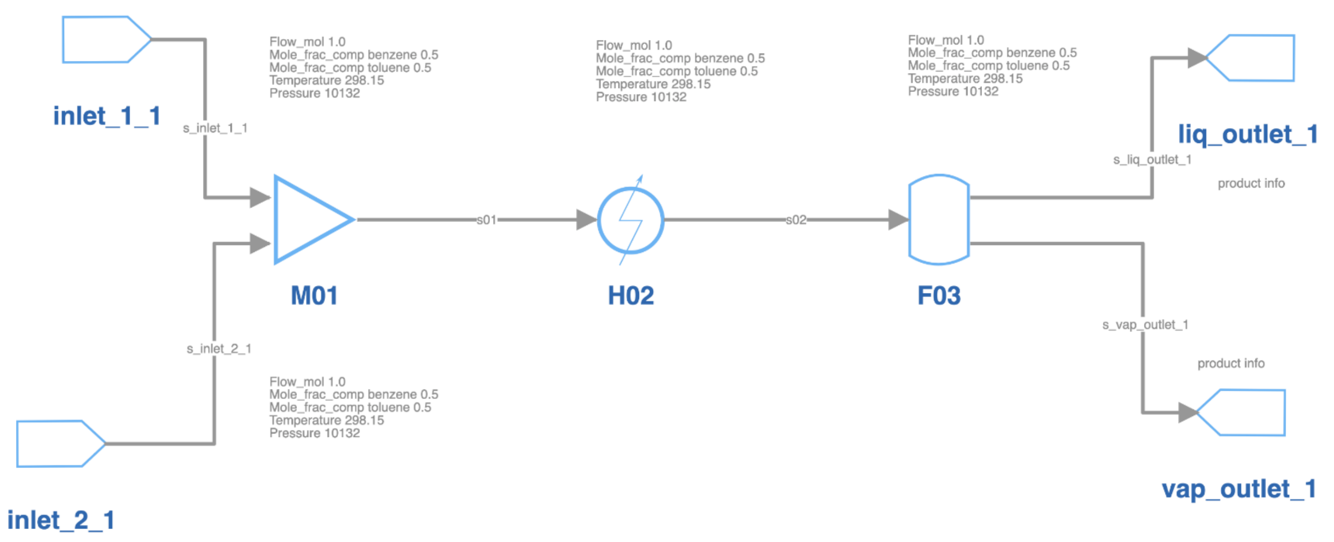
# Diagnostics and Plan/Do/Evaluate Cycle



# Diagnostics Prototype Screenshot

IDAES sample_visualization Refresh Save Export View Diagnostics

**FLOWSHEET** Stream Names Labels




**STREAM TABLE** Hide Fields

| Variable | | s_inlet_2_1 | s_inlet_1_1 | s01 | s02 |
|---|---|---|---|---|---|
| flow_mol | mol/s | 1 | 1 | 1 | 1 |
| mole_frac_comp benzene | – | 0.5 | 0.5 | 0.5 | 0.5 |
| mole_frac_comp toluene | – | 0.5 | 0.5 | 0.5 | 0.5 |
| temperature | K | 298.15 | 298.15 | 298.15 | 298.15 |
| pressure | Pa | 101325 | 101325 | 101325 | 101325 |

**DIAGNOSTICS**

**ISSUES** ▼

▶ 2 Caution

**CONFIG** ▼

| Name | Value |
|---|---|
| variable bounds absolute tolerance | 0.0001 |
| variable bounds relative tolerance | 0.0001 |
| variable bounds violation tolerance | 0 |
| constraint residual tolerance | 0.00001 |
| variable large value tolerance | 10000 |
| variable small value tolerance | 0.0001 |
| variable zero value tolerance | 1e-8 |
| jacobian large value caution | 10000 |
| jacobian large value warning | 100000000 |
| jacobian small value caution | 0.0001 |
| jacobian small value warning | 1e-8 |

**STATISTICS** ▼

| | | | | |
|---|---|---|---|---|
| DoF | Value: 0 | | | |
| Variables | Value: 166 | Unused: 166 | Fixed: 166 | Ineq: 166 |
| Inequalities | Value: 0 | | | |
| Constraints | Value: 153 | Eq: 153 | Ineq: 153 | |
| Objects | Value: 0 | Deact: 0 | | |
| Blocks | Value: 21 | Deact: 21 | | |
| Expressions | Value: 118 | | | |

IDAES Institute for the Design of Advanced Energy Systems

# Diagnostics Prototype Screenshot Detail

**DIAGNOSTICS**

**ISSUES ▼**

▶ 2 Caution

**CONFIG ▼**

| Name | Value |
|---|---|
| variable bounds absolute tolerance | 0.0001 |
| variable bounds relative tolerance | 0.0001 |
| variable bounds violation tolerance | 0 |
| constraint residual tolerance | 0.00001 |
| variable large value tolerance | 10000 |
| variable small value tolerance | 0.0001 |
| variable zero value tolerance | 1e-8 |
| jacobian large value caution | 10000 |
| jacobian large value warning | 100000000 |
| jacobian small value caution | 0.0001 |
| jacobian small value warning | 1e-8 |

**STATISTICS ▼**

| | | | | |
|---|---|---|---|---|
| DoF | Value: 0 | | | |
| Variables | Value: 166 | Unused: 166 | Fixed: 166 | Ineq: 166 |
| Inequalities | Value: 0 | | | |
| Constraints | Value: 153 | Eq: 153 | Ineq: 153 | |
| Objects | Value: 0 | Deact: 0 | | |
| Blocks | Value: 21 | Deact: 21 | | |
| Expressions | Value: 118 | | | |

IDAES
Institute for the Design of Advanced Energy Systems

# Browse to view cautions

**DIAGNOSTICS**

**ISSUES** ▼

- ▼ 2 Caution
  - ▼ Caution 1: variables with extreme values
    - ▶ extreme_values : 72
  - ▼ Caution 2: variables close to their bounds
    - ▶ var_near_bounds : 12

**CONFIG** ▼

| Name | Value |
|---|---|
| variable bounds absolute tolerance | 0.0001 |
| variable bounds relative tolerance | 0.0001 |
| variable bounds violation tolerance | 0 |
| constraint residual tolerance | 0.00001 |
| variable large value tolerance | 10000 |
| variable small value tolerance | 0.0001 |
| variable zero value tolerance | 1e-8 |
| jacobian large value caution | 10000 |
| jacobian large value warning | 100000000 |
| jacobian small value caution | 0.0001 |
| jacobian small value warning | 1e-8 |

**STATISTICS** ▼

| | | | | |
|---|---|---|---|---|
| DoF | Value: 0 | | | |
| Variables | Value: 166 | Unused: 166 | Fixed: 166 | Ineq: 166 |
| Inequalities | Value: 0 | | | |
| Constraints | Value: 153 | Eq: 153 | Ineq: 153 | |
| Objects | Value: 0 | Deact: 0 | | |
| Blocks | Value: 21 | Deact: 21 | | |
| Expressions | Value: 118 | | | |

# View variables with “extreme values”

**DIAGNOSTICS**

**ISSUES** ▼

▼ 2 Caution

▼ Caution 1: variables with extreme values

▼ extreme_values : 72

Search

| | |
|---|---|
| fs.M01.minimum_pressure[0.0,1] | 101324.99999999999 |
| fs.M01.minimum_pressure[0.0,2] | 101324.99999999997 |
| fs.M01.inlet_1_state[0.0].mole_frac_comp[toluene] | 0.00001 |
| fs.M01.inlet_1_state[0.0].pressure | 101325 |
| fs.M01.inlet_2_state[0.0].mole_frac_comp[benzene] | 0.00001 |
| fs.M01.inlet_2_state[0.0].pressure | 130000 |
| fs.M01.mixed_state[0.0].pressure | 101324.99999999997 |
| fs.M01.inlet_1_state[0.0].mole_frac_phase_comp[Liq,toluene] | 0.000004226501650736347 |
| fs.M01.inlet_1_state[0.0].mole_frac_phase_comp[Vap,toluene] | 0.0000016239937390247786 |
| fs.M01.inlet_1_state[0.0].pressure_sat_comp[benzene] | 101327.63707622685 |
| fs.M01.inlet_1_state[0.0].pressure_sat_comp[toluene] | 38933.18380178968 |
| fs.M01.inlet_1_state[0.0].enth_mol_phase[Liq] | 59403.464839923574 |
| fs.M01.inlet_1_state[0.0].enth_mol_phase[Vap] | 89602.98646808404 |
| fs.M01.inlet_1_state[0.0].enth_mol_phase_comp[Liq,benzene] | 59404.366946131 |
| fs.M01.inlet_1_state[0.0].enth_mol_phase_comp[Liq,toluene] | 24005.117152243132 |
| fs.M01.inlet_1_state[0.0].enth_mol_phase_comp[Vap,benzene] | 89602.59458088383 |
| fs.M01.inlet_1_state[0.0].enth_mol_phase_comp[Vap,toluene] | 58559.44869427196 |
| fs.M01.inlet_2_state[0.0].mole_frac_phase_comp[Liq,benzene] | 0.0000018948688821340677 |
| fs.M01.inlet_2_state[0.0].mole_frac_phase_comp[Vap,benzene] | 0.000004334161298390463 |
| fs.M01.inlet_2_state[0.0].pressure_sat_comp[benzene] | 297350.9006893184 |
| fs.M01.inlet_2_state[0.0].pressure_sat_comp[toluene] | 129996.82889148161 |
| fs.M01.inlet_2_state[0.0].enth_mol_phase[Liq] | 25813.44309761046 |
| fs.M01.inlet_2_state[0.0].enth_mol_phase[Vap] | 59876.807741757904 |
| fs.M01.inlet_2_state[0.0].enth_mol_phase_comp[Liq,benzene] | 60968.170946131 |
| fs.M01.inlet_2_state[0.0].enth_mol_phase_comp[Liq,toluene] | 25812.518818909797 |
| fs.M01.inlet_2_state[0.0].enth_mol_phase_comp[Vap,benzene] | 90653.83510796717 |
| fs.M01.inlet_2_state[0.0].enth_mol_phase_comp[Vap,toluene] | 59874.87498323029 |
| fs.M01.mixed_state[0.0].pressure_sat_comp[benzene] | 156860.8353206145 |
| fs.M01.mixed_state[0.0].pressure_sat_comp[toluene] | 63534.71863733716 |
| fs.M01.mixed_state[0.0].enth_mol_phase[Liq] | 38039.7329520597 |
| fs.M01.mixed_state[0.0].enth_mol_phase[Vap] | 77818.48958992377 |
| fs.M01.mixed_state[0.0].enth_mol_phase_comp[Liq,benzene] | 59130.24754889732 |
| fs.M01.mixed_state[0.0].enth_mol_phase_comp[Liq,toluene] | 23688.342072640477 |
| fs.M01.mixed_state[0.0].enth_mol_phase_comp[Vap,benzene] | 89419.03516967919 |
| fs.M01.mixed_state[0.0].enth_mol_phase_comp[Vap,toluene] | 58329.53049329868 |

# Filter variables by name (etc.)

**DIAGNOSTICS**

**ISSUES** ▼

▼ 2 Caution

▼ Caution 1: variables with extreme values

▼ extreme_values : 72

Filter: fs.M01.inlet_1_state[0.0]

| | |
|---|---|
| fs.M01.inlet_1_state[0.0].mole_frac_comp[toluene] | 0.00001 |
| fs.M01.inlet_1_state[0.0].pressure | 101325 |
| fs.M01.inlet_1_state[0.0].mole_frac_phase_comp[Liq,toluene] | 0.00004226501650736347 |
| fs.M01.inlet_1_state[0.0].mole_frac_phase_comp[Vap,toluene] | 0.000016239937390247786 |
| fs.M01.inlet_1_state[0.0].pressure_sat_comp[benzene] | 101327.63707622685 |
| fs.M01.inlet_1_state[0.0].pressure_sat_comp[toluene] | 38933.18380178968 |
| fs.M01.inlet_1_state[0.0].enth_mol_phase[Liq] | 59403.464839923574 |
| fs.M01.inlet_1_state[0.0].enth_mol_phase[Vap] | 89602.98646808404 |
| fs.M01.inlet_1_state[0.0].enth_mol_phase_comp[Liq,benzene] | 59404.366946131 |
| fs.M01.inlet_1_state[0.0].enth_mol_phase_comp[Liq,toluene] | 24005.117152243132 |
| fs.M01.inlet_1_state[0.0].enth_mol_phase_comp[Vap,benzene] | 89602.59458088383 |
| fs.M01.inlet_1_state[0.0].enth_mol_phase_comp[Vap,toluene] | 58559.44869427196 |

▼ Caution 2: variables close to their bounds

▶ var_near_bounds : 12

Search

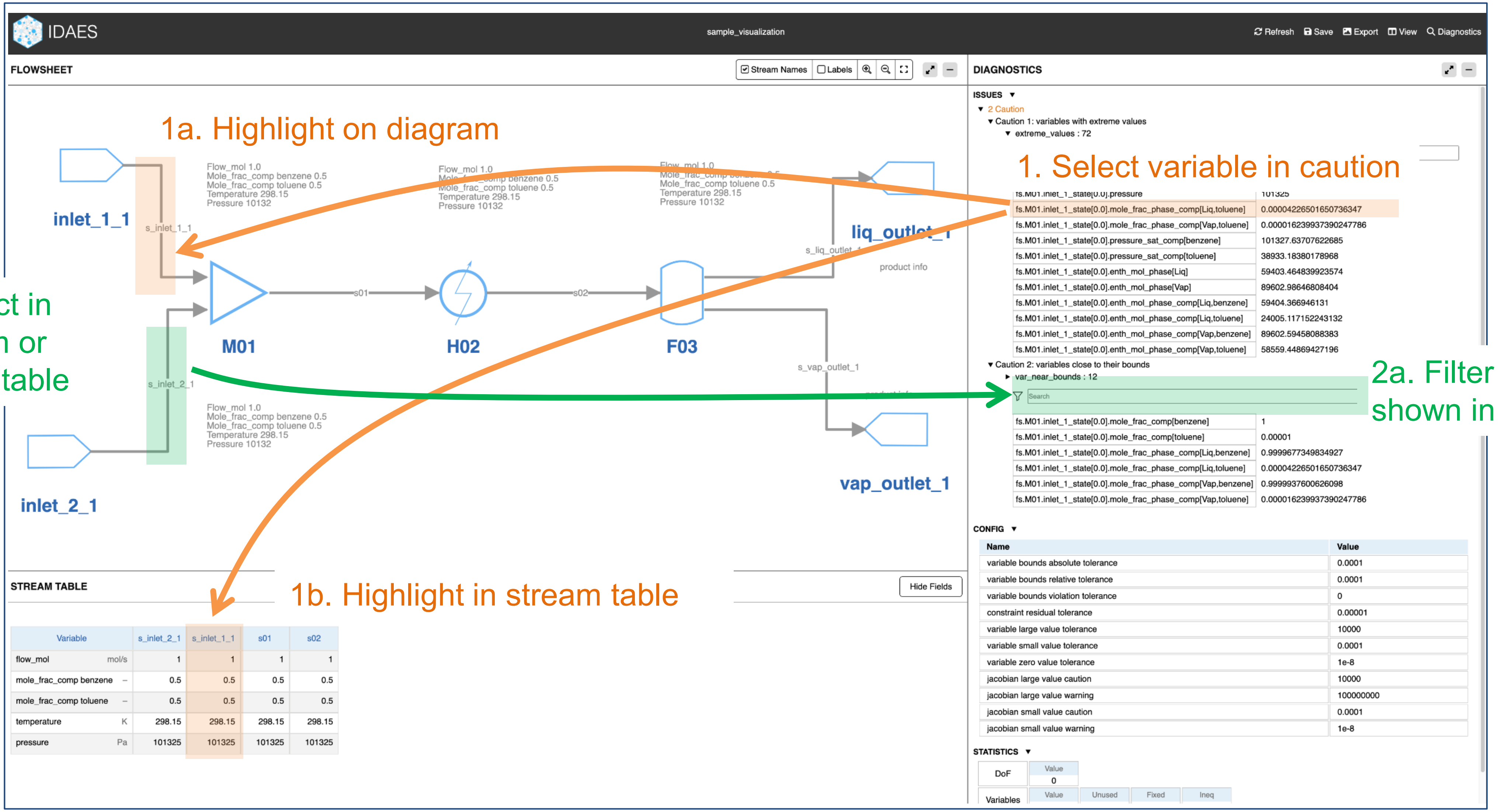
| | |
|---|---|
| fs.M01.inlet_1_state[0.0].mole_frac_comp[benzene] | 1 |
| fs.M01.inlet_1_state[0.0].mole_frac_comp[toluene] | 0.00001 |
| fs.M01.inlet_1_state[0.0].mole_frac_phase_comp[Liq,benzene] | 0.9999677349834927 |
| fs.M01.inlet_1_state[0.0].mole_frac_phase_comp[Liq,toluene] | 0.00004226501650736347 |
| fs.M01.inlet_1_state[0.0].mole_frac_phase_comp[Vap,benzene] | 0.9999937600626098 |
| fs.M01.inlet_1_state[0.0].mole_frac_phase_comp[Vap,toluene] | 0.000016239937390247786 |

**CONFIG** ▼

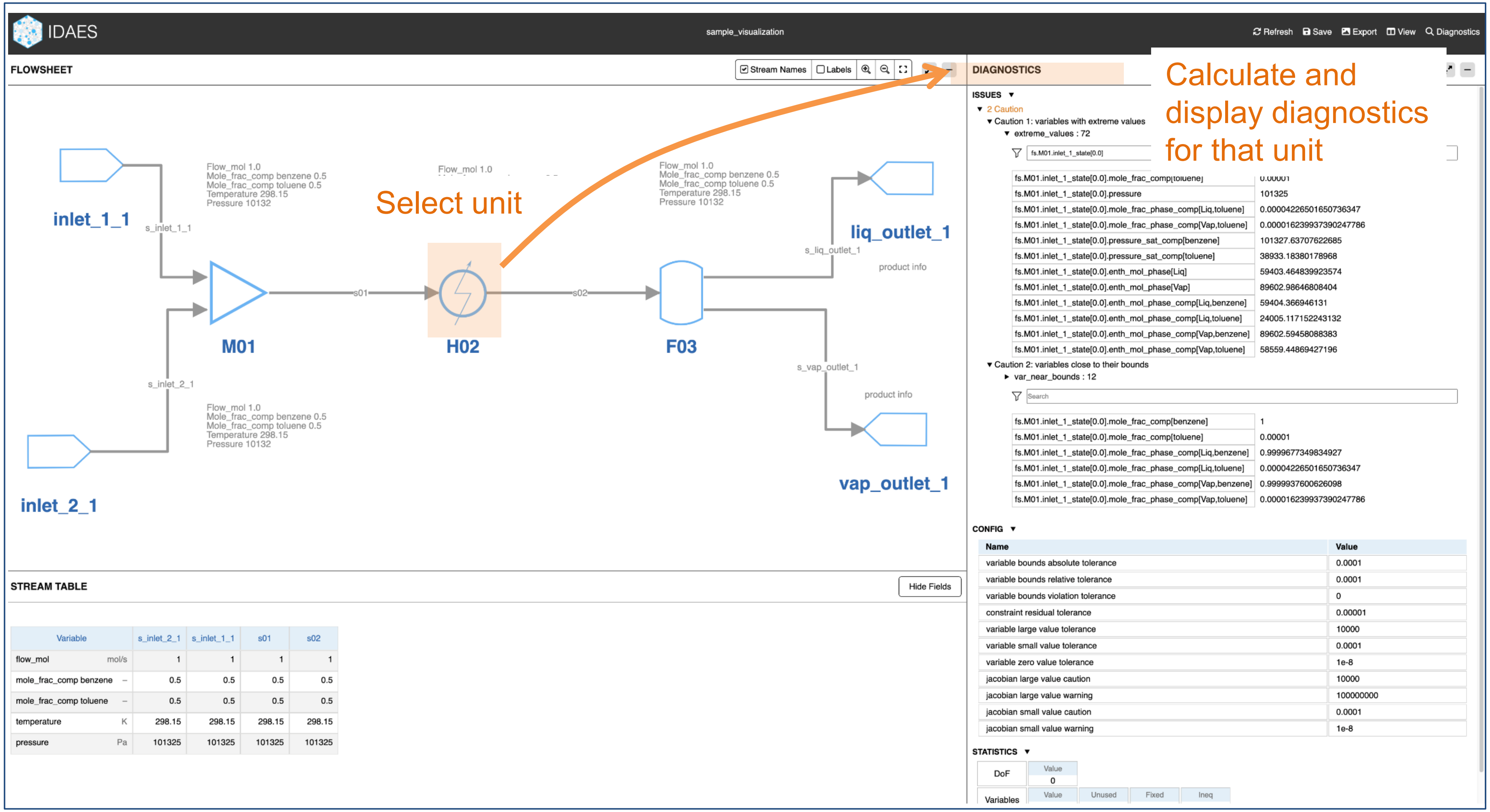
| Name | Value |
|---|---|
| variable bounds absolute tolerance | 0.0001 |
| variable bounds relative tolerance | 0.0001 |

IDAES
Institute for the Design of Advanced Energy Systems

# Advanced interactive explorations

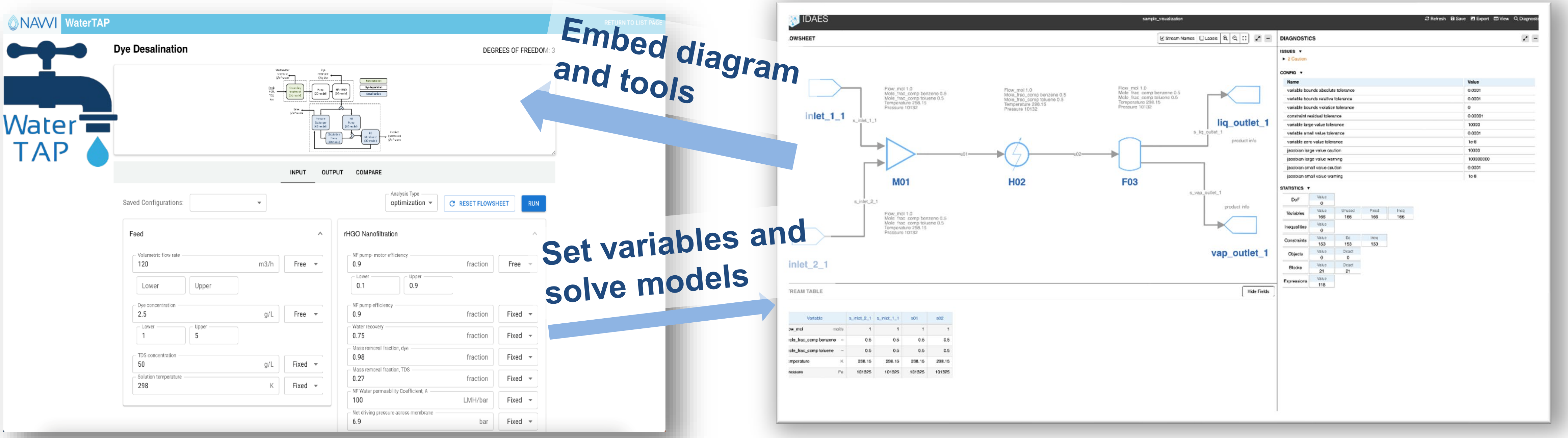


# Advanced interactive explorations (more..)



# Integration with other PSE projects

- All the capabilities of the Flowsheet Visualizer can be potentially embedded into other Web UIs
- Capabilities being developed in e.g. WaterTAP can be “ported” to the FV
- Beginnings of a PSE UI “ecosystem”..?



# Summary

The Flowsheet Visualizer (FV) can easily visualize IDAES models today

Interactive access to IDAES Diagnostics Toolkit is being actively added

This will make the FV a more useful tool across IDAES PSE

IDAES
Institute for the Design of Advanced Energy Systems

# Please come talk to us! We want your feedback

Dan Gunter, team lead

Sheng Pang, IDAES UI developer

Sarah Poon, User Experience (UX)

Mike Pesce, UI developer
WaterTAP, PARETO

*Cody O'Donnell, IDAS UI designer (emeritus)*

# Acknowledgements

**The IDAES team gratefully acknowledges support from the U.S. DOE's Office of Fossil Energy and Carbon Management through the Simulation-Based Engineering/Crosscutting Research Program.**

**National Energy Technology Laboratory**: David Miller, Tony Burgard, John Eslick, **Andrew Lee**, **Miguel Zamarripa**, Jinliang Ma, Dale Keairns, Jaffer Ghouse, Ben Omell, Chinedu Okoli, Richard Newby, Arun Iyengar, Anca Ostace, Steve Zitney, Anuja Deshpande, Alex Noring, Naresh Susarla, Radhakrishna Gooty, **Doug Allen**, Ryan Hughes, Andres Calderon, **Brandon Paul**, Adam Atia, John Brewer, Nadejda Victor, Maojian Wang, Peng Liu

**Sandia National Laboratories**: **John Siirola**, **Bethany Nicholson**, **Michael Bynum**, Jordan Jalving, Emma Johnson, Katherine Klise, Shawn Martin, **Miranda Mundt**, **Edna Soraya Rawlings**, Kyle Skolfield

**Lawrence Berkeley National Laboratory**: Deb Agarwal, **Dan Gunter**, Keith Beattie, John Shinn, Hamdy Elgammal, Joshua Boverhof, Karen Whitenack, Oluwamayowa Amusat, **Sarah Poon**

**Carnegie Mellon University**: Larry Biegler, Chrysanthos Gounaris, Ignacio Grossmann, Carl Laird, John Eason, Owais Sarwar, Natalie Isenberg, Chris Hanselman, Marissa Engle, Qi Chen, Cristiana Lara, **Robert Parker**, Ben Sauk, Vibhav Dabadghao, Can Li, David Molina Thierry, Mingrui Li, Seolhee Cho, Georgia Stinchfield, Jason Sherman

**West Virginia University**: Debangsu Bhattacharyya, Paul Akula, Quang-Minh Le, Nishant Giridhar, Matthew Alastanos

**University of Notre Dame**: **Alexander Dowling**, Xian Gao, Nicole Cortes

**Georgia Tech**: Nick Sahinidis, Yijiang Li, Selin Bayramoglu



IDAES Institute for the Design of Advanced Energy Systems