TRANSFORM Workshop

M. Scott Greenwood, PhD

Oak Ridge National Laboratory
Advanced Reactor Engineering Group

American Modelica Conference
September 21, 2020

OAK RIDGE
National Laboratory

TRANSFORM Introduction

- what is it
- where does it fit
- what can you do with it

TRANSFORM – Transient Simulation Framework of Reconfigurable Models

- **2013–2015** – Originally conceived as a high-level interface approach for modeling of advanced reactors. Including exploration of deployment and collaboration methods.
  - For example: web-apps, Excel, FMU
  - The general flexibility and capability of the language was a draw to continue growing expertise in Modelica
  - However, at that time there was no standard fluid or heat transfer libraries
  - Led to difficulty/limited usefulness in developing a high-level interface which had no components with which to model
- **2015–Present** – Scott Greenwood took over the Modelica development and re-imagined TRANSFORM to be a general library of components for modeling a variety of thermal-hydraulic and other multi-physics systems.

AdvSMR

High-level interface to allow a plug-and-play type approach for modeling of advanced reactors.

2013

NHES

Template based approach for modeling hybrid or integrated energy systems (IES).

2015

TRANSFORM

TRANSFORM released as a Modelica library with a large selection of TH components. Including the ability to model tritium and its diffusion through solids.

2017

MSR Models & SMR-GAIN

TRANSFORM expanded capability for source term accountancy for salt-fueled MSRs. TRANSFORM also used in support of stability analysis in GAIN award and nuclear thermal propulsion system modeling.

2018

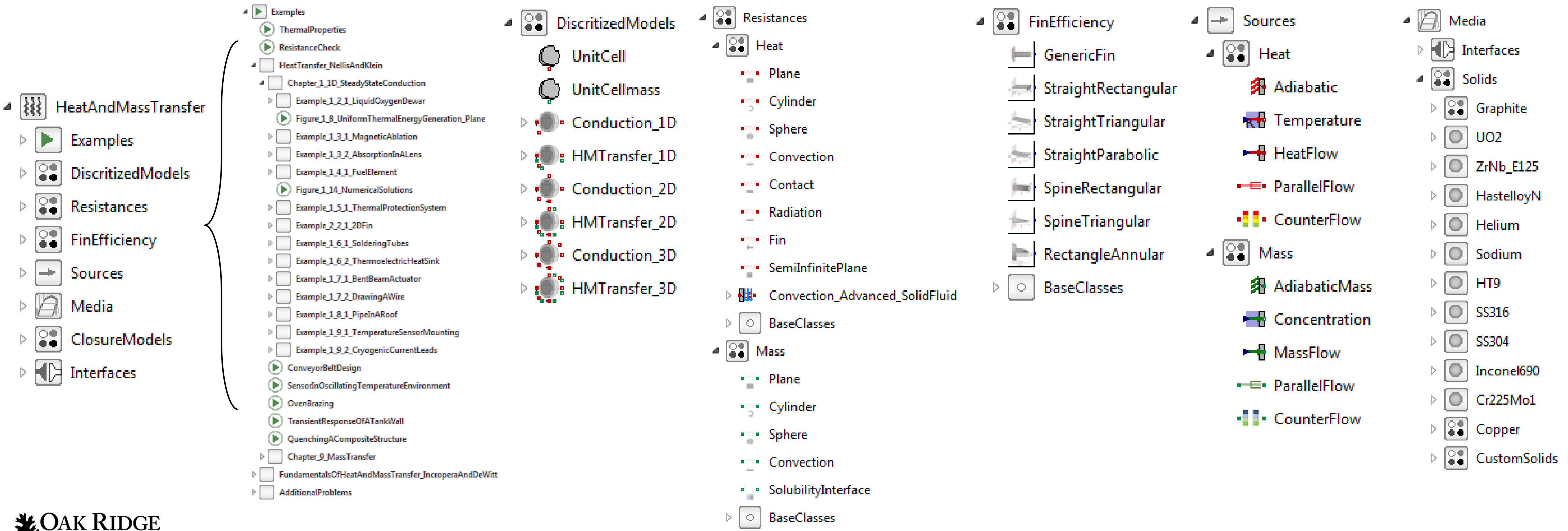
NTP & More

TRANSFORM leveraged to model system behavior of nuclear thermal propulsion systems, benchmark CIET experimental data, and design molten salt experiments for the Versatile Test Reactor.

2019

Library Package Example: Heat & Mass Transfer

- Standard implementation of diffusive heat and (trace) mass transfer



OAK RIDGE National Laboratory

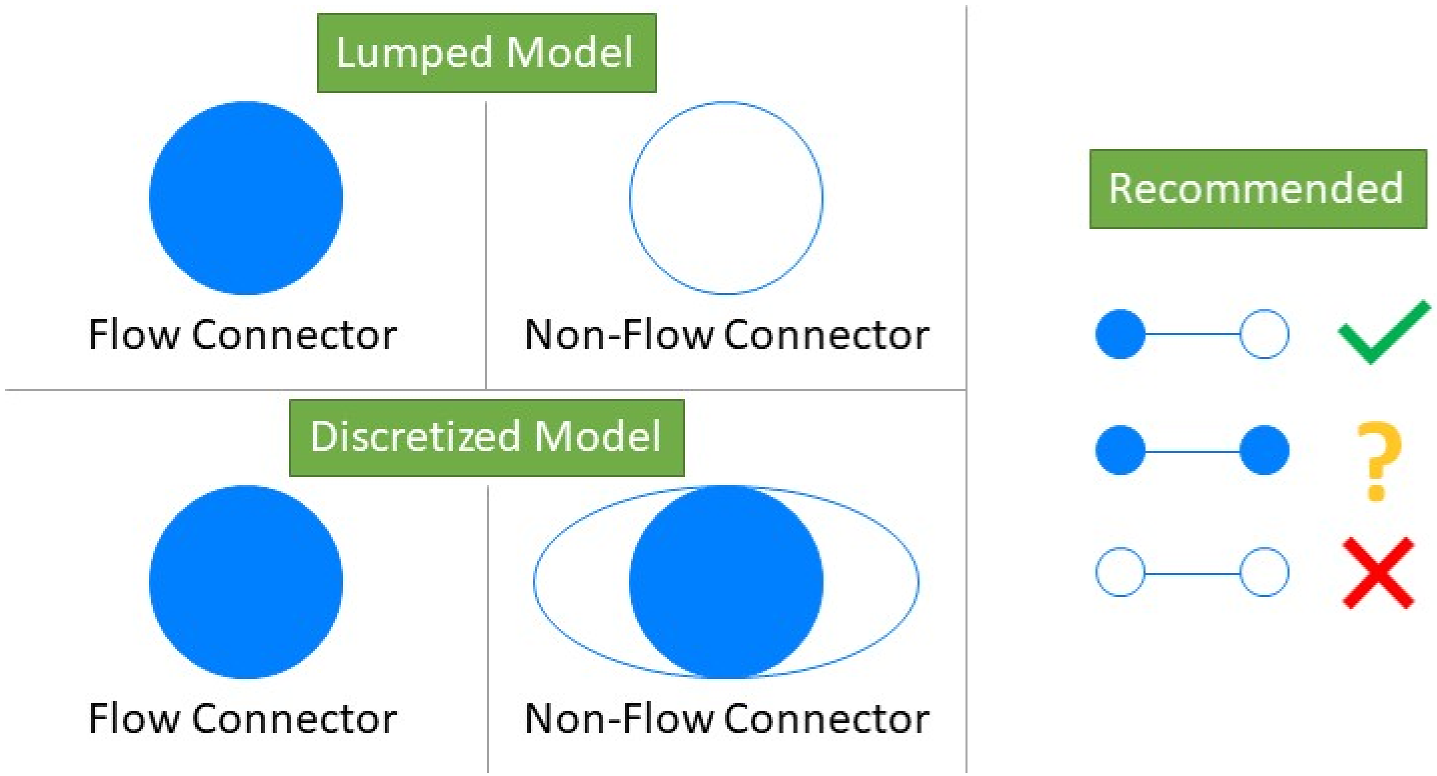
A Brief Demo of the Heat and Mass Library

Real Quick: Connectors!

- 5 min - stream, flow, non-flow, actual stream, instream

How to connect connectors?

- Pay attention to the connector visual cue!



Connectors: Fluid

- FluidPort
- FluidPort_Flow
- FluidPort_State
- FluidPorts_Flow
- FluidPorts_State

```
connector FluidPort
  "Interface for quasi one-dimensional fluid flow in a piping network
  (incompressible or compressible, one or more phases, one or more substances)"

  replaceable package Medium = Modelica.Media.Interfaces.PartialMedium
    "Medium model" a ;

  flow Medium.MassFlowRate m_flow
    "Mass flow rate from the connection point into the component";
  Medium.AbsolutePressure p "Thermodynamic pressure in the connection point";
  stream Medium.SpecificEnthalpy h_outflow
    "Specific thermodynamic enthalpy close to the connection point if m_flow < 0";
  stream Medium.MassFraction Xi_outflow[Medium.nXi]
    "Independent mixture mass fractions m_i/m close to the connection point if m_flow < 0";
  stream Medium.ExtraProperty C_outflow[Medium.nC]
    "Properties c_i/m close to the connection point if m_flow < 0";
end FluidPort;
```

- Note: The connector graphic used in TRANSFORM helps guide the user on what the port defines.

Connectors: Heat and Mass

- Heat

```
connector HeatPort
  "Interface for one-dimensional heat transfer"

  flow Modelica.SIunits.HeatFlowRate Q_flow
    "Heat flow rate. Flow from the connection point into the component is positive.";
  Modelica.SIunits.Temperature T "Temperature at the connection point";

end HeatPort;
```

- Mass (trace)

```
connector MolePort "Interface for one-dimensional mole/mass transfer"

  parameter Integer nC = 1 "Number of substances";

  flow SI.MolarFlowRate n_flow[nC]
    "Molar flow rate. Flow from the connection point into the component is positive.";
  Modelica.SIunits.Concentration C[nC] "Concentration at the connection point";

end MolePort;
```

HeatPort
HeatPort_Flow
HeatPort_State
HeatPorts_Flow
HeatPorts_State
MolePort
MolePort_Flow
MolePort_State
MolePorts_Flow
MolePorts_State

Model Creation: Connectors

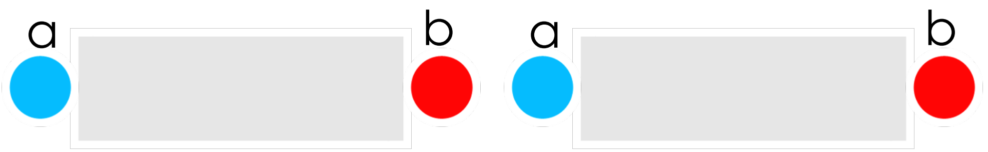
- Connectors provide a method of passing a collection or related data

- Simplest connector
  - Real input
  - Real output
- Connectors
  - Can hold any number of variables
  - Types are:
    - flow | a + b = 0
    - “non-flow” | a = b
    - stream | a = inStream(b); b = inStream(a)
- Common naming convention is “port”
  - e.g., port_a, port_b …. Not inlet/outlet

```
connector FluidPort

  flow SI.MassFlowRate m_flow;
  SI.AbsolutePressure p;
  stream SI.SpecificEnthalpy h_outflow;

end FluidPort;
```



Download TRANSFORM

Let's Go to GitHub

- Download/clone the TRANSFORM repository
  - https://github.com/ORNL-Modelica/TRANSFORM-Library

- Download/clone the TRANSFORM Training repository
  - https://github.com/ORNL-Modelica/TRANSFORM-Training
  - For future reference

Demonstration:
Simple Heat Loop

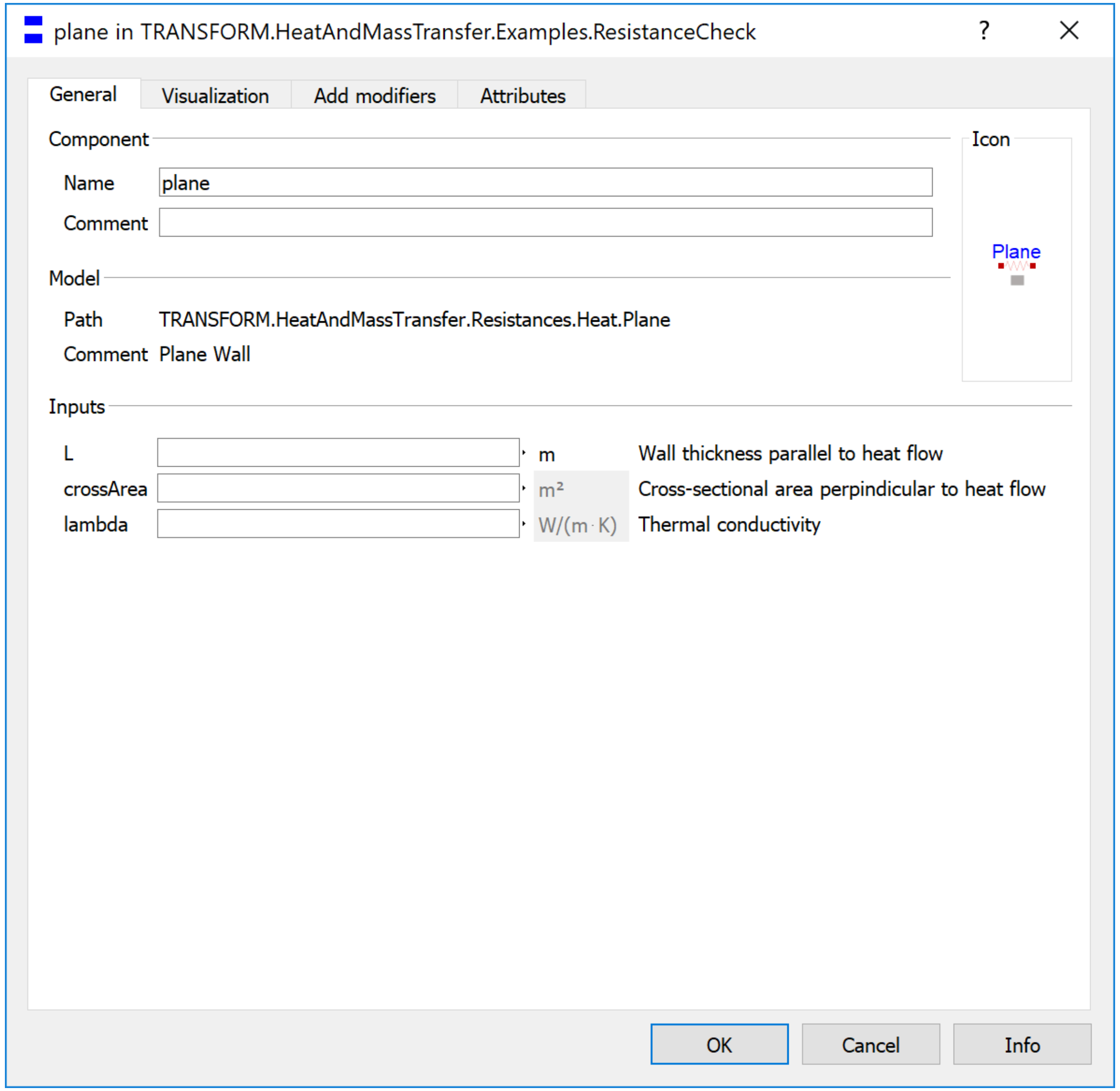
- Create a model of a force-flow reactor model using TRANSFORM (60 min)

Grab Bag:

- Some random, important information

Parameter GUI: “Inputs” group

- Time dependent inputs use generally use GUI rather than connector
  - To maintain flexibility in application
  - While reducing number of “connections” (lines all over the screen)
  - Allows a wide variety of applications to be supported



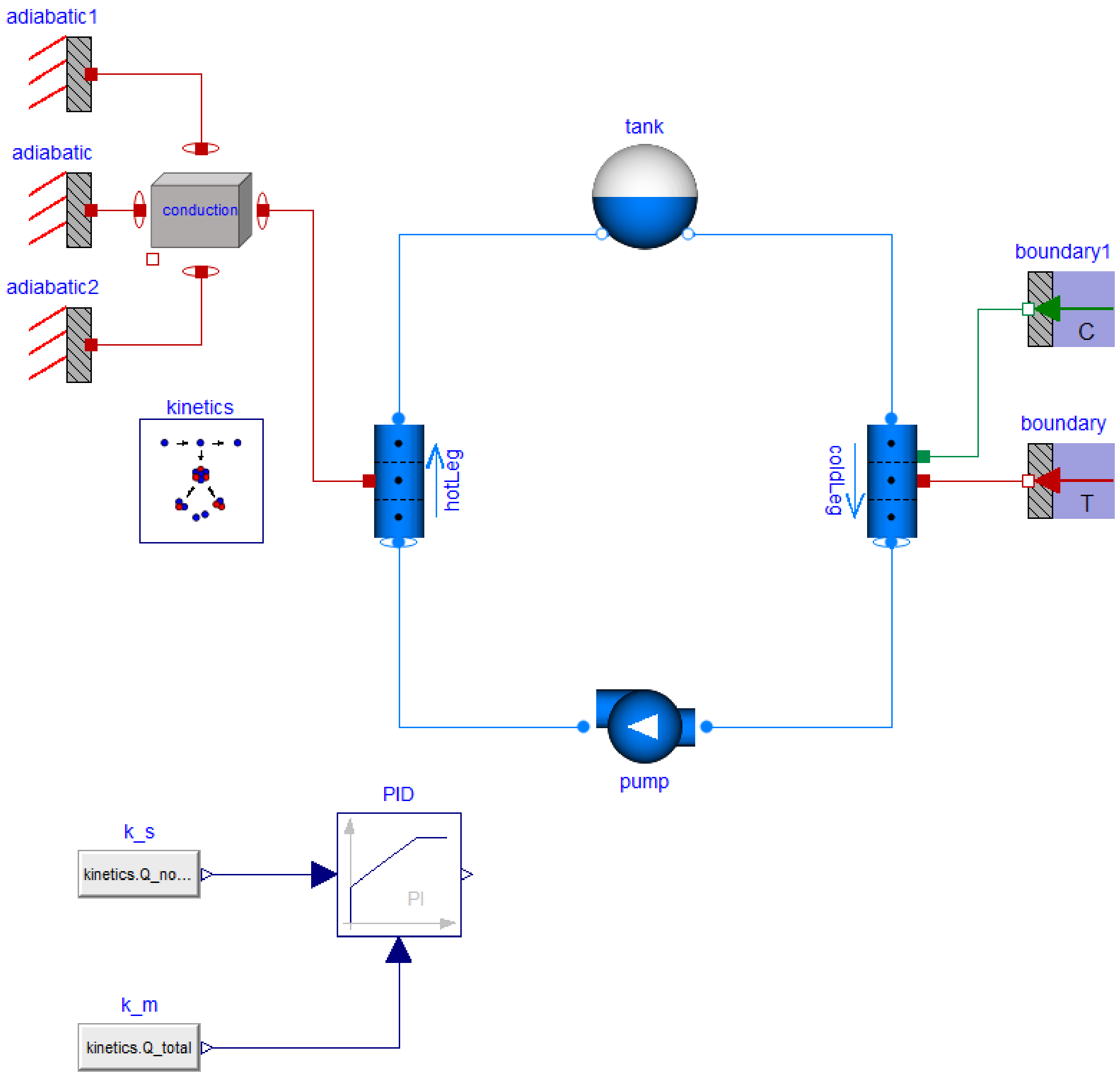
Hands-On Examples:

Create a Reactor Model

- Steps to create a simple model and gradually make it more complex

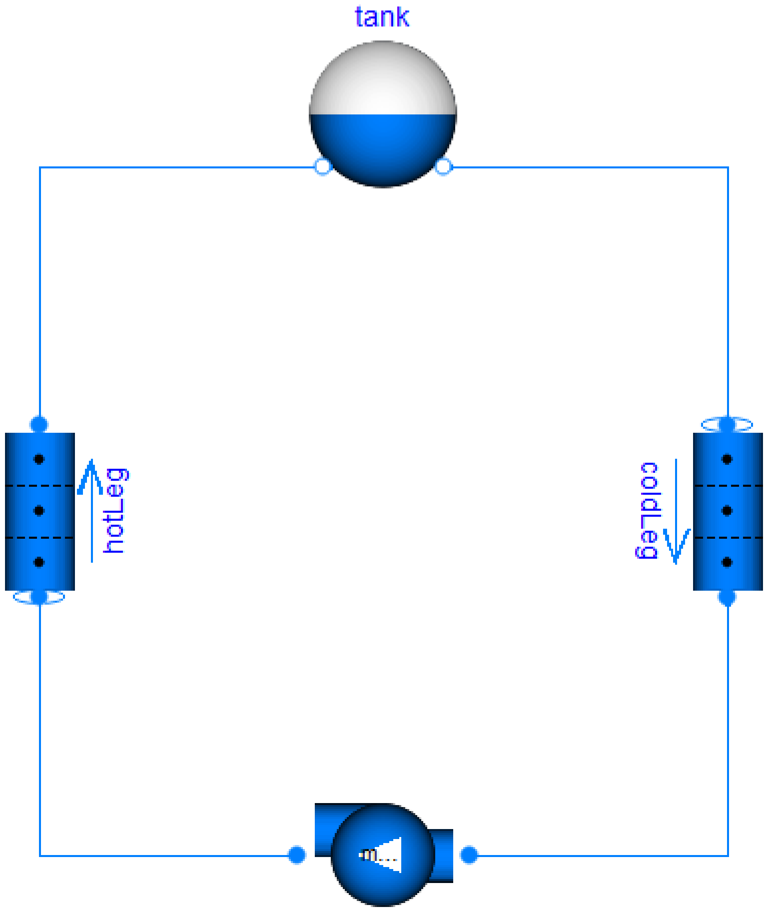
Example: Loopty Loop

- Goal:
  - Using TRANSFORM components, create a forced flow reactor model with:
    - Nuclear heating
    - Level indicators
    - Pump curves
    - PID control
    - Heat transfer



Step 1: Add base components

1. Add a new model to your packager
2. Add 2 pipes to the model
   - TRANSFORM.Fluid.Pipes.GenericPipe_MultiTransferSurface
3. Add a tank. Why do we need this?
   - TRANSFORM.Fluid.Volumes.ExpansionTank
4. Add a pump
   - TRANSFORM.Fluid.Machines.Pump_SimpleMassFlow
5. Connect components like in the picture



Tips:
- While selecting a component, click "H", "V", or "ctrl+R" to flip horizontally, vertically, or rotate.
- In the editor, click "ctrl+A" to select all and then "ctrl+shift+L" to autoformat code. May have to do twice.

Oak Ridge National Laboratory

Step 2: Specify parameters

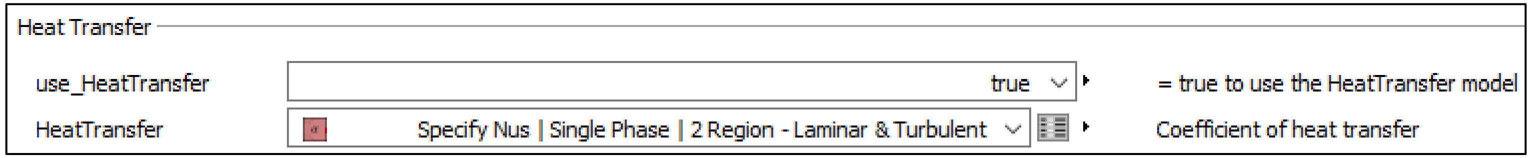
1. Select both pipes.
   - Specify diameter = 2” with nV = 4
   - Under “Initialization” tab set p_start = 1 bar, T_a_start = 50°C, and m_flow_a_start = 1 kg/s
2. For “coldLeg” pipe only, under “Advanced” tab specify
   - exposeState_a = false
   - exposeState_b = true
3. Select the tank.
   - Set tank area based on a 1’ pipe ($A=\pi r^2$)
   - Set level_start = 1 m, p_start = 1 bar, and h_start based on p = 1 bar and T = 50°C
     - h_start = tank.Medium.specificEnthalpy_pT(tank.p_start, 50 + 273.15)
4. Select the pump
   - Set m_flow_nominal = 1 kg/s
5. Select all components.
   - Set Medium = Modelica.Media.Water.StandardWater

- Statistics
  - Original Model
    - Number of components: 189
    - Variables: 1641
    - Constants: 62 (62 scalars)
    - Parameters: 298 (357 scalars)
    - Unknowns: 1281 (1761 scalars)
    - Differentiated variables: 44 scalars
    - Equations: 823
    - Nontrivial: 628
  - Translated Model
    - Constants: 881 scalars
    - Free parameters: 58 scalars
    - Parameter depending: 102 scalars
    - Continuous time states: 26 scalars
    - Time-varying variables: 334 scalars
    - Alias variables: 805 scalars
    - Assumed default initial conditions: 18
    - Number of mixed real/discrete systems of equations: 0
    - Sizes of linear systems of equations: {2, 2, 2, 2, 2, 2, 2, 2}
    - Sizes after manipulation of the linear systems: {0, 0, 0, 0, 0, 0, 0, 0}
    - Sizes of nonlinear systems of equations: { }
    - Sizes after manipulation of the nonlinear systems: { }
    - Number of numerical Jacobians: 0

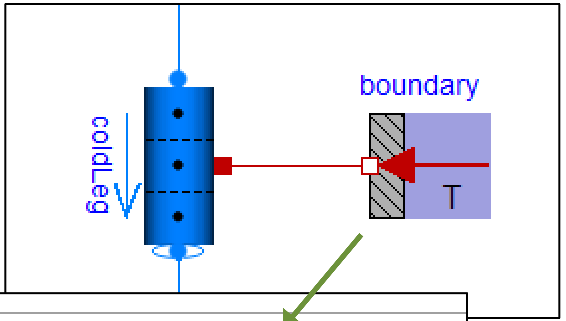
```
Integration started at T = 0 using integration method DASSL
(DAE multi-step solver (dassl/dasslrt of Petzold modified by Das
Integration terminated successfully at T = 100
   CPU-time for integration                    : 0.168 seconds
   CPU-time for one grid interval              : 0.336 milliseconds
   CPU-time for initialization                 : 0.002 seconds
   Number of result points                     : 502
   Number of grid points                       : 501
   Number of accepted steps                    : 76
   Number of f-evaluations (dynamics)          : 158
   Number of crossing function evaluations     : 576
   Number of Jacobian-evaluations              : 54
   Number of model time events                 : 0
   Number of input time events                 : 0
   Number of state events                      : 0
   Number of step events                       : 0
   Minimum integration stepsize                : 1.69e-010
   Maximum integration stepsize                : 37.6
   Maximum integration order                   : 1
```

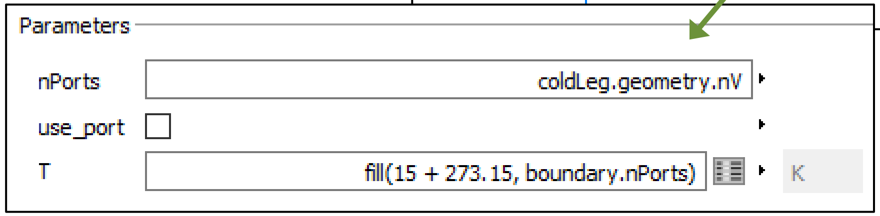
Tips:
- While in code editor or in a parameter GUI, click “ctrl+SPACE” to get autocomplete

Step 3: Add heat transfer



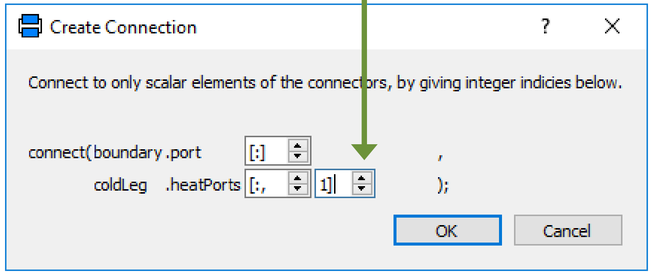
1. Select the coldLeg pipe
   - Turn on heat transfer and select a heat transfer correlation
   - Add multi-node temperature boundary
     - TRANSFORM.HeatAndMassTransfer.BoundaryConditions.Heat.Temperature_multi

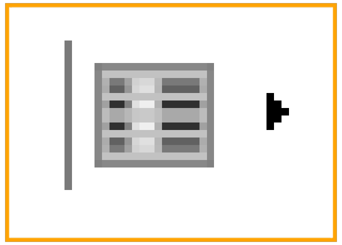




2. Select the hotLeg pipe
   - Add 10 kW of internal heating to each volume
3. Simulate for 10,000 seconds
4. Run plot_FlowLoop_Step_3.mos
5. Change boundary.T back to default.
   - Can you explain the behavior observed?

Specify the first heat transfer surface



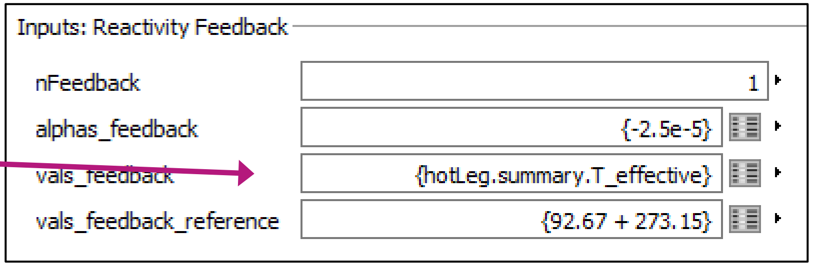


Tips:
- Check model early and often!!!

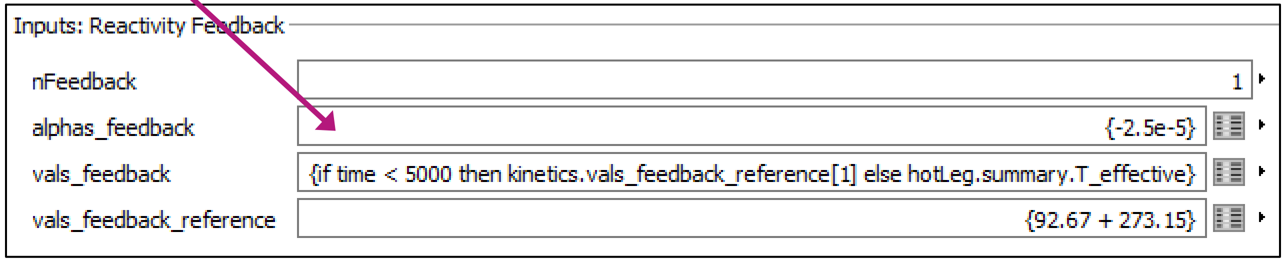
OAK RIDGE National Laboratory

Step 4: Add nuclear kinetics

1. Add a nuclear kinetics component to the model
   - TRANSFORM.Nuclear.ReactorKinetics.PointKinetics_L1_powerBased
   - Set Q_nominal = 40 kW
   - Set a temperature feedback ("kinetics" tab) based on hotLeg effective temperature



2. Change the internal heat generation in the hotLeg
   1. Set Q_gen = kinetics.Q_total/4
3. Simulate. What just happened!?
4. Try easing the reactor with some simple logic
   - Modify vals_feedback to have a time delay
5. Simulate. Is the issue fixed?



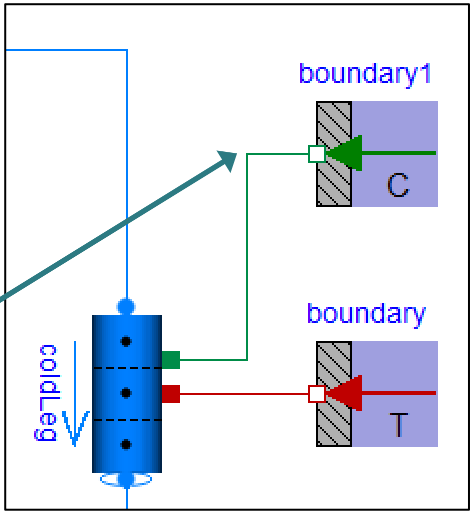
Tips:
- Autocomplete in the GUI only works if no other text exists beyond the cursor

Step 5: Add and remove a trace component

```
model FlowLoop_Step_5 "Add nuclear heating"

  package Medium = Modelica.Media.Water.StandardWater(extraPropertiesNames={"Tritium"});
```

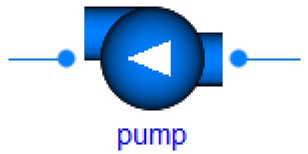
1. Need to modify the media to indicate an extra substance
   - Add to the code side the following line
     - package Medium = Modelica.Media.Water.StandardWater(extraPropertiesNames={"Tritium"});
   - Select all fluid components and change media to "Medium"
2. In the hotLeg InternalTraceGen model
   - Set mC_gen = {1e-4*kinetics.Q_total} -/s
   - Trace components are unitless. "-" can mean whatever the user desires (e.g., atoms)
3. In the coldLeg, turn on mass transfer
   - Set mass transfer to specify alphaM
   - Set D_ab0 = 1 and alphaM0 = {1000}
4. Add a multi-node concentration boundary
   - TRANSFORM.HeatAndMassTransfer.BoundaryConditions.Mass.Concentration_multi
   - Default values are fine
5. Simulate. Can you see your substance in the system?
   - Search in variable browser for mC



Tips:
- The Dymola simulation search permits the use of wild cards "*". Use it!

Step 6: Change pump component

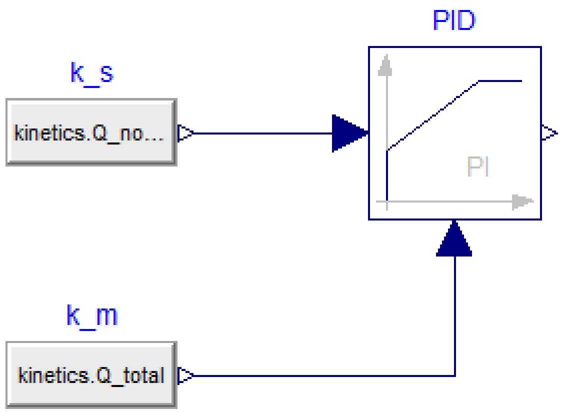
1. Right click the pump and select "Change Class"
2. Select "Pump_Controlled"
3. In the pump component
   - Set p_a_start = 1 bar, p_b_start = 1.1 bar
   - Set T_a_start = 50°C and m_flow_start = 1 kg/s
4. Simulate. What changed and why?
5. In the pump change "controlType" from "RPM" to "m_flow"
6. Simulate. What changed and why?

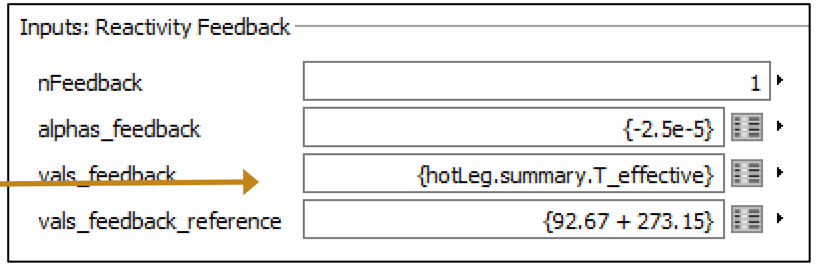


Tips:
- Using GUI to change component names will update all references to it.
- Update variable using the Edit>Variables for all uses to be updated as well.

Step 7: Use controller to alter power behavior

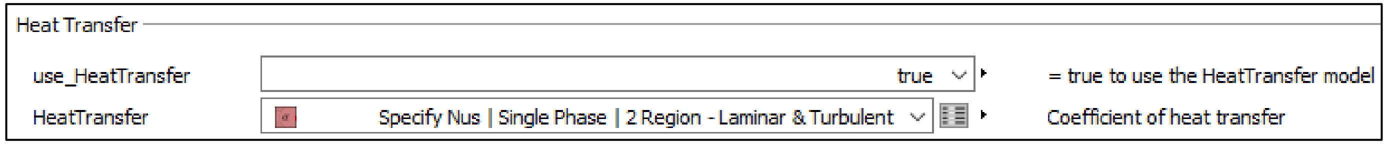
1. Drag a PID component in the model
   - TRANSFORM.Controls.LimPID
   - Set controllerType = PI
   - Set k_s and k_m = 1/kinetics.Q_nominal
   - Set k = 1e-5
2. Add 2 real expression inputs into the model
   - Modelica.Blocks.Sources.RealExpression
   - Connect them to the PID
   - Set the one connected to k_s equal to kinetics.Q_nominal
   - Set the one connected to k_m equal to kinetics.Q_total
3. Set kinetics.rho = PID.y
4. Simulate. What happened?
5. Remove the "if then" logic from vals_feedback
6. Simulate. Much better now, yes?



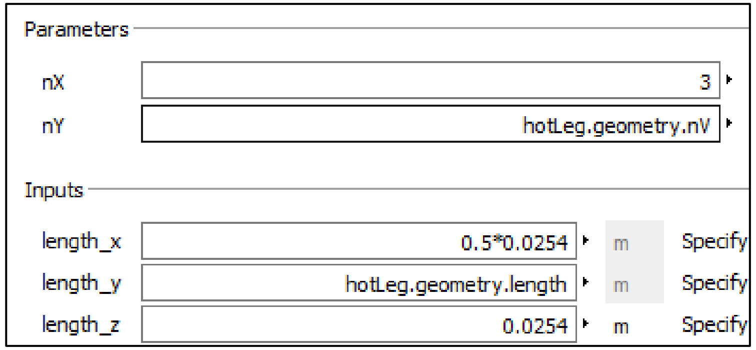


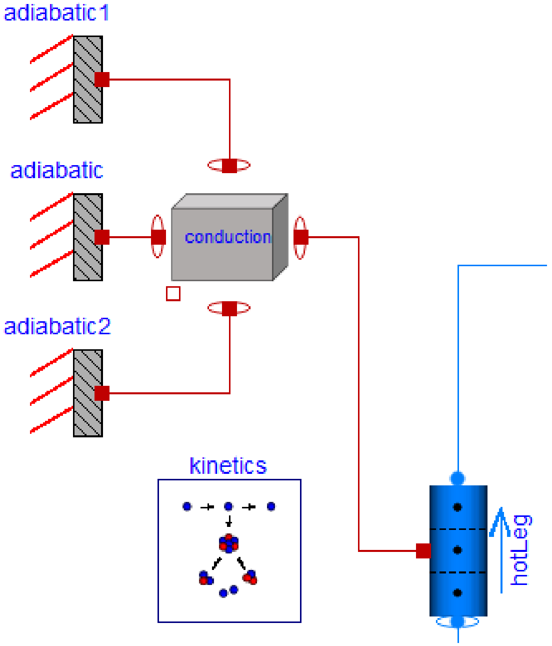
Tips:
- Hover over connections to get connector information

Step 8: Include a solid heating element (e.g., fuel)



1. Turn on heat transfer in the hotLeg
2. Turn off internal heat generation
3. Drag a 2D Conduction component into the model
   - TRANSFORM.HeatAndMassTransfer.DiscritizedModels.Conduction_2D
   - Under "Advanced" tab set all exposeStates* = true
   - Under "Initialization" tab and set temperatures 50°C
   - Set geometry as shown
   - Under "InternalHeatModel"
     - Set Q_gen = kinetics.Q_total/(conduction.geometry.nX*conduction.geometry.nY)
   - Set Material to SS316
4. Drag 3 adiabatic multi-node boundary conditions into the model
   - TRANSFORM.HeatAndMassTransfer.BoundaryConditions.Heat.Adiabatic_multi
   - Set "adiabatic" nPorts = conduction.geometry.nY and the other two to nX
5. Connect models as shown
6. Simulate

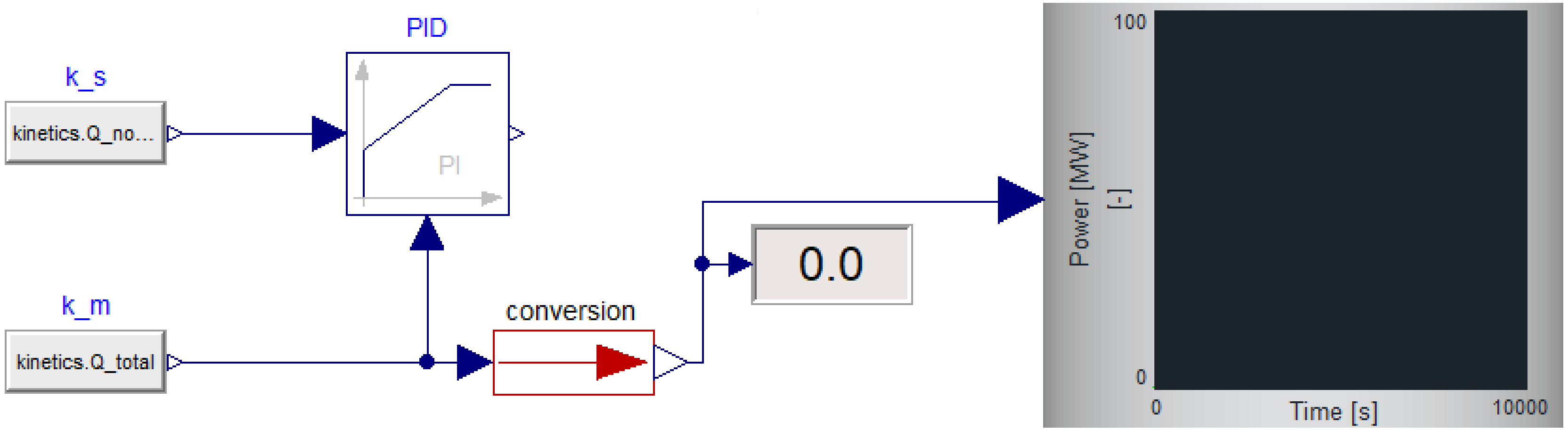




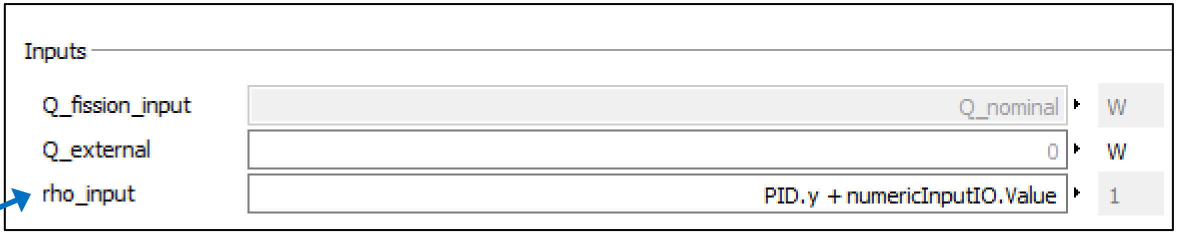
Tips:
- Look into sigmoid or easing functions to smooth transitions

Step 9: Add Visual Components

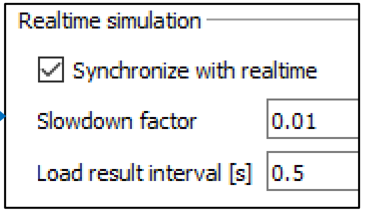
1. Add the following to the model and connect
   - TRANSFORM.Utilities.Visualizers.DynamicGraph
   - TRANSFORM.Utilities.Visualizers.displayReal
   - TRANSFORM.Units.Conversions.Models.Conversion
2. For conversion, set to_kilo
3. In plot, set y_max = 100 and t_end = 10,000
4. Under the coldLeg and hotLeg components
   - Under "Visualization" tab turn on show_colors
   - Set val_min = 40°C and val_max = 100°C
     - Careful with the units!

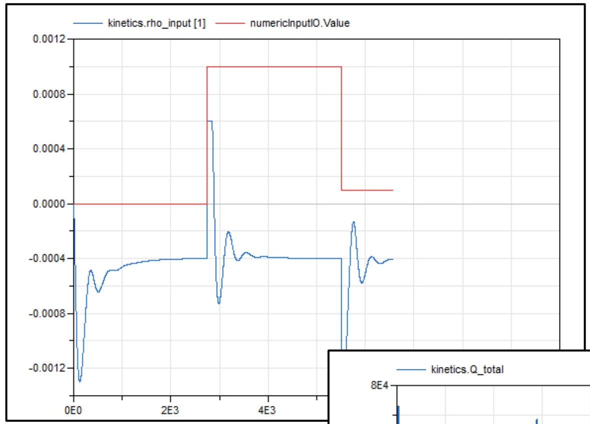


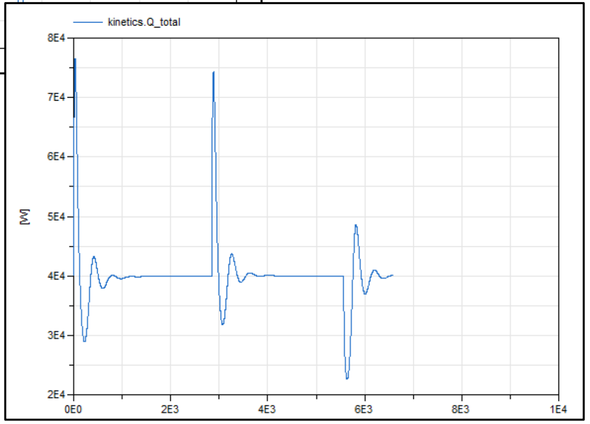
Step 10: Add Realtime User Input



1. Add a user input to the model
   – UserInteraction.Inputs.NumericInputIO
2. Add the input value to the kinetic model
3. Switch to the simulation view
4. Open the model viewer 
5. Under simulation settings
   – “Realtime” tab – set settings
   – “Compiler” tab – select DDE server

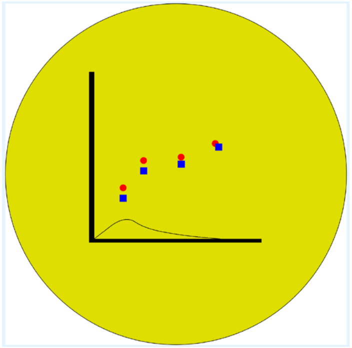






6. Simulate and plot kinetics.Q_total and kinetics.rho
7. Change the input value and press enter. What happened?

Step 11: Add Unit Test



1. Add unit test checker to model
   - TRANSFORM.Utilities.ErrorAnalysis.UnitTests
2. In unitTest parameter GUI assign one or more variables to x
3. Simulate to ensure everything is working
4. Open python/Spyder and
   - Run createUnitScripts.py
   - Run regTestsWin_customBuildPy.py
     - May require custom buildingspy regression script from ModelicaPy (see Session 4)
     - Select "y" if prompted
5. Check the Resources folder. What happened?

Step 12: Extend/Modify TRANSFORM – Heat Transfer

1. Create a new package in TRANSFORM_Training_2019 called HeatTransfer
2. Duplicate the used correlation into the new HeatTransfer package with a modified name and description
   - TRANSFORM.Fluid.ClosureRelations.HeatTransfer.Models.DistributedPipe_1D_MultiTransferSurface.Nus_SinglePhase_2Region
3. Replace Nu_DittusBoelter with Nu_SiederTate
   - Check the heat transfer model. What is missing?
4. Change the hotLeg correlation to the new correlation
5. Simulate. What changed?
   - Use the compare feature to view the difference in effective solid temperature
     - conduction.summary.T_effective

Step 13: Read results in Python/Matlab

Python

- Example in TRANSFORM Resources folder
- Install buildingspy: pip install buildingspy
- Change directory to location of ‘.mat’
- Create notebook or python file and add:

```
from buildingspy.io.outputfile import Reader

r = Reader(“MATFILE.mat’,’dymola’)

time, variable = r.values(‘VARNAME’)
```

Matlab

- Dymola provides useful functions
  - C:\Program Files\Dymola 2020\Mfiles\dymtools
- Example found in Resources folder
- Grab dymget.m and dymload.m from
  - C:\Program Files\Dymola 2020\Mfiles\dymtools

```
% Load the .mat file. If not found see 'err' for error
[dymstr,err] = dymload(fullPath_mat);

% Get time variable. Only need to load once as it is the same for all variables
t = dymget(dymstr,'Time');
```

Step 14: Create an FMU

1. Generate the FMU from the Dymola GUI
2. Open the FMU in FMU Simulator
   - FMPy Github (https://github.com/CATIA-Systems/FMPy)
   - Windows Executable: https://www.3ds.com/products-services/catia/products/dymola/free-downloads/fmu-simulator/
3. Simulate to ensure everything is working

FUNCTIONAL MOCK-UP UNIT

Thank you.

Scott Greenwood

greenwoodms@ornl.gov