INL/MIS-22-66312

IES
Integrated Energy Systems

# FORCE – Transient Physical Modeling Workshop

## Hybrid Model Development

**Presented by: Dr. Daniel Mikkelson**
**Prepared by:**
**Dr. Daniel Mikkelson and Dr. Konor Frick**

# Session Agenda

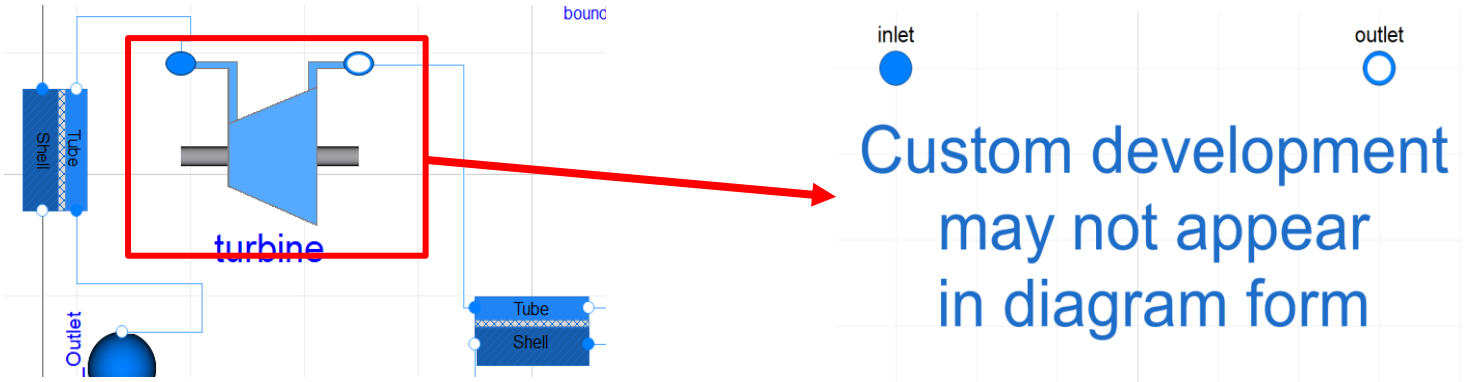
1. Component model development (30 min)
   a) Defining variables with units
   b) Using functions
   c) Figure design
   d) UI terminology
2. Additional model enhancement (10 min)
   a) Add features to integrated models
3. Controls (10 min)
   a) Hybrid standard sensor/actuator buses
4. How is information conveyed to and from Hybrid? (10 min)
   a) .mat standard output
   b) Text file reading and writing
   c) Making easily accessible data
   d) Scripting



IES Integrated Energy Systems

# Component Model Development

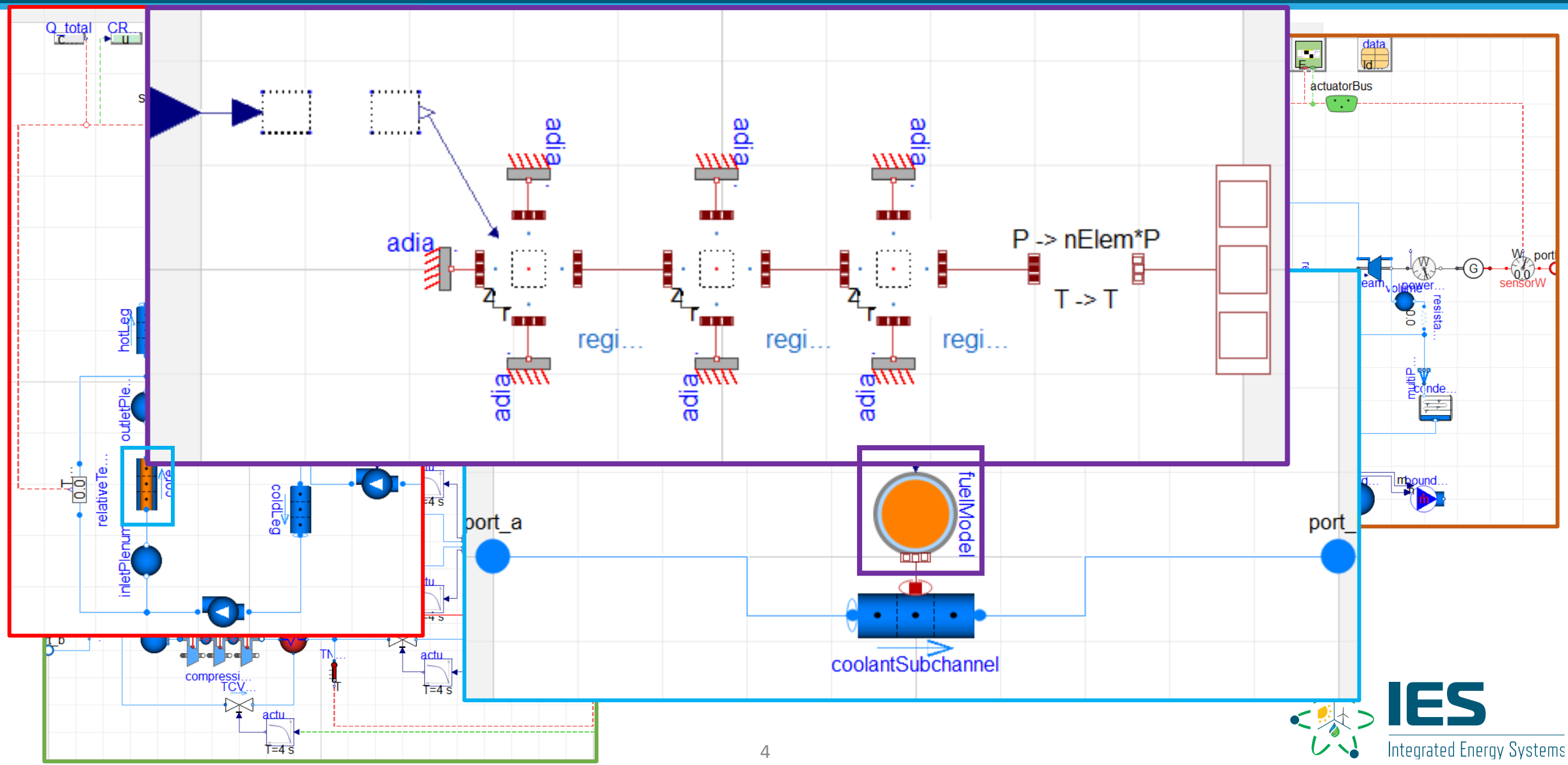
- Final Modelica systems are typically interconnected instantiations of developed models
- In the drag-and-drop GUI, objects use connectors to complete the set of equations
  - Connectors “flow” a variable and “stream” its characteristics along its path
  - Connectors must be properly internally connected
- Individual models are typically self-contained, needing only port connections to be well-posed






IES Integrated Energy Systems

# Component Model Development






IES Integrated Energy Systems

# Component Model Development

- Component models can be built in various ways
  - Example: Simplified heat exchanger (NTU, LMTD)
  - Allows for custom physics, assumptions
  - Goal should always be to code as reconfigurable and replaceable as possible
    - Non drag-and-drop models do tend to be more application-specific

- Extension of basic physics packages possible
  - Example: Custom valves
  - Allows for new iterations of other models
  - Ensures appropriate connection of ports and desired base physics packages

IES
Integrated Energy Systems

# Component Model Development

- Variable types you may use within Modelica are:
  - Integer
  - Real
  - Boolean
  - String
  - Package
    - Technically a collection of variables, but typically declared in variable section
- Where possible, units are used within Hybrid
  - Some variable type declarations include units along with them
- Keywords:
  - Parameter
  - Constant
  - Replaceable – commonly used with package declarations
  - Input – typically used in functions
  - Output – typically used in functions
- Modelica operator “der()”

# Component Modeling – NTU HX

- An NTU heat exchanger is an efficiency-based heat exchanger requiring no “real” geometric data (no pipe lengths, number of pipes, etc.)

$$Q_{NTU} = Q_{max} \frac{1 - e^{-NTU(1-C_R)}}{1 - C_R e^{-NTU(1-C_R)}}$$

- Need to calculate $Q_{max}$ and $C_R$ to find heat rate
- Once calculation is complete, a method of object integration needs to be developed

# Component Modeling – NTU HX

```
Modelica.Units.SI.Power Q
  "Power source/sink in each of the heat transfer volumes";
Modelica.Units.SI.Power Q_total "Total actual HX power";
Modelica.Units.SI.Power Q_max "Minimum of Q_max_s and Q_max_t";
Modelica.Units.SI.Power Q_max_s
  "Max Q of shell side assuming reaches inlet tube temperature";
Modelica.Units.SI.Power Q_max_t
  "Max Q of tube side assuming reaches inlet shell temperature";
Modelica.Units.SI.Power Q_min "This value is used in Cr calculations";
Real Cr(start = Cr_init) "Mass*heatcapacity ratio of the HX, minimum divided by maximum.
                          In this model, it's calculated using massflow*deltaEnthalpy";

parameter Real K_tube(unit = "1/m4") "Geometry-less pressure loss coefficient";
parameter Real K_shell(unit = "1/m4") "Geometry-less pressure loss coefficient";

replaceable package Tube_medium = Modelica.Media.Water.StandardWater ;
replaceable package Shell_medium = Modelica.Media.Water.StandardWater ;

parameter Real NTU = 4 "Characteristic NTU of HX" annotation(Dialog(tab="General", group="Sizing"));
```



IES Integrated Energy Systems

# Component Modeling – NTU HX

```
TRANSFORM.Fluid.Interfaces.FluidPort_State Tube_out(redeclare package Medium = Tube_medium)  a ;
TRANSFORM.Fluid.Interfaces.FluidPort_Flow  Tube_in (redeclare package Medium =  Tube_medium)  a ;
TRANSFORM.Fluid.Interfaces.FluidPort_Flow  Shell_in(redeclare package Medium = Shell_medium)  a ;
TRANSFORM.Fluid.Interfaces.FluidPort_State Shell_out(redeclare package Medium =Shell_medium)  a ;
TRANSFORM.Fluid.Volumes.MixingVolume Shell(
  redeclare package Medium = Shell_medium,
  p_start=p_start_shell - dp_general,
  use_T_start=false,
  h_start=0.5*h_start_shell_outlet + 0.5*h_start_shell_inlet,
  redeclare model Geometry =
      TRANSFORM.Fluid.ClosureRelations.Geometry.Models.LumpedVolume.GenericVolume
      (V=V_Shell/2, angle=0, dheight=dh_Shell/2), use_HeatPort=false,
  Q_gen=-Q,
  nPorts_a=1,
  nPorts_b=1)
  a ;
```

Text view of drag-and-drop components instantiated

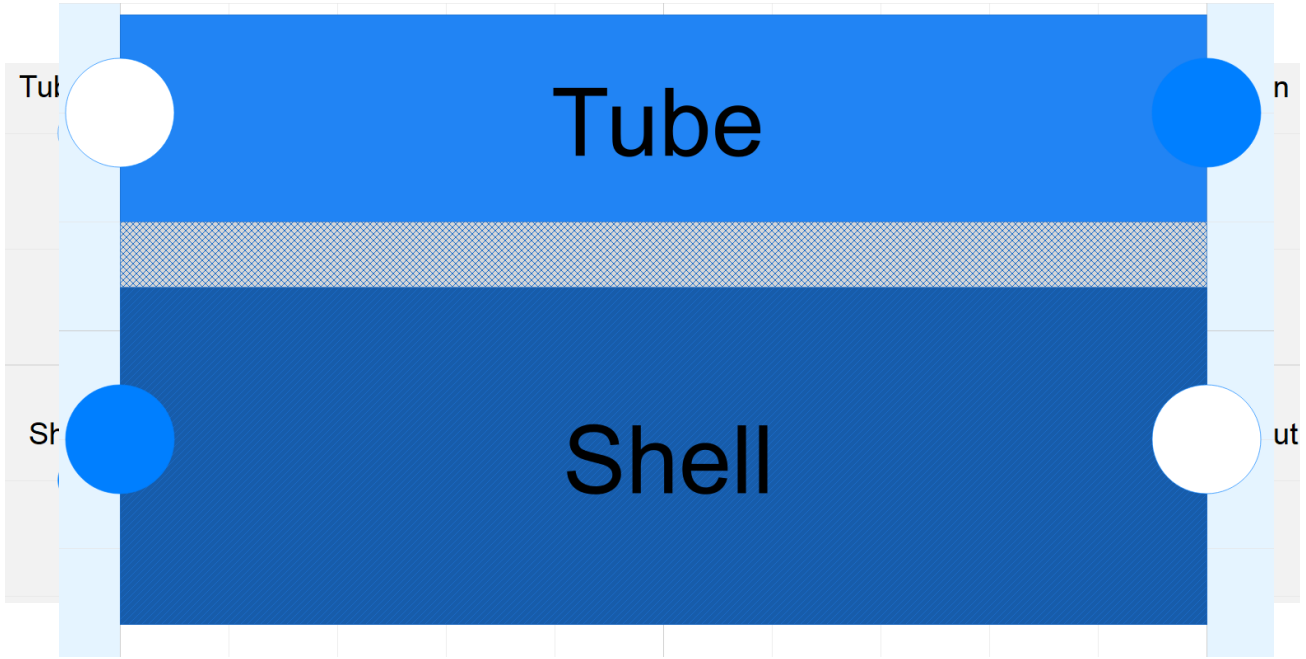
Package redeclaration at this level either passes the parameter package down to the next level of model instantiation (Tube_medium, Shell_medium) or evaluates it on this level (Geometry model)

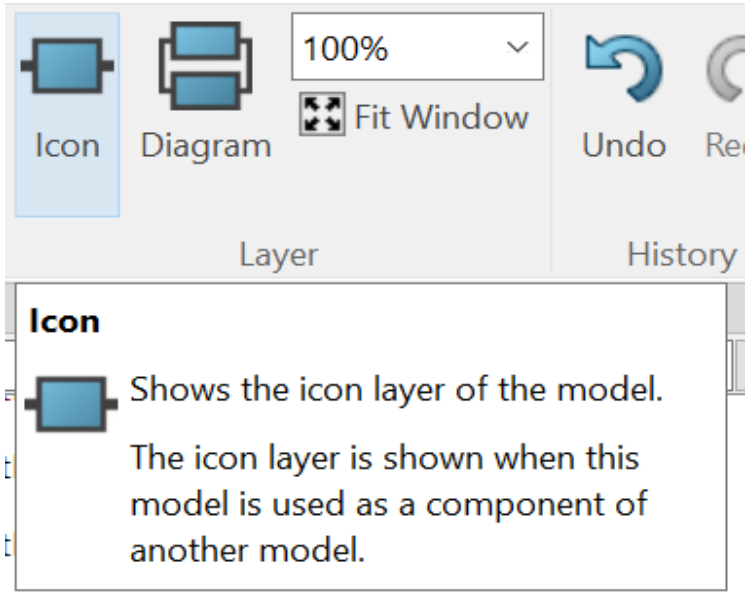
# Component Modeling – NTU HX

```
Q_max_s =abs(Shell_dp.m_flow)* -Shell_medium.specificEnthalpy_pT(Shell.medium.state.p,Tin_t) + hin_s);
Q_max_t = abs(Tube_dp.m_flow)* Tube_medium.specificEnthalpy_pT(Tube_dp.state.p,Tin_s) - hin_t);
//By using absolute values, the lesser positive or negative Q becomes the more limiting side.
  if(abs(Q_max_s) < abs(Q_max_t)) then
  Q_max = Q_max_s;
  Q_min = Q_max_t;
else
  Q_max = Q_max_t;
  Q_min = Q_max_s;
end if;
// "Calculate the theoretical exit enthalpy assuming that there is some form of boiling or condensation
//Including this in an algorithm appears to allow for more of the model to be differentiable based on t
//
he
Te
Tex_s = Shell_medium.temperature_ph(Shell_out.p,hex_s);
hex_s = hin_s - Q_max*(1-exp(-NTU))/(abs(Shell_dp.m_flow));
xchange_t = NHES.Fluid.HeatExchangers.Utilities.Functions.quality(hin_t,hf_t,hg_t)
- NHES.Fluid.HeatExchangers.Utilities.Functions.quality(hex_t,hf_t,hg_t);

xchange_s = NHES.Fluid.HeatExchangers.Utilities.Functions.quality(hin_s,hf_s,hg_s)
- NHES.Fluid.HeatExchangers.Utilities.Functions.quality(hex_s,hf_s,hg_s);

//Check for phase change by determining if a change in quality occurs.
if((xchange_t > 0.0 or xchange_t < 0.0 or xchange_s > 0.0 or xchange_s < 0.0)) then
  Cr = 0.0;
else
  Cr = abs(Q_max)/abs(Q_min);
end if;
//Calculate Q using Cr, Q_max, and NTU. Note that if Cr = 0, this simplifies to Q_max*(1-exp(-NTU)) jus
Q = (1-exp(-NTU*(1-Cr)))*Q_max/(1-Cr*exp(-NTU*(1-Cr)));
```

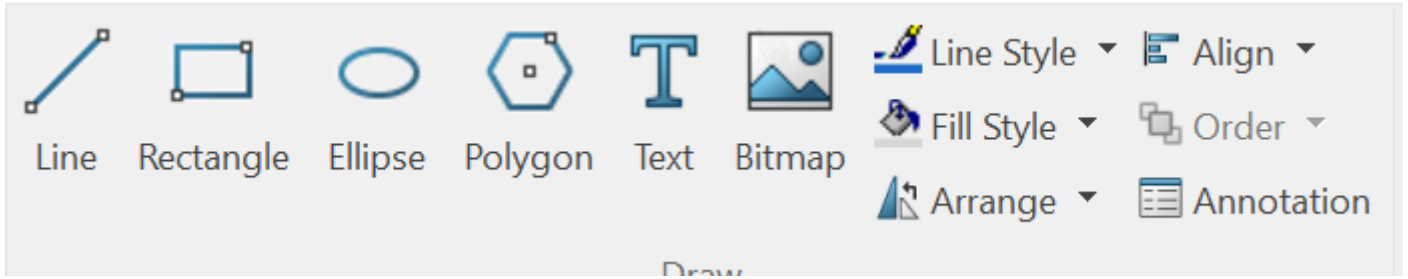
```
Q = (1-exp(-NTU*(1-Cr)))*Q_max/(1-Cr*exp(-NTU*(1-Cr)));
```

# Component Modeling – NTU HX




Dymola contains drawing methods and image import capabilities to design the instantiation graphic






IES Integrated Energy Systems

# Component Modeling – Custom Valves

```
    model ValveLinear "Valve for water/steam flows with linear pressure drop"
// medium states
state_a = Medium.setState_phX(port_a.p, inStream(port_a.h_outflow), inStream(port_a.Xi_outflow));
state_b = Medium.setState_phX(port_b.p, inStream(port_b.h_outflow), inStream(port_b.Xi_outflow));

// Pressure drop in design flow direction
dp = port_a.p - port_b.p;

// Design direction of mass flow rate
m_flow = port_a.m_flow;
assert(m_flow > -m_flow_small or allowFlowReversal, "Reversing flow occurs even though allowFlowReversal is false");

// Mass balance (no storage)
port_a.m_flow + port_b.m_flow = 0;

// Transport of substances
port_a.Xi_outflow = inStream(port_b.Xi_outflow);
port_b.Xi_outflow = inStream(port_a.Xi_outflow);

port_a.C_outflow = inStream(port_b.C_outflow);
port_b.C_outflow = inStream(port_a.C_outflow);
      port_b.h_outflow = inStream(port_a.h_outflow);
```

IES Integrated Energy Systems

# Component Modeling – Custom Valves

- We can have other kinds of valves though.
- What about a valve that internally enforces a [0,1] limit?
- Or a valve that opens at a given pressure?

```
p_in = port_a.p;
if p_in>p_spring then
   der(opening) = (1-opening)/tau;
else
   der(opening) = (open_min-opening)/tau;
   end if;
dp = port_a.m_flow*sqrt(port_a.m_flow*port_a.m_flow + 0.001*0.001)*K/((opening+0.001));
```



IES Integrated Energy Systems

# Component Modeling - Review

- Variable types and declarations
  - Reals, strings, integers, packages
- If statements,
  - Explicit, must be balanced
- GUI development
- UI development
  - Annotation, “replaceable” keyword
- Model extension
  - Use of partial models
  - Allows for many models built from few base classes



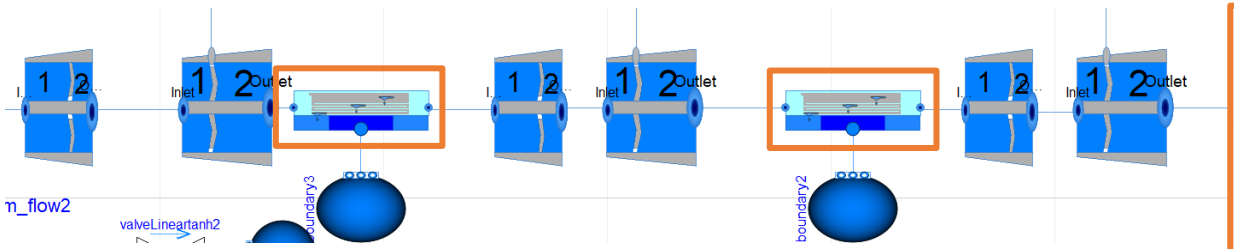
IES Integrated Energy Systems

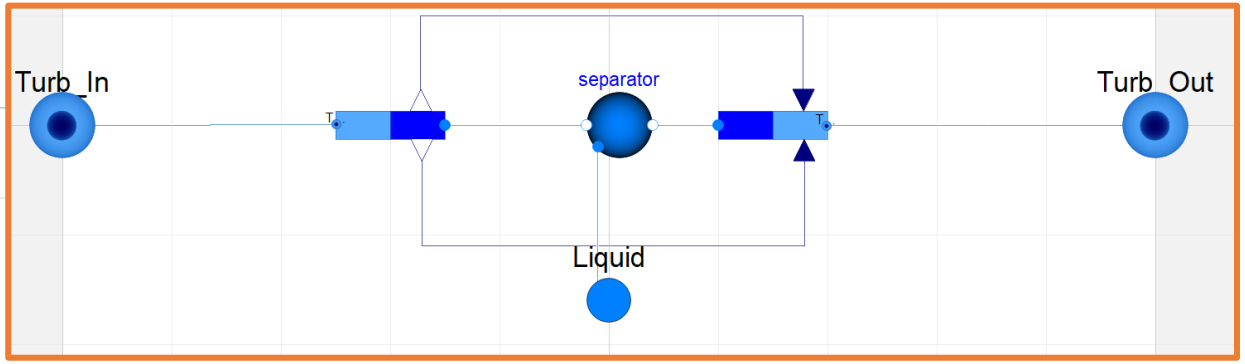
# Enhancing Existing Models

- Use of coding and established models is possible
  - Example: Cylindrical physics adaptive moisture separator
  - Can be advantageous for additional calculations not typically made in base components
  - Can allow for replacement of old equations in new models

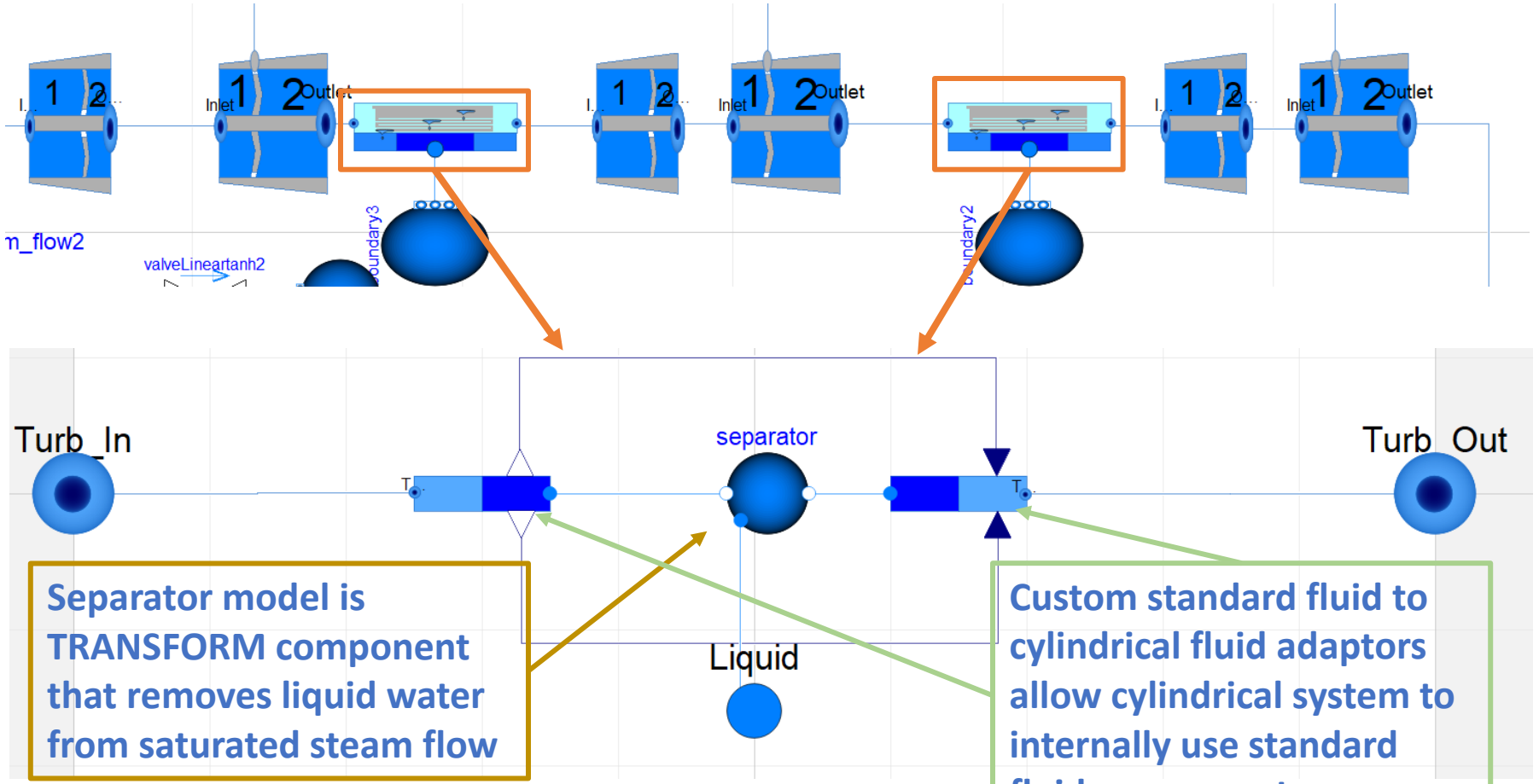
# Model Enhancement

- Hybrid contains a cylindrical fluid motion based stage by stage turbine model
- Required a new connector port that communicates entirety of velocity vector
- To avoid re-writing all components, some old models are adapted



# Model Enhancement



# Model Enhancement

- Using top-level variables can help determine some values may be “hidden” or simply difficult to find within a large system
  - Example: STHX standard model does not calculate Q_total, but that variable could easily be added
- Variables embedded within object instantiations can be referenced by Name.subset.variable

# Controls in Hybrid

- Published models are designed to follow the same control system
  - Example: HTGR model
  - Standard control mechanism built into package development
  - Sensor buses measure sensor inputs
  - Actuator buses send control signals

- Outside input can be used for control setpoints
  - 2 main methods for this:
    - Text file generation for in-model designated value reading
    - Use of the Python interface

# Controls in Hybrid

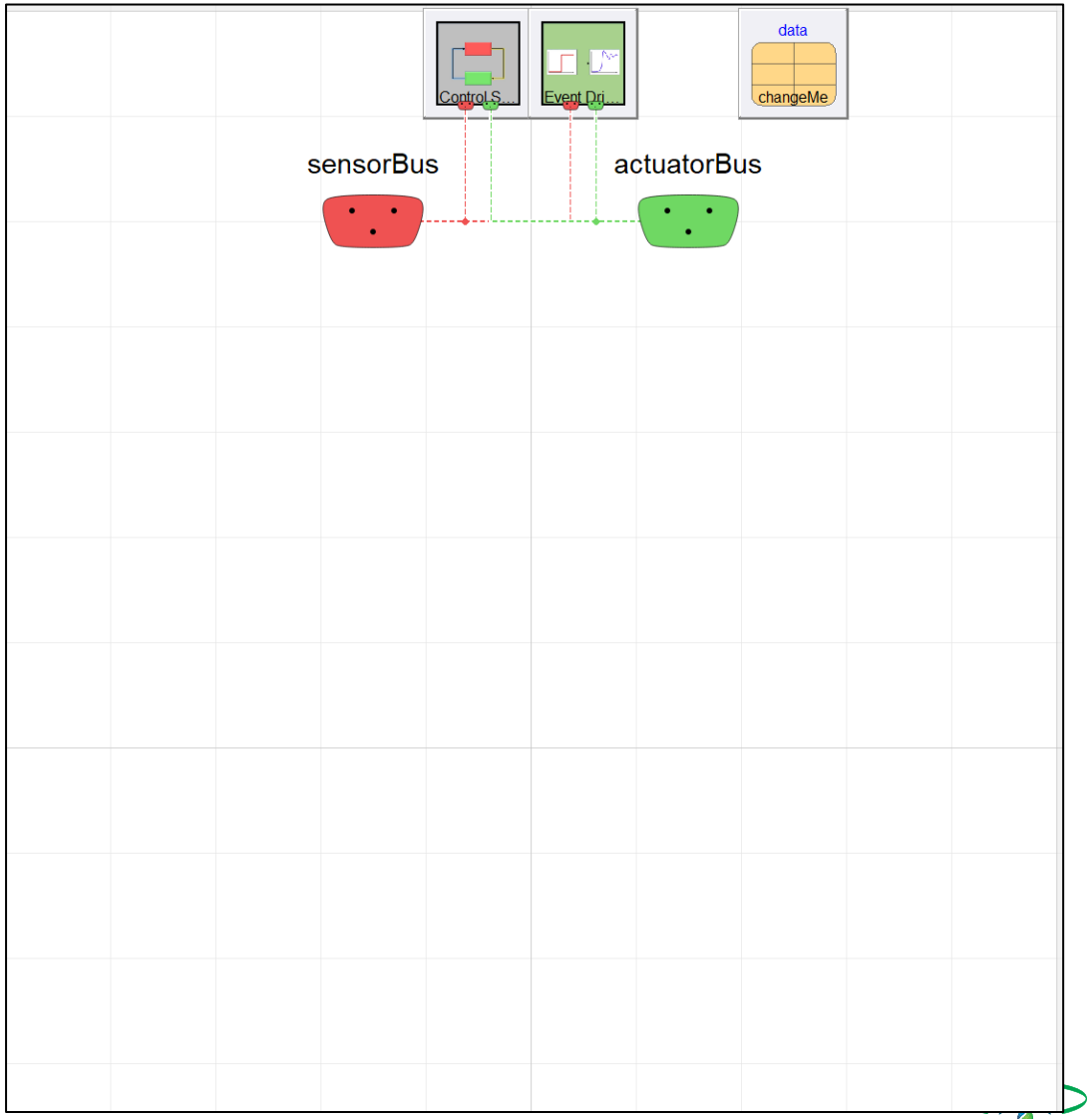
- Control methods use a variety of different components combined to obtain desired actions
  - P/PI/PID controller
  - Switches
  - Multi-signal add blocks
  - Min/max filters
  - Timers and delays
  - Outside input
- All control options for single application must ensure that the equation set is balanced
  - Use “dummy” connections to fill out equation set



IES
Integrated Energy Systems

# Controls in Hybrid

- Controls are part of standard submodule that main Hybrid models extend from
- This structure is included in every development package

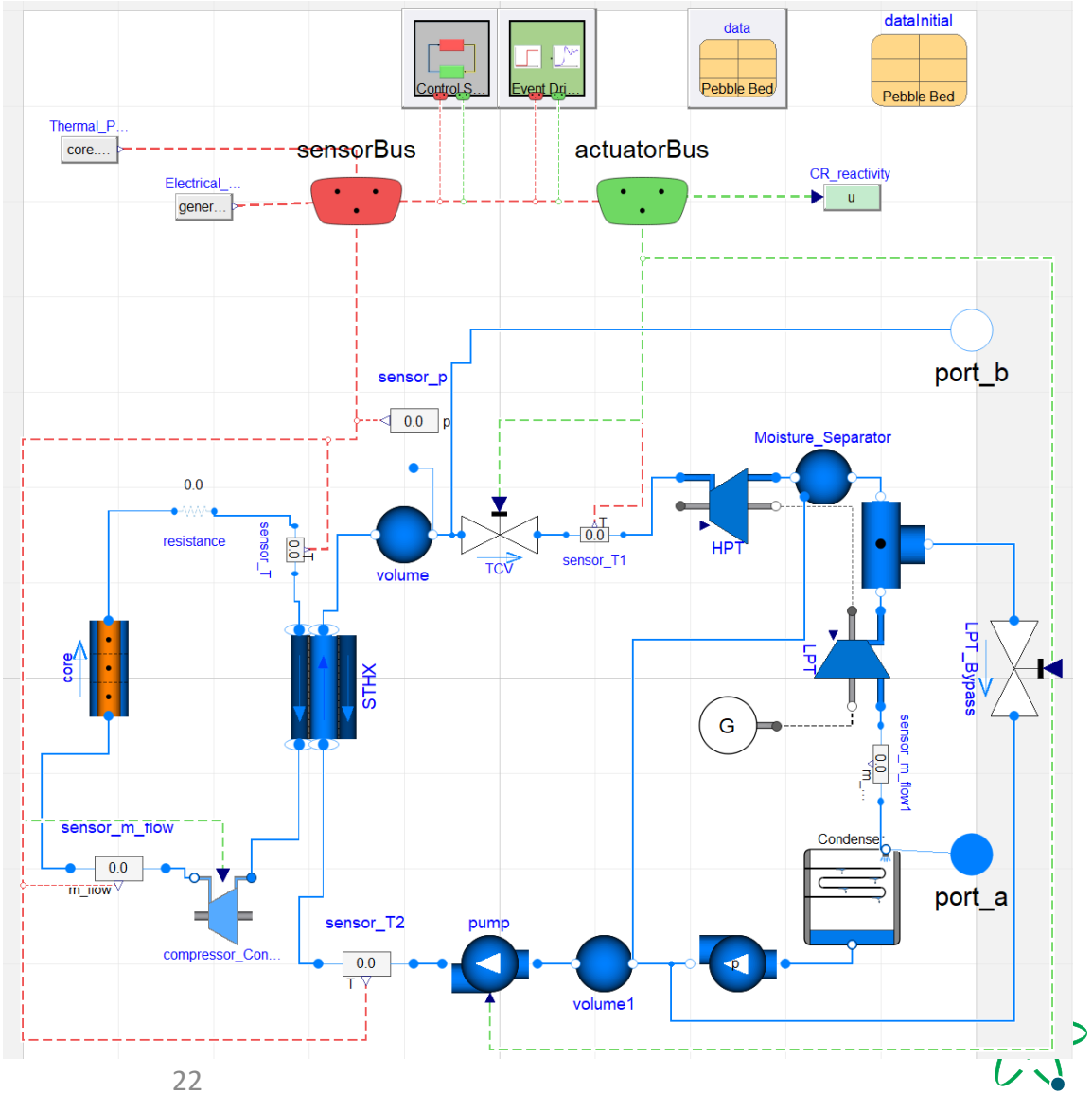





IES Integrated Energy Systems

# Controls in Hybrid

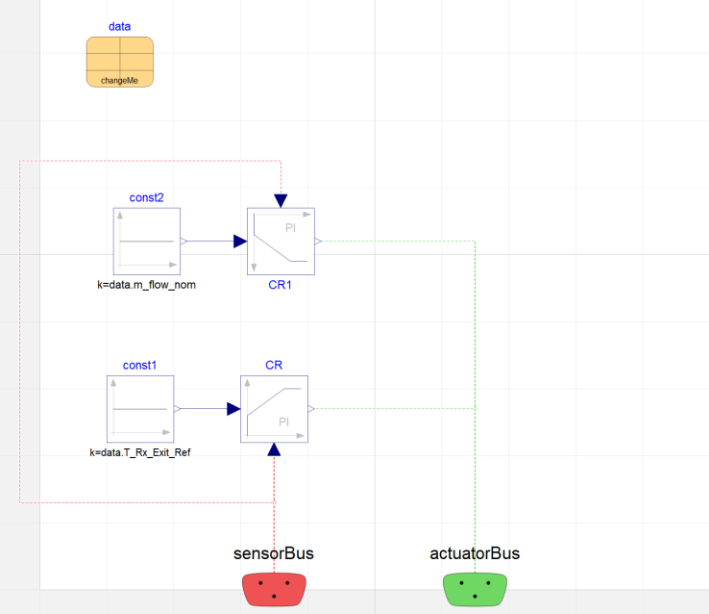
- Red dashed lines connect to sensor bus
- Green dashed lines connect to actuator bus
- Control systems can be interchanged at top-level

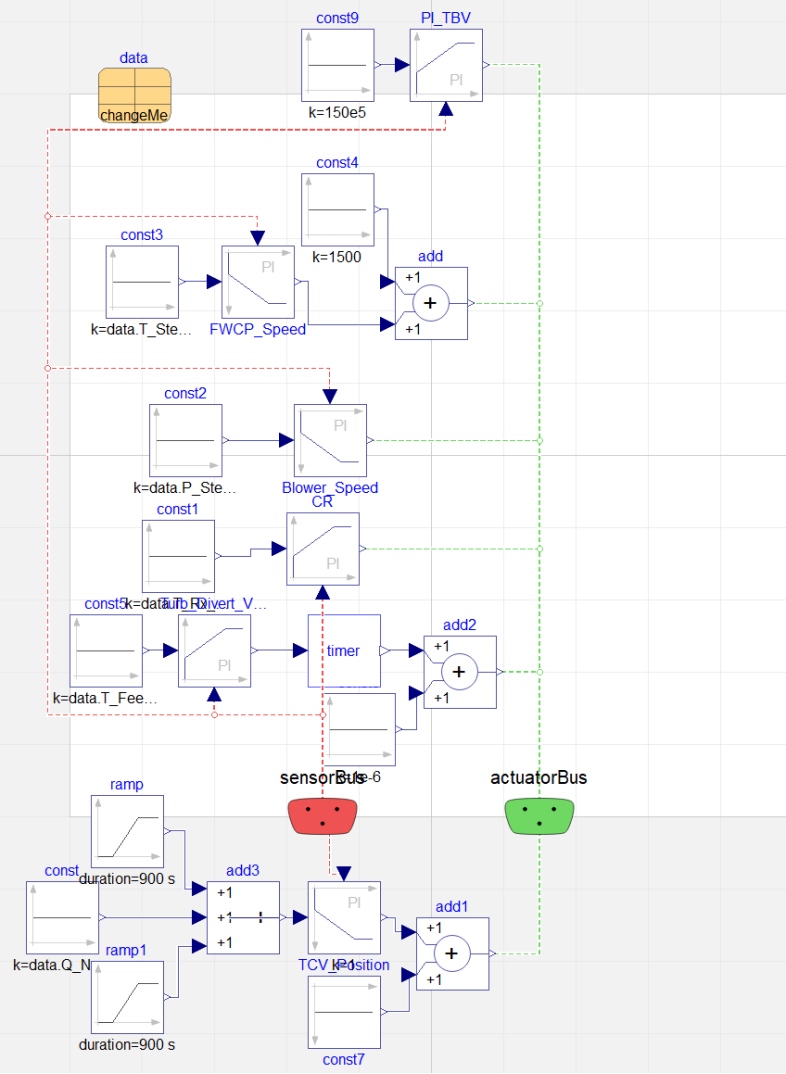


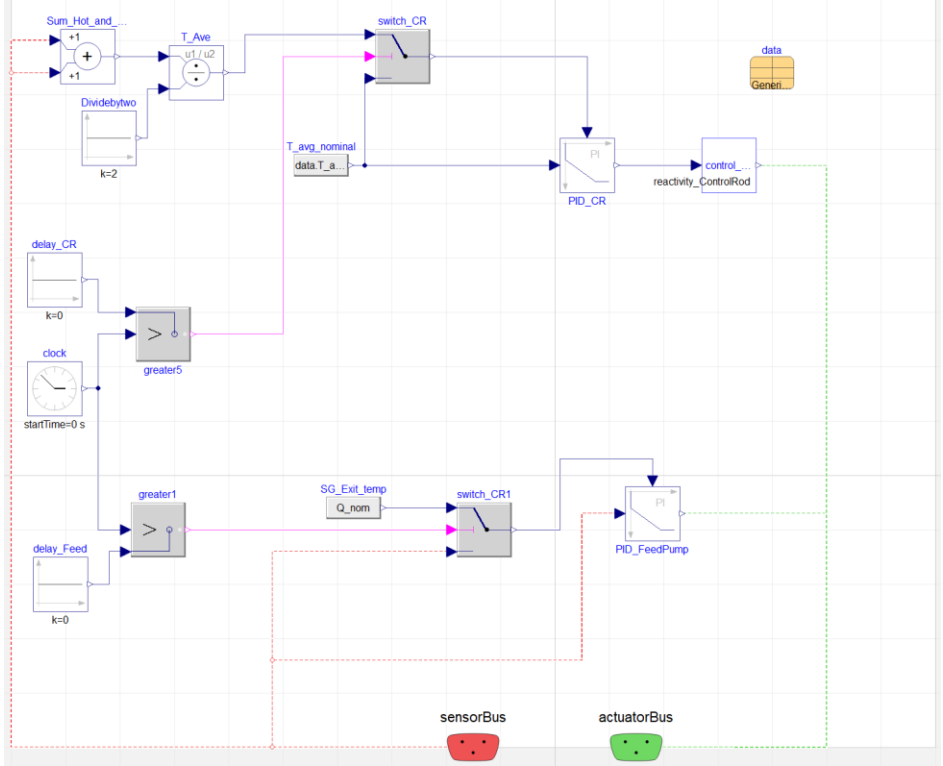



IES Integrated Energy Systems

# Controls in Hybrid






IES Integrated Energy Systems

# Controls in Hybrid – Text Input

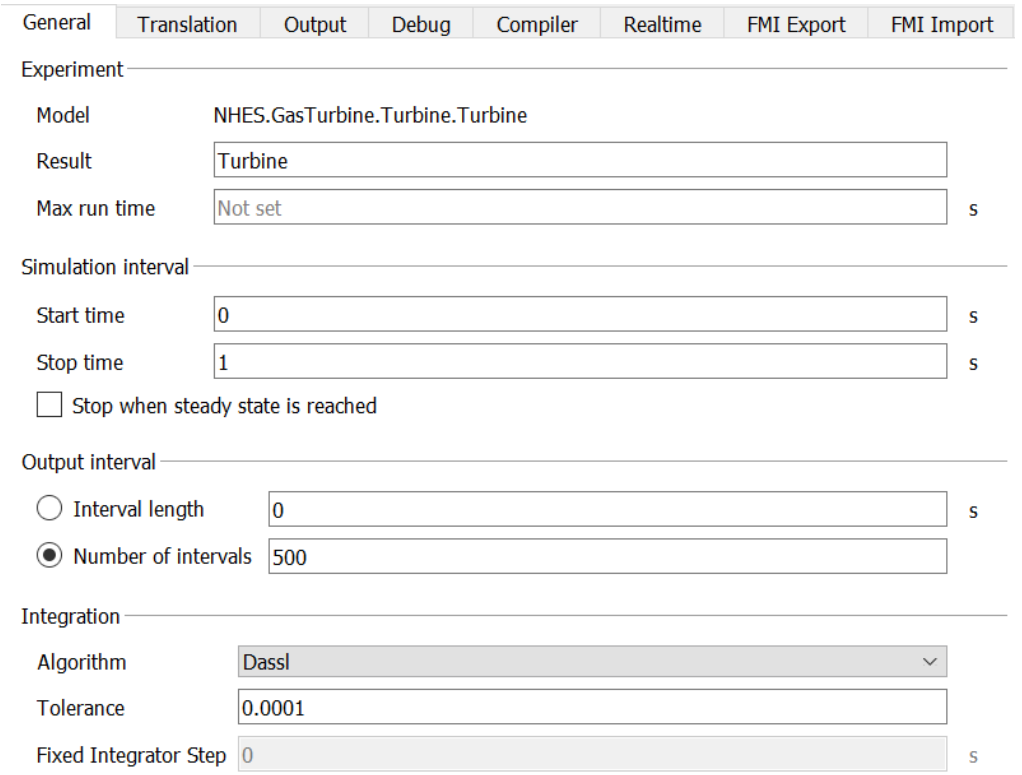
- Modelica standard library component Blocks.Sources.CombiTimeTable
- Reads values from second column of table from file
- Model parameters are text file name, table name, and time interval

| double | BOP(168,2) #Comme | double | NetDemand(168,2) |
|---|---|---|---|
| 0 | 381839590.297 | 0 | 23477782.3211 |
| 1 | 391187130.461 | 1 | -108378615.959 |
| 2 | 6042176.91138 | 2 | -168166138.839 |
| 3 | 147556178.237 | 3 | -45170911.9826 |
| 4 | 309377136.901 | 4 | 19438496.2208 |
| 5 | 303454190.736 | 5 | -46466311.757 |
| 6 | 348413012.253 | 6 | 47259297.1509 |
| 7 | 130566312.461 | 7 | 123826503.358 |
| 8 | 412170792.628 | 8 | 129749920.261 |
| 9 | 45319131.8992 | 9 | 195600686.271 |
| 10 | 228825022.847 | 10 | 241760712.801 |
| 11 | 289588649.823 | 11 | 337319096.664 |
| 12 | 102012866.462 | 12 | 372335351.97 |
| 13 | 188105637.172 | 13 | 385594609.923 |
| 14 | 358188099.205 | 14 | 404239546.583 |
| 15 | 41535328.3639 | 15 | 421325039.017 |
| 16 | 209993776.358 | 16 | 438220475.573 |
| 17 | 84669633.177 | 17 | 455055225.533 |
| 18 | 363490063.243 | 18 | 466066895.714 |
| 19 | 38664551.6855 | 19 | 473683321.603 |
| 20 | 75755792.4757 | 20 | 466941019.326 |
| 21 | 259458790.02 | 21 | 430432881.872 |
| 22 | 263891535.494 | | |

IES
Integrated Energy Systems

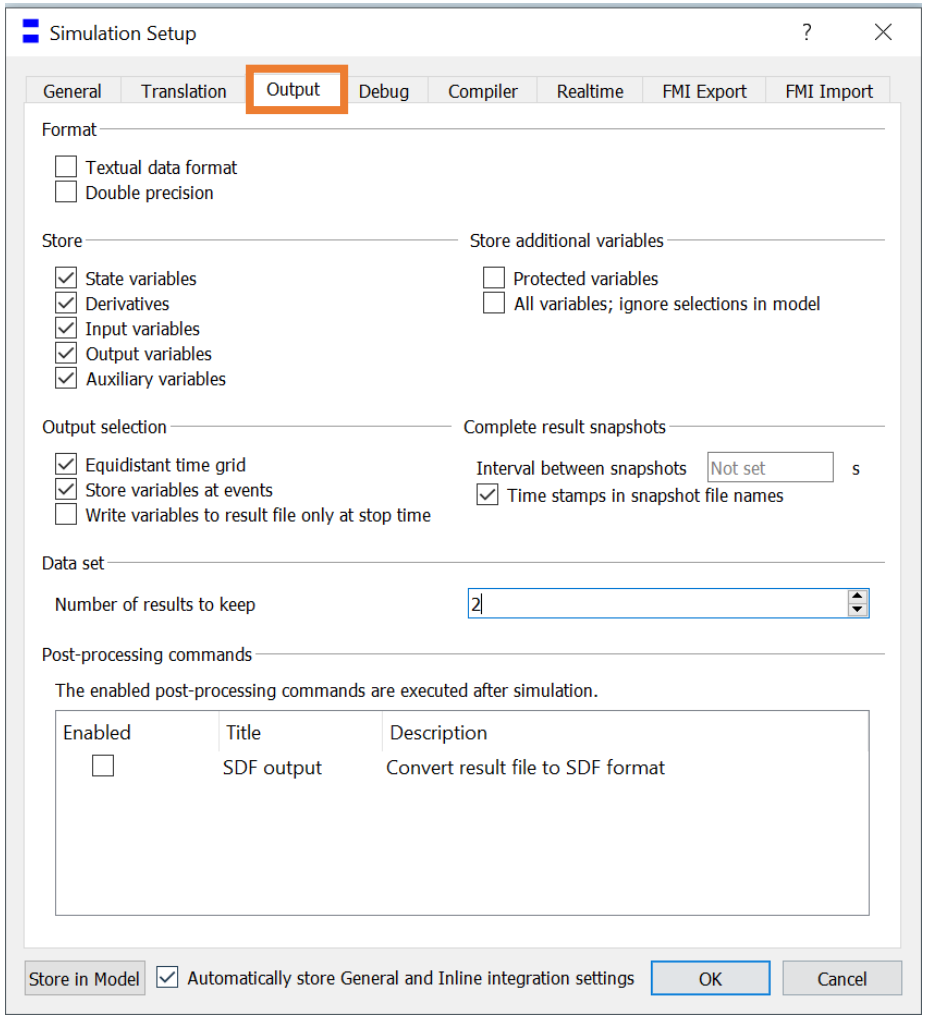
# Information Output Format

- Default output format is a binary .mat file
  - This file is readable by Matlab
  - We have Python scripts that can also read these files
- Output frequency is mostly user-defined
  - User chooses either number of output intervals or specified interval length
  - System state during events are typically also written into .mat file



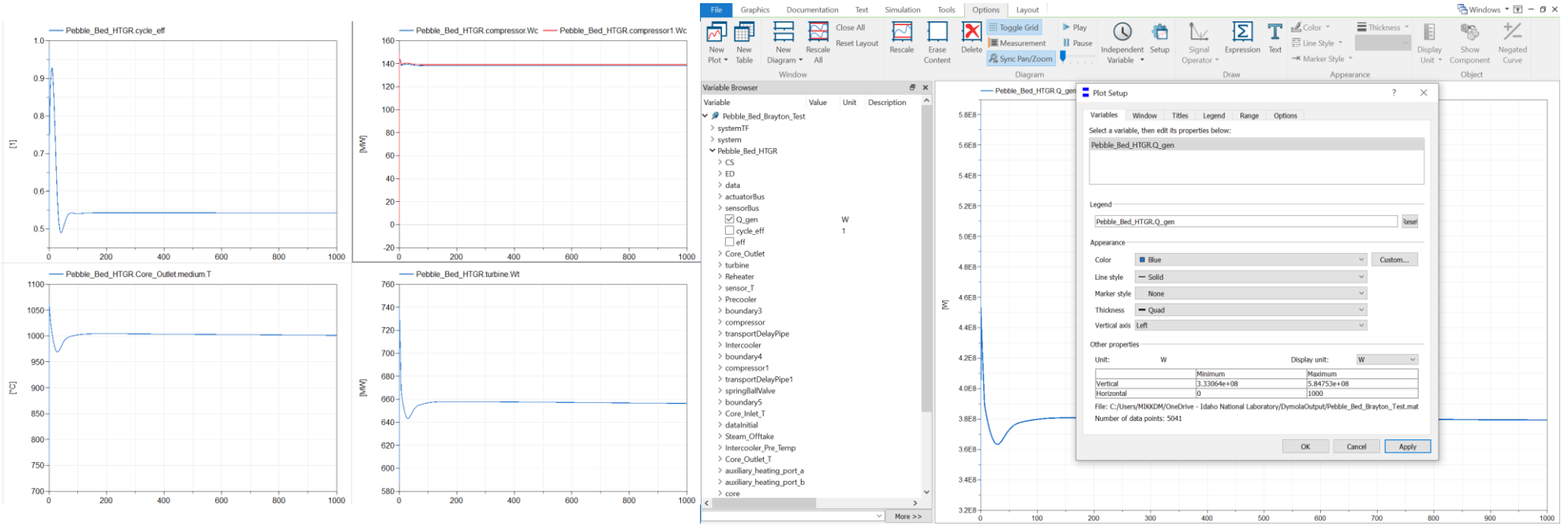
# Information Output

- Default output options are seen in the figure on the right
  - "Textual data format" option allows for more straightforward reading option
- Options exist regarding what calculations to store:
  - Add protected variables
  - Storing at events



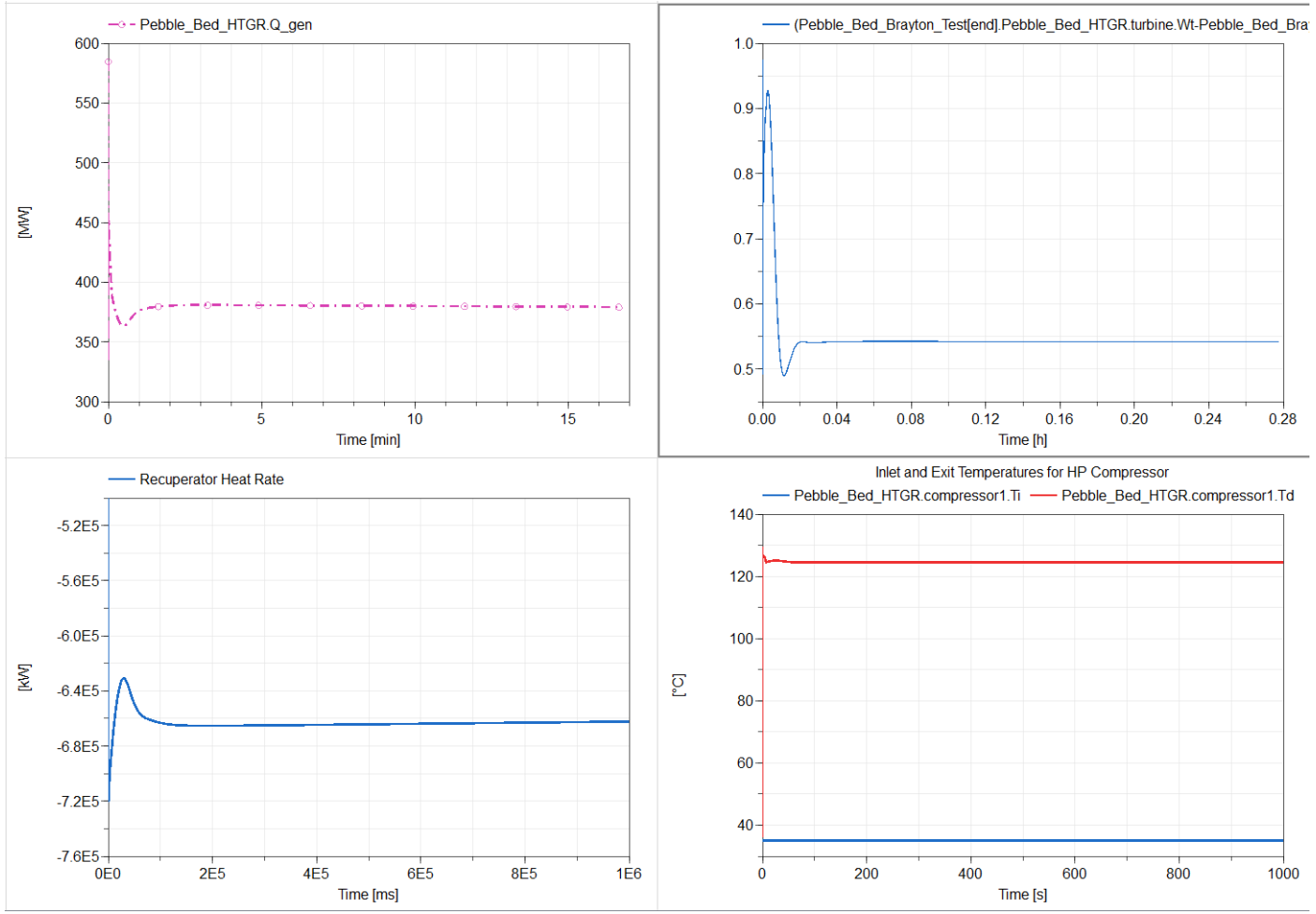
# Plotting Results

- Dymola has plotting capabilities built into the program
- Plots can be accessed during simulation or generated from loaded results
- Plot generation can be scripted

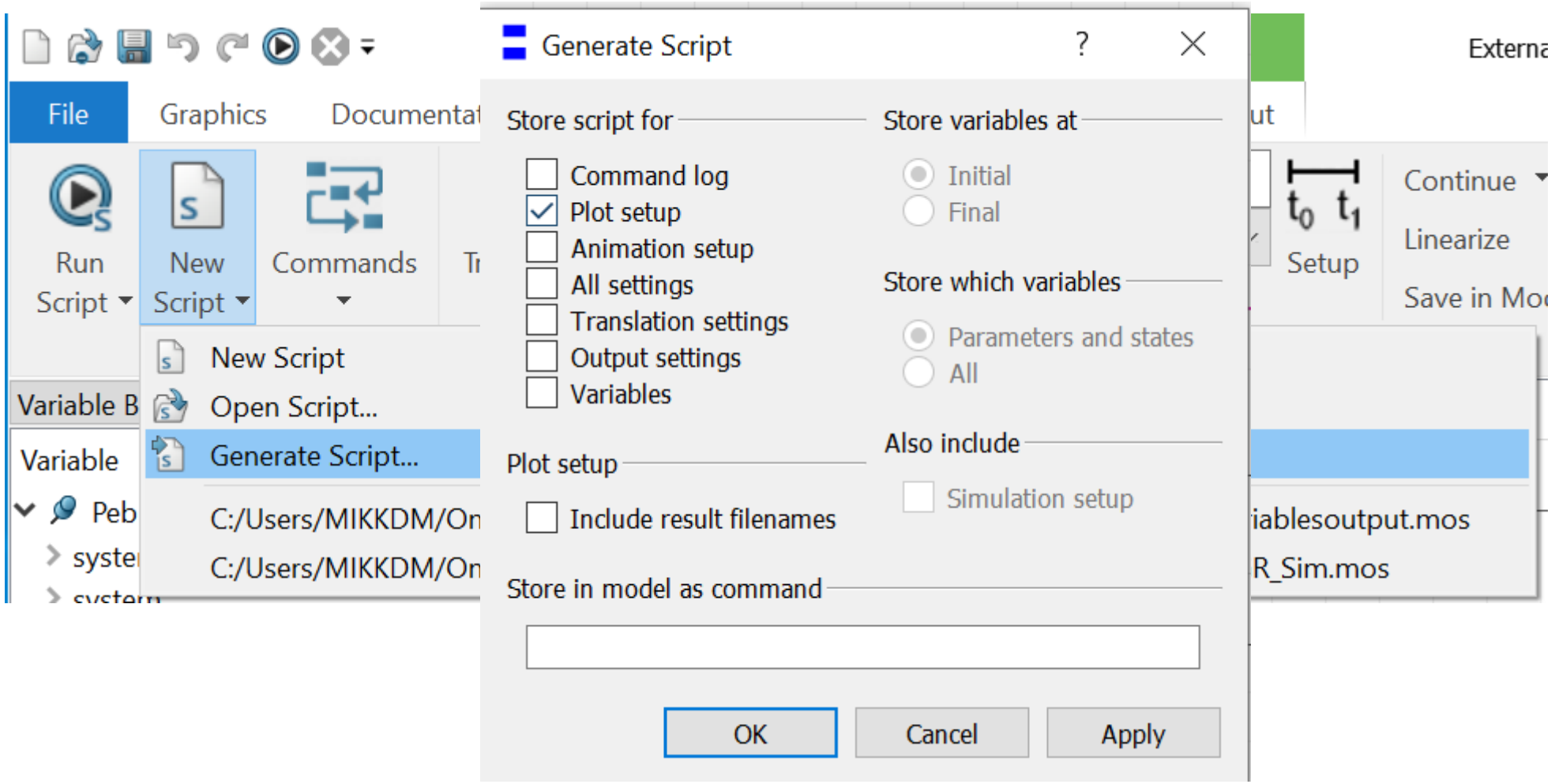


# Generating Plotting Scripts

- Generating plots to the right is done via Simulation tab within Dymola
- Custom expressions can be plotted (see top right graph)



# Generating Plotting Scripts



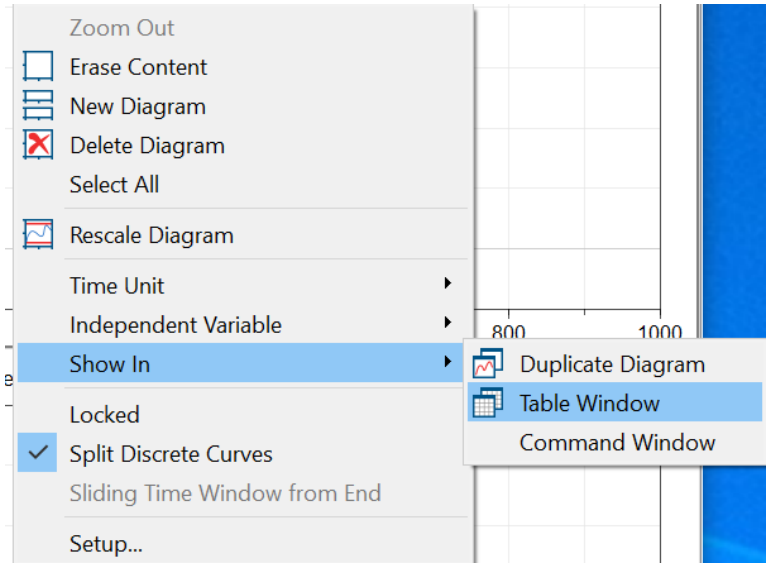
# Plotting Script Example

```
// Script generated by Dymola Tue Feb 15 17:04:50 2022
// Plot commands
removePlots(false);
createPlot(id=1, position={4, 62, 884, 547}, y={"Pebble_Bed_HTGR.Q_gen"}, ran
createPlot(id=2, position={35, 35, 884, 547}, y={"Pebble_Bed_HTGR.cycle_eff"}
createPlot(id=2, position={35, 35, 884, 547}, y={"Pebble_Bed_HTGR.compressor.
createPlot(id=2, position={35, 35, 884, 547}, y={"Pebble_Bed_HTGR.Core_Outlet
createPlot(id=2, position={35, 35, 884, 547}, y={"Pebble_Bed_HTGR.turbine.Wt"
createPlot(id=4, position={0, 0, 1479, 1047}, y={"Pebble_Bed_HTGR.Q_gen"}, ra
createPlot(id=4, position={0, 0, 1479, 1047}, y=fill("", 0), range={0.0, 0.28
plotExpression(apply((Pebble_Bed_Brayton_Test[end].Pebble_Bed_HTGR.turbine.Wt
Pebble_Bed_Brayton_Test[end].Pebble_Bed_HTGR.compressor.Wc)/Pebble_Bed_Brayto
createPlot(id=4, position={0, 0, 1479, 1047}, y={"Pebble_Bed_HTGR.Reheater.Q"
createPlot(id=4, position={0, 0, 1479, 1047}, y={"Pebble_Bed_HTGR.compressor1
createTable(id=3, position={55, 50, 914, 487}, y={"Pebble_Bed_HTGR.compressor
```

IES
Integrated Energy Systems

# Exporting Specific Variables from Dymola

- There are more plotting options in Excel
- Specific variables can be selected and placed into “Table Window”
- At that point, copy and paste will insert the results directly into Excel sheet
  - Note that events may result in 2 values listed in a single cell, for plotting purposes these commas must be removed




| | 0 | 0.0045180586 | 0.0045180595 | 0.0045180605 | 0.19992003 | 0.39984006 | 0.5997601 | 0.7996801 | 0.9996002 |
|---|---|---|---|---|---|---|---|---|---|
| Pebble_Bed_HTGR.compressor.Wc [MW] | 0.9991799 | 149.60558 | 149.60558 | 149.60558 | 142.72832 | 143.46992 | 143.47206 | 143.3964 | 143.30826 |
| Pebble_Bed_HTGR.compressor1.Wc [MW] | 1.0057143 | 150.58292 | 150.58292 | 150.58292 | 143.66074 | 144.40718 | 144.40933 | 144.33316 | 144.24445 |
| Pebble_Bed_HTGR.compressor1.Tstart_in [°C] | 850 | | | | | | | | |
| Pebble_Bed_HTGR.compressor1.pout [bar] | 19.533932 | 63.56334 | 63.56334 | 63.56334 | 90.25168 | 88.64565 | 86.86325 | 85.14937 | 83.5005 |
| Pebble_Bed_HTGR.compressor1.hout [J/kg] | 1606167.4 | 2089831.5 | 2089831.5 | 2089831.5 | 2074944.4 | 2076567.5 | 2076572.1 | 2076406.8 | 2076214 |
| Pebble_Bed_HTGR.cycle_eff [1] | 0.97458786 | 0.49266833 | 0.49266833 | 0.49266833 | 0.6730174 | 0.693019 | 0.70545113 | 0.71750605 | 0.7292967 |
| Pebble_Bed_HTGR.Reheater.Q [MW] | -262.43204 | -498.30692 | -498.30692 | -498.30692, -498.30695 | -717.0858 | -718.968 | -717.0089 | -715.27826 | -713.62305 |
| Pebble_Bed_HTGR.Reheater.Tin_t [°C] | 475.60684 | 468.89438 | 468.89438 | 468.89438 | 606.87915 | 607.5953 | 606.45 | 605.5258 | 604.66437 |
| Pebble_Bed_HTGR.Reheater.hin_t [J/kg] | 3888426.5 | 3853567.3 | 3853567.5 | 3853567.5 | 4570145.5 | 4573864.5 | 4567917 | 4563117.5 | 4558644 |

# Initialization

- Determining a physical initialization state can be the most challenging portion of a successful simulation within Dymola
- Initialization default values can be overridden in simulation
- At the end of simulations, a complete system state is written to a text file
  - Default file name is dsfinal.txt and it is overwritten every simulation
- That text file can be imported – overriding the guess values written into the text

# Variable Naming Convention

- Variables within Modelica are not allowed to have spaces in their names and names must begin with letters
  - Vector notation within Modelica does have spaces, [i, j, k, etc.]
- Variable names in result files have object levels separated by periods
  - Example: CTES.E_Stored
  - Example: NSSS.core.port_a.m_flow
  - Example: HTGR.core[3].medium.T
- Time is in seconds by default but this can be changed in the plot or table window
- Other variable units are SI standard by default

# Summary

- We develop models within Hybrid with the intent of integrating them into larger drag-and-drop systems
- This development can start from line-by-line coding or by adjusting existing models
- Hybrid contains standardization of controls and model structures to streamline integration
- Output can be interpreted internally or sent externally for analysis



IES Integrated Energy Systems

# Available Literature on Models

- Literature:
  1) https://www.osti.gov/biblio/1569288-status-report-nuscale-module-developed-modelica-framework. -- Frick, Konor L. Status Report on the NuScale Module Developed in the Modelica Framework. United States: N. p., 2019. Web. doi:10.2172/1569288.
  2) https://www.osti.gov/biblio/1333156-status-component-models-developed-modelica-framework-high-temperature-steam-electrolysis-plant-gas-turbine-power-plant -- Suk Kim, Jong, McKellar, Michael, Bragg-Sitton, Shannon M., and Boardman, Richard D. Status on the Component Models Developed in the Modelica Framework: High-Temperature Steam Electrolysis Plant & Gas Turbine Power Plant. United States: N. p., 2016. Web. doi:10.2172/1333156.
  3) https://www.osti.gov/biblio/1468648-status-report-component-models-developed-modelica-framework-reverse-osmosis-desalination-plant-thermal-energy-storage --Kim, Jong Suk, and Frick, Konor. Status Report on the Component Models Developed in the Modelica Framework: Reverse Osmosis Desalination Plant & Thermal Energy Storage. United States: N. p., 2018. Web. doi:10.2172/1468648.
  4) https://www.osti.gov/biblio/1333156-status-component-models-developed-modelica-framework-high-temperature-steam-electrolysis-plant-gas-turbine-power-plant -- Suk Kim, Jong, McKellar, Michael, Bragg-Sitton, Shannon M., and Boardman, Richard D. Status on the Component Models Developed in the Modelica Framework: High-Temperature Steam Electrolysis Plant & Gas Turbine Power Plant. United States: N. p., 2016. Web. doi:10.2172/1333156
  5) https://www.osti.gov/biblio/1557660-design-operation-sensible-heat-peaking-unit-small-modular-reactors -- Frick, Konor, Doster, Joseph Michael, and Bragg-Sitton, Shannon. Design and Operation of a Sensible Heat Peaking Unit for Small Modular Reactors. United States: N. p., 2018. Web. doi:10.1080/00295450.2018.1491181.
  6) https://www.osti.gov/biblio/1557661-thermal-energy-storage-configurations-small-modular-reactor-load-shedding -- Frick, Konor, Misenheimer, Corey T., Doster, J. Michael, Terry, Stephen D., and Bragg-Sitton, Shannon. Thermal Energy Storage Configurations for Small Modular Reactor Load Shedding. United States: N. p., 2018. Web. doi:10.1080/00295450.2017.1420945.
  7) https://www.osti.gov/biblio/1562960-dynamic-performance-analysis-high-temperature-steam-electrolysis-plant-integrated-within-nuclear-renewable-hybrid-energy-systems -- Kim, Jong Suk, Boardman, Richard D., and Bragg-Sitton, Shannon M. Dynamic performance analysis of a high-temperature steam electrolysis plant integrated within nuclear-renewable hybrid energy systems. United Kingdom: N. p., 2018. Web. doi:10.1016/j.apenergy.2018.07.060.
  8) https://www.osti.gov/biblio/1357452-modeling-control-dynamic-performance-analysis-reverse-osmosis-desalination-plant-integrated-within-hybrid-energy-systems. Kim, Jong Suk, Chen, Jun, and Garcia, Humberto E. Modeling, control, and dynamic performance analysis of a reverse osmosis desalination plant integrated within hybrid energy systems. United States: N. p., 2016. Web. doi:10.1016/j.energy.2016.05.050.