MANUAL

HYBRID User Manual

Konor Frick, Daniel Mikkelson, Andrea Alfonsi, Cristian Rabiti,

Idaho National Laboratory

HYBRID User Manual

Konor Frick

Daniel Mikkelson

Andrea Alfonsi

Cristian Rabiti

Contents

1   Introduction

One of the goals of the HYBRID modeling and simulation project is to assess the economic viability of hybrid systems in a market that contains renewable energy sources like wind. The hybrid system would be a nuclear reactor that not only generates electricity, but also provides heat to another plant that produces by-products, like hydrogen or desalinated water. The idea is that the possibility of selling heat to a heat user absorbs (at least part of) the market volatility introduced by the renewable energy sources.

The system that is studied is modular and made of an assembly of components. For example, a system could contain a hybrid nuclear reactor, a gas turbine, a battery and some renewables. This system would correspond to the size of a balance area, but in theory any size of system is imaginable. The system is modeled in the 'Modelica/Dymola' language. To assess the economics of the system, an optimization procedure is varying different parameters of the system and tries to find the minimal cost of electricity production.

1.1   Modelica Models

Idaho National Laboratory (INL) has been developing the NHES package, a library of high-fidelity process models in the commercial Modelica language platform Dymola since early 2013 [1], [2], [3], [4]. The Modelica language is a non-proprietary, object oriented, equation-based language that is used to conveniently model complex, physical systems. Modelica is an inherently time-dependent modeling language that allows the swift interconnection of independently developed models. Being an equation-based modeling language that employs differential algebraic equation (DAE) solvers, users can focus on the physics of the problem rather than the solving technique used, allowing faster model generation and ultimately analysis. This feature alongside system flexibility has led to the widespread use of the Modelica language across industry for commercial applications. System interconnectivity and the ability to quickly develop novel control strategies while still encompassing overall system physics is why INL has chosen to develop the Integrated Energy Systems (IES) framework in the Modelica language.

1.2   Individual Components

The current version of the NHES library employs both third party components from the Modelica Standard Library [5] and TRANSFORM [6] and components developed internal to the project for specific subsystems. For example, the NHES library contains a large variety of models for the development of a high-temperature steam electrolysis plant, a gas turbine, a basic Rankine cycle balance of plant, and a light water nuclear reactor. Components included in the library that support the development of these systems include 1-D pipes, pressurizers, condensers, turbines (steam

and gas), heat exchangers, a simple logic-based battery, a nuclear fuel subchannel, etc. Third party models include numerous additional models including source/sink components (e.g., fluid boundary conditions), additional heat exchanger models, logical components for control system development, multi-body components, additional supporting functions (e.g., LAPACK, interpolation, smoothing), etc. Please see the specific libraries for additional information.

1.3 Hybrid Requirements

The repository itself can be found here: `https://github.com/idaholab/hybrid`

Software requirements are as follows:

1. Commercial Modelica platform Dymola – `https://www.3ds.com/products-services/catia/products/dymola/latest-release/`.

2. Risk Analysis and Virtual ENviroment (RAVEN) – `https://raven.inl.gov/SitePages/Software%20Infrastructure.aspx`

3. Python 3 – `https://docs.conda.io/en/latest/miniconda.html`

4. Microsoft Visual Studio Community Edition. – `https://visualstudio.microsoft.com/downloads/`

Note: Steps 3 and 4 can be accomplished by following the RAVEN installation instructions in step two. The installation procedure will be outlined below. All physical models are run within the Dymola simulation framework graphical user interface (GUI). Background information on the Modelica as a language as well as good general guidance on coding practices can be found at the two references shown below.

1. `https://webref.modelica.university/`

2. `https://mbe.modelica.university/`

2 HYBRID Installation Procedure

2.1 Overview

The installation of the HYBRID repository is a straightforward procedure; depending on the usage purpose and machine architecture, the installation process slightly differs.

In the following sections, the recommended installation procedure is outlined. For alternatives, we encourage checking the RAVEN wiki. The windows 10 machine on which HYBRID is tested and developed, uses the standard installation procedures outlined below.

The installation process will involve four steps:

- Installing prerequisites, which depends on your operating system;

- Installing conda;

- Installing RAVEN.

- Installing HYBRID.

2.2 Cloning the Hybrid Repository

The first step in installing the package is to clone the HYBRID repository. To do this, use

```
git clone https://github.com/idaholab/HYBRID.git
```

This will download the repository into a folder called 'hybrid'. To go inside the folder, use

```
cd hybrid
```

2.2.1 Install RAVEN and its plugins as a sub-module

The next step is to download and install RAVEN and the submodule (e.g. TEAL, HERON) plugins as a sub-module of the HYBRID repository.

A submodule allows you to keep another Git repository in a subdirectory of your repository. The other repository has its own history, which does not interfere with the history of the current repository. This can be used to have external dependencies such as third party libraries for example.

In order to get RAVEN do the following in the hybrid folder

```
git checkout devel
```

Update the Branch

```
git pull
```

to add RAVEN as a submodule

```
git submodule update --init --recursive
```

Install and Compile RAVEN. Once you have downloaded RAVEN as a sub-module, you have to install it. go to the RAVEN Wiki for information about how to install it. Run all the tests outlined in the RAVEN wiki.

2.2.2 Inform the Framework Paths

In order to set up the hybrid repository, you must inform the framework about the location of the Dymola python interface. For doing so, navigate to the hybrid directory:

to add RAVEN as a submodule

```
cd <path to your hybrid repository>/hybrid
```

Run the following command:

```
./scripts/write_hybridrc.py -p DYMOLA_PATH
```

Where DYMOLAPATH is the path to the python interface egg folder in the DYMOLA installation locally. For example:

```
./scripts/write_hybridrc.py -p
        "/c/Program\_Files/Dymola\_2020x/Modelica/Library/
⎵⎵⎵⎵⎵⎵⎵⎵python_interface/dymola.egg"
```

2.3 Setup of Dymola for the Regression Testing System

To properly setup the Hybrid repository to work with the regression system one needs to download Dymola as mentioned above and activate the license. Once those two steps are complete the dymola.mos file needs to be edited. The dymola.mos file is the file that tells Dymola what libraries

4

to load when the application is opening and where the working directory is located. For the automatic regression test system to properly test the downloaded library the proper NHES library must be loaded automatically by dymola in the dymola.mos file located at C:/Program Files/Dymola 2020/insert/dymola.mos for example. To properly run the tests the NHES and the TRANSFORM libraries need to be automatically loaded by Dymola upon startup. This can be accomplished by adding to the Dymola.mos until it looks something like:

```
RunScript("$DYMOLA/insert/displayunit.mos", true);
definePostProcessing("SDF output", "Convert result file to SDF format",
"Modelica.Utilities.System.command(\"\\\"%DYMOLA%/bin/dsres2sdf\\\"
%RESULTFILE%.mat %RESULTFILE%.sdf\")");

openModel("C:\Users\FRICKL\Desktop\TRANSFORM3_20_2020\TRANSFORM-
Library\TRANSFORM\package.mo"); //Loads Transform package.mo

openModel("C:\msys64\home\FRICKL\hybrid_devel\hybrid\models
\NHES\package.mo");//Loads NHES package from hybrid directory

cd("C:\Users\FRICKL\Desktop\TESsystem");
//Place where all the Dymola runs will occur.
```

Having the Dymola.mos file written like this will allow Dymola to automatically load all the needed libraries for Regression testing when conducted using the Regression Test Harness. It should be noted that typically the dymola.mos file will need its permissions to be changed to allow a user to write this. On a Windows machine this is done by running as an administrator on the system and changing the properties of the file to include "read and write" rights, as opposed to "read-only" rights.

2.4   Run Regression tests related to the HYBRID project

Now that the dymola.mos file has been edited to automatically load the TRANSFORM and NHES packages one can run all tests associated with the HYBRID repository. To do this follow the instructions below.

```
cd <path to your hybrid repository>/hybrid
./run_tests -jX -lY --only-run-types ZZ
```

where:

- "X" is the number of processors to use for testing

- "Y" is the maximum load to limit to execute tests

- "ZZ" is the subtype of tests to be run. Currently only "raven" and "dymola" are available.

If all tests need to be run, just execute the following command.

```
cd <path to your hybrid repository>/hybrid
./run_tests -jX -lY
```

The output will look like the following:

```
#######################################################################
#
#  Testing of Hybrid RAVEN Modules  (./run_tests)
#
#######################################################################
/c/msys64/home/FRICKL/cleaning_hybrid/hybrid
Found $DYMOLA_PATH and set to C:/Program Files/Dymola 2021/Modelica/Library/python_interface/dymola.egg
Loading raven_libraries conda environment ...
CONDA
... Run Options:
... Mode: 1
... Verbosity: 0
... Clean: 0
... Mode: CONDA
... Conda Defs:
... Loading RAVEN libraries ...
... Detected OS as --os windows ...
... Using Python command python
... $RAVEN_LIBS_NAME set through raven/.ravenrc to raven_libraries
... >> If this is not desired, then remove it from the ravenrc file before running.
... >> RAVEN environment is named "raven_libraries"
... found conda path in ravenrc: C:/Users/FRICKL/AppData/Local/Continuum/miniconda3/etc/profile.d/conda.sh
... >> If this is not the desirable path, rerun with argument --conda-defs [path] or remove the entry from raven/.ravenrc file.
... Found conda definitions at C:/Users/FRICKL/AppData/Local/Continuum/miniconda3/etc/profile.d/conda.sh
conda 4.8.3
raven_libraries              C:\Users\FRICKL\AppData\Local\Continuum\miniconda3\envs\raven_libraries
... Found library environment ...
... Activating environment ...
... Activating environment ...
... done!
rook: loading init file "C:/msys64/home/FRICKL/cleaning_hybrid/hybrid/scripts/rook.ini"
rook: ... loaded setting "add_non_default_run_types = dymola,raven"
rook: ... loaded setting "add_run_types = dymola,raven"
rook: ... loaded setting "test_dir = tests"
rook: ... loaded setting "testers_dirs = scripts/testers,raven/scripts/TestHarness/testers/"
rook: found 27 test dirs under "tests" ...
rook: loading init file "C:/msys64/home/FRICKL/cleaning_hybrid/hybrid/scripts/rook.ini"
rook: ... loaded setting "add_non_default_run_types = dymola,raven"
rook: ... loaded setting "add_run_types = dymola,raven"
rook: ... loaded setting "test_dir = tests"
rook: ... loaded setting "testers_dirs = scripts/testers,raven/scripts/TestHarness/testers/"
(1/27) Success( 40.44sec) tests\dymola_tests\BOP_L1_Boundaries_a_Test\
(2/27) Success( 41.22sec) tests\dymola_tests\BOP_L1_Boundaries_b_Test\
(3/27) Success( 15.27sec) tests\dymola_tests\Desalination_1_pass\
(4/27) Success( 15.84sec) tests\dymola_tests\Desalination_2pass_mixing\
(5/27) Success( 14.17sec) tests\dymola_tests\Desalination_2_pass\
(6/27) Success( 24.64sec) tests\dymola_tests\Desalination_NHES_basic\
(7/27) Success( 22.12sec) tests\dymola_tests\Desalination_ROmodule\
(8/27) Success( 42.10sec) tests\dymola_tests\Desalination_NHES_complex\
(9/27) Success( 17.21sec) tests\dymola_tests\GTTP_Test\
(10/27) Success( 36.21sec) tests\dymola_tests\Generic_Modular_PWR\
(11/27) Success( 23.93sec) tests\dymola_tests\HTSE_Power_Test\
(12/27) Success( 32.00sec) tests\dymola_tests\HTSE_Steam_Test\
(13/27) Success( 39.60sec) tests\dymola_tests\NSSS_test\
(14/27) Success( 55.31sec) tests\dymola_tests\NuScale_4Loop\
(15/27) Success( 36.51sec) tests\dymola_tests\NuScale_Nominal_Test\
(16/27) Success( 15.14sec) tests\dymola_tests\Simple_Breakers_Test\
(17/27) Success( 26.16sec) tests\dymola_tests\NuScale_primary_test\
(18/27) Success( 15.70sec) tests\dymola_tests\StepDownTurbines\
(19/27) Success( 16.61sec) tests\dymola_tests\StepDownTurbines_complex\
(20/27) Success( 14.34sec) tests\dymola_tests\Supervisory_Control_Test\
(21/27) Success( 14.31sec) tests\dymola_tests\Test_Battery_Storage\
(22/27) Success( 34.01sec) tests\dymola_tests\Test_Thermal_Storage\
(23/27) Success( 37.58sec) tests\dymola_tests\Thermocline_Cycling\
"failing"
(24/27) Skipped( None! ) tests\dymola_tests\TightlyCoupled_FY18_Battery\
"failing"
(25/27) Skipped( None! ) tests\dymola_tests\TightlyCoupled_FY18_TES\
(26/27) Success( 25.44sec) tests\dymola_tests\Thermocline_Insulation\
(27/27) Success( 31.37sec) tests\raven_tests\train\TrainArmaOnData
```

```
PASSED:  25
SKIPPED:  2
FAILED:  0
```

3   Running and Creating New Code

The physical modelica models are the cornerstone of the Hybrid repository. They are designed to represent physical industrial processes that can be configured into different potential integrated energy systems (IES). Table 1 gives an overview of the main types of integrated energy systems, along with models currently incorporated in the hybrid repository.

3.1   Understanding and Running Existing Models

The hybrid repository is broken down using the templating system shown in Figure 1. The top level is the overall system package which incorporates all of the Modelica models contained within the NHES package. Then inside of the NHES package are the different subpackages (Systems, Electrical, Thermal, etc...). Within each of the subpackages are further subpackages as seen in the Systems package. Within the Systems package there are further subpackages called *SubSystem Category* (Examples, PrimaryHeatSystem, EnergyStorage, etc...). Then within these SubSystem Categories there is yet another level of subpackage that is called *SubSystem_Specific*. Within the *SubSystem_Specific* category is where development takes place and potential configurations of the different processes take shape. Inside each SubSystem_Specific there is a template that includes *Examples*, *Subsystem Dummy*, *CS_Dummy*, *ED_Dummy*, Data, BaseClasses, and usually a Components folder. For existing systems the Examples folder contains a runnable example the user can execute to see how the code runs at a top level and what scenarios it is capable of running. An example of which is depicted in Figure 2. For each example the user can double click on the main system which will open the table in the upper left hand corner of Figure 2 which provides inputs for the user to change parameters about the system. Then if the user wishes to modify the control system utilized they can either choose from the drop-down menu, or click the button at the right of the "CS" line to open the table in the lower left hand section which provides options to delay when different control systems come online.

These example tests provide a good way for the user to become associated with the large subsystem in terms of how they work and the different parameters that can be utilized to tune and interact with the models. In addition to the examples file a deeper understanding of the model can be realized by looking into the component structure of the model. This is typically accomplished through looking at the filled-out Subsystem Dummy section. For the Westinghouse 4-Loop plant this can be seen in Figure 3. This model includes several subcomponents connected into a singular model. Each model with its' own set of parameters. Using this version of the model it is possible to discern the inner workings of the model in terms of sensors, physical descriptions of the code, inlet and outlet conditions, and system dependencies. In addition to the SubSystem Dummy section, large process models typically include a control system section which is created from the CS_Dummy file in the branch. These control system files can be added as a control system for the Subsystem to control different valves, pumps, and control drives within the process from the drop-

Table 1: Examples of large-Scale Systems within the Hybrid repository used in the creation of Integrated Energy Systems.

| Category	Description	Specific Example
Primary Heat System	Provides base load heat and power	Nuclear Reactor
Energy Manifold	Distributes thermal energy among subsystems	Steam Manifold
Balance of Plant	Serves as primary electricity supply from energy not used in other subsystems	Turbine, condenser, and feed-train
Industrial Process	Generates high value product(s) using heat and electricity from other systems	Steam Electrolysis, gas to liquids, reverse osmosis
Energy Storage	Serves as energy buffer to increase overall system robustness and system that can increase profits during highly fluctuating energy prices	Electric Batteries, Two-Tank Sensible Heat Storage, Thermocline
Secondary Energy Source	Delivers small amounts of topping heat required by industrial processes or rapid dynamics in grid demand that cannot be met the remainder of the system	Natural Gas Turbine
Switch Yard	Distributes electricity among subsystems, including the grid	Electricity Distribution
Electrical Grid	Sets the behavior of the grid connected to the IES	Large Grid Behavior
Control Center	Provides proper system control and test scenarios	Control System/ Supervisory Control System
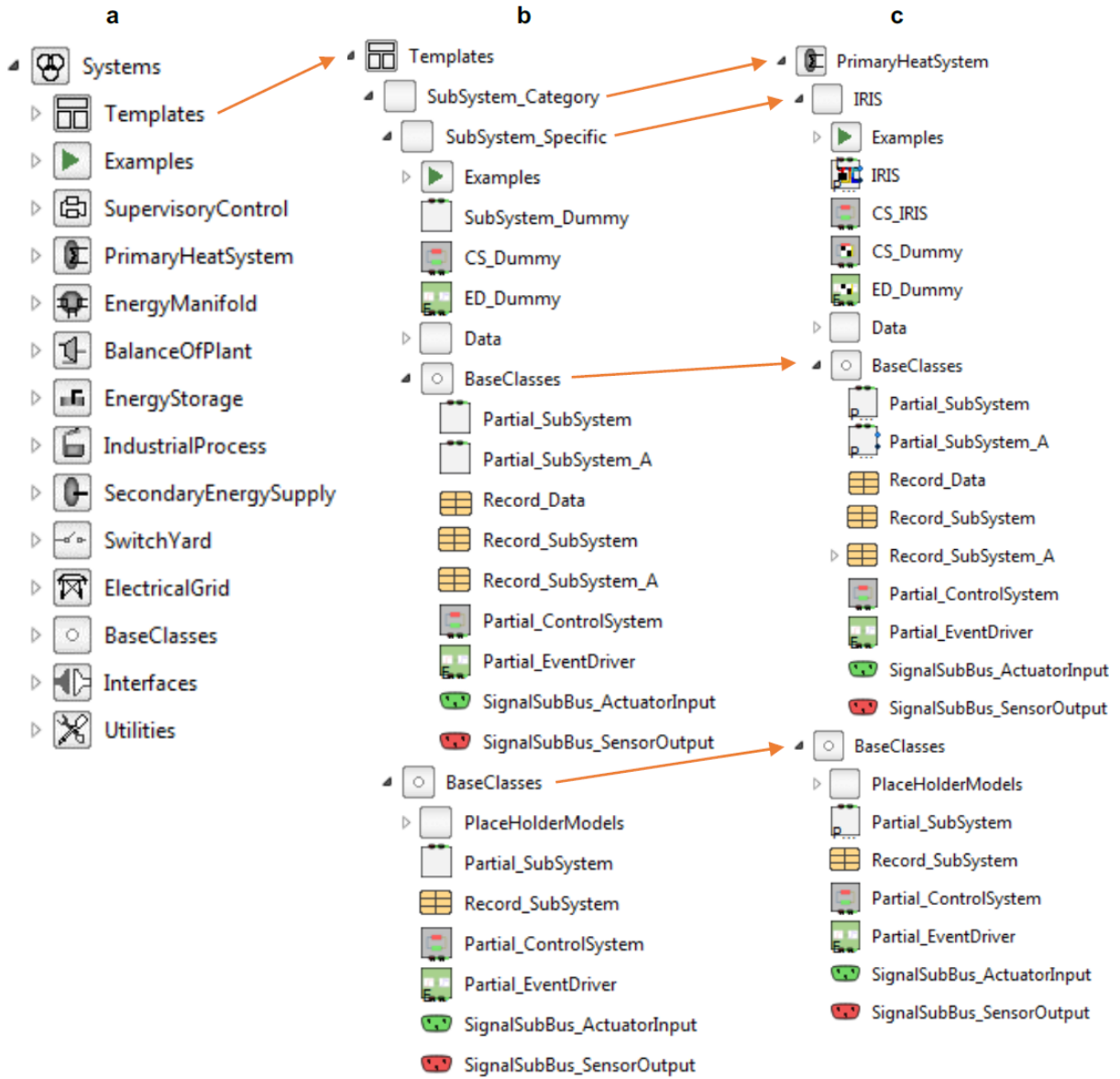
Figure 1: a) Overall Modelica package, b) template structure for creating new subsystem categories and specific subsystem models within a category, c) example of a specific implementation of a primary heat system using the template approach.
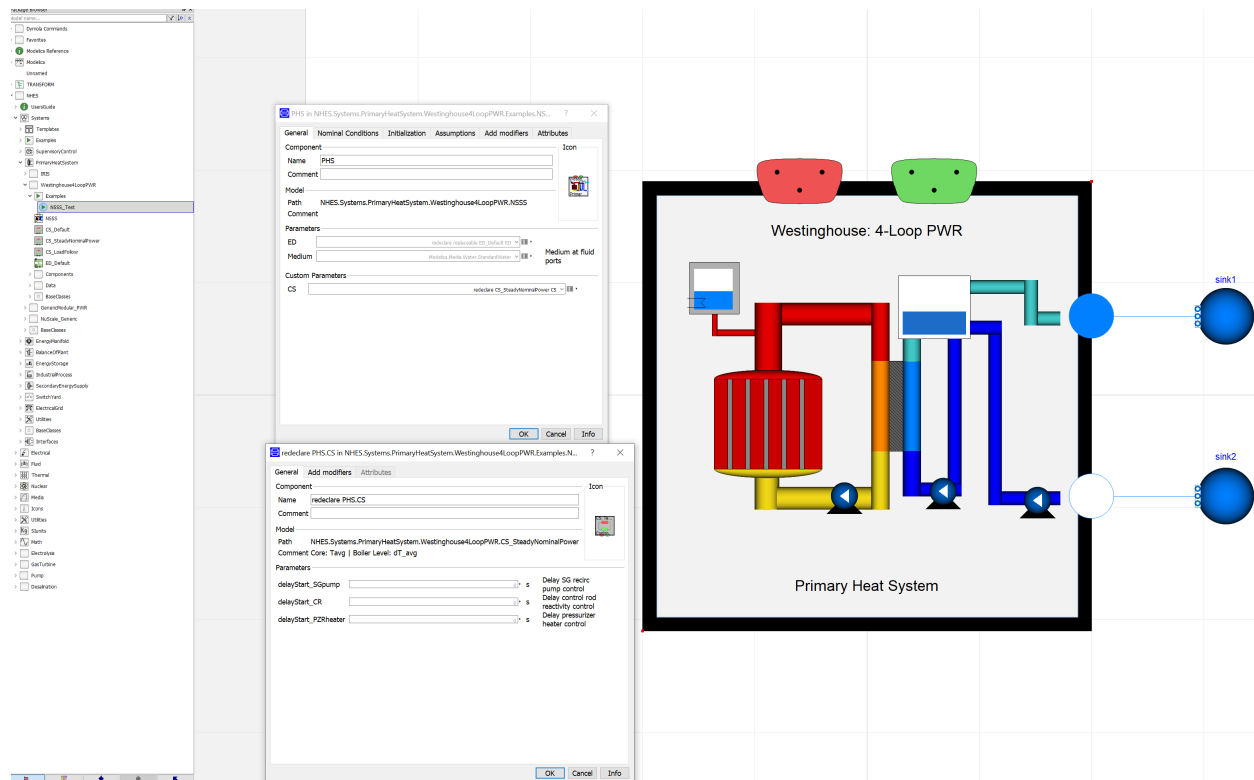
Figure 2: Exploded view of the NSSS_Test example within the NHES library with control system options opened up

down menu in the "CS" section seen in Figure 2. large process systems may have several different potential control systems based upon what type of Integrated Energy System they are operating within. An illustration of one of the Westinghouse Control systems can be seen in Figure 4.
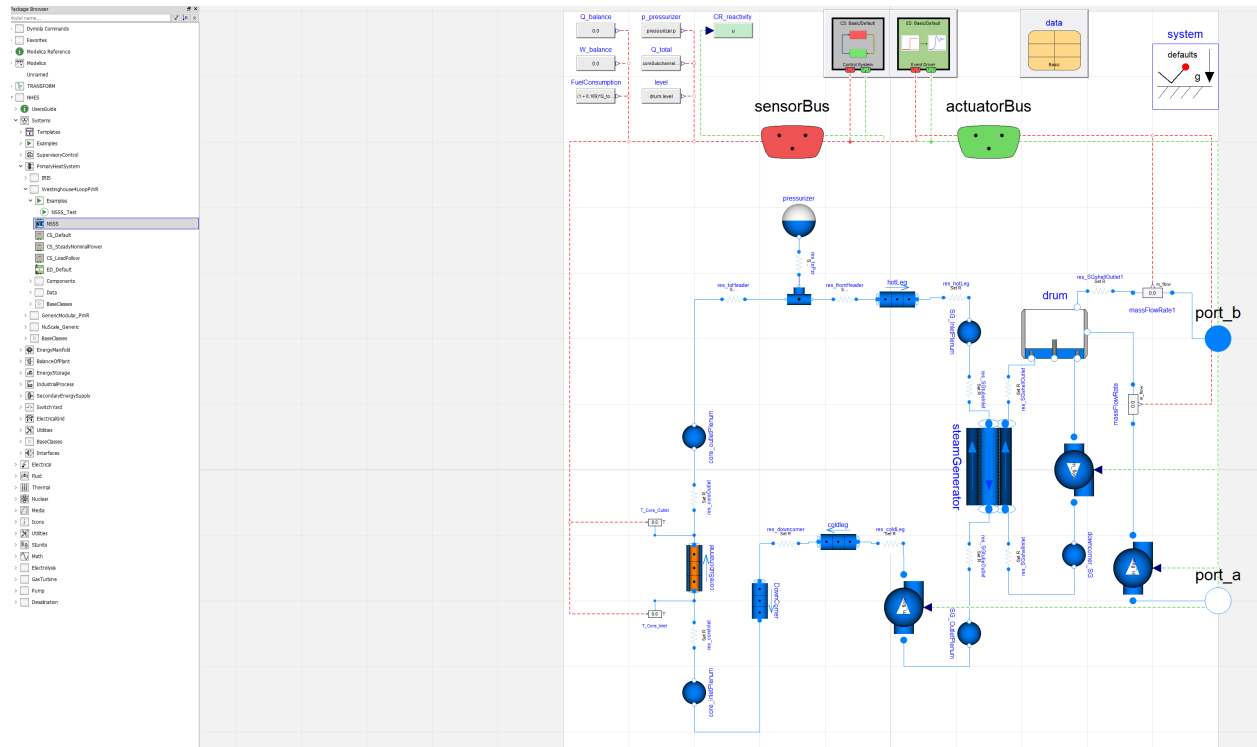

Figure 3: Subsystem for the Westinghouse-4 Loop model.

Assuming the user is creating a new package with new components specific to the model it is best to include those models with a "components" folder in the subpackage containing the "Base-Classes" folder. The Data folder is typically where the main data structures in terms of "records" of kept for the process model. Records are files that are intended to be used as an input deck to the main model for use as a set of "parameters" the components will read from. The ED_dummy file within the *Subsystem_Specific* category is the Event Driver file and is rarely used and can be ignored from a user perspective.

3.1.1 Modifying Existing Models for Specific Runs

A starting point from which a user can begin model development and analysis is from an existing Example model. To properly edit the *Examples* within the hybrid repository while still maintaining the regression system one needs to create a duplicate model of the example file that is to be edited. This can be done by right clicking on the file as shown below and creating a duplicate class.

This file will the be placed in the Examples folder where edits can be made to it for new and
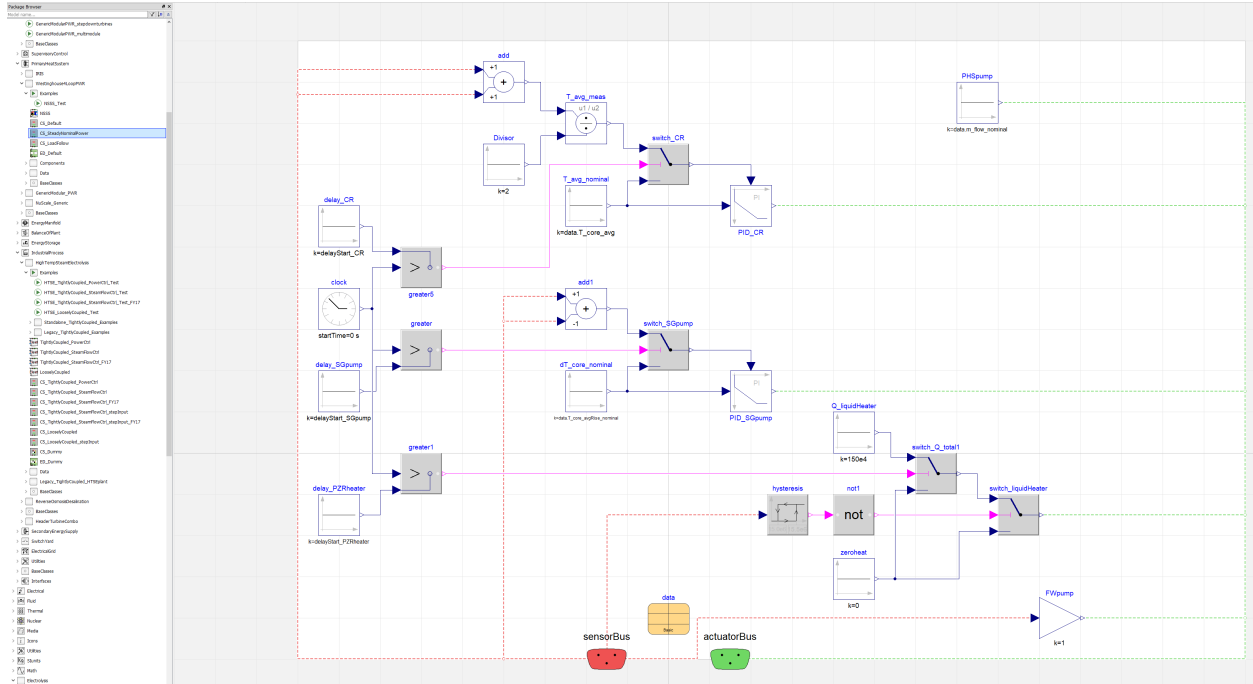
Figure 4: Control System (CS) for the Westinghouse 4 Loop model

unique runs. This includes things such as new control schemes, sizing, timeruns, etc..

3.2   Configuring Existing Models into Integrated Energy Systems

Each subsystem of the Integrated Energy Systems is inherently interesting on its own and large spans of time can be spent researching and fine tuning them independently. However, the developer team is aware that in the evolving energy landscape, and to the extent users will come across this repository, that integrated energy systems are the primary focus.

This focus includes systems that involve the distribution of heat and electrical energy among several subsystems and the control schemes utilized to accomplish this. Therefore, this section seeks to provide an introductory understanding of how to connect subsystems together within the Hybrid repository. To accomplish this the NuScale_Coupling_Test Example will be created starting from the GenericModularPWR_park system.

The first step is to take a similar example that has the Supervisory Control System in the top level. In this case the GenericModularPWR_park was used. A duplicate class was created and all the components aside the Steam Manifold, Turbine, Simple Breakers, infinite grid, supervisory control system, delay start, and data capacity were removed. See below.

Then the primary side of the NuScale was added in this case the *NuScale_Taveprogram* version
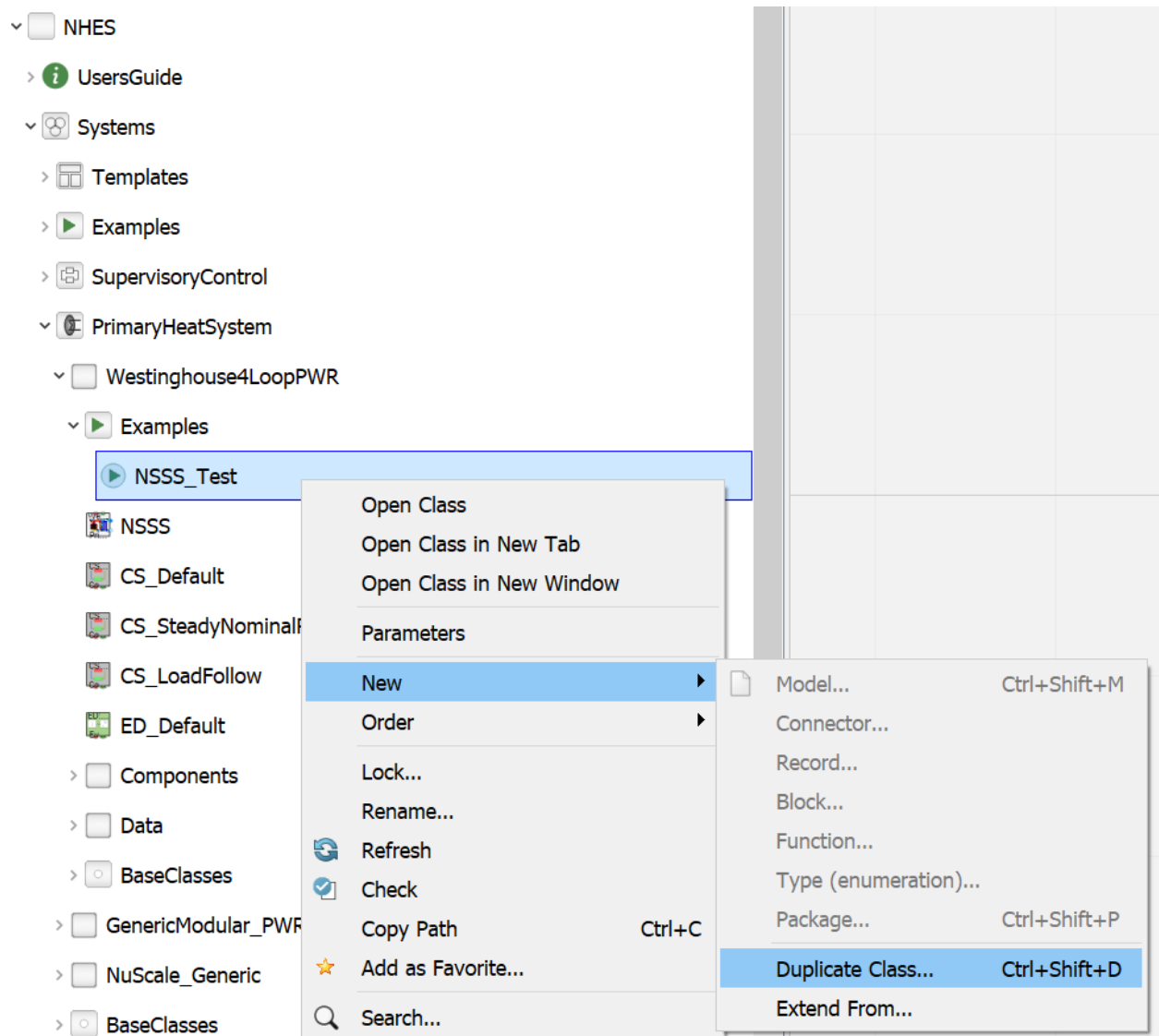
13

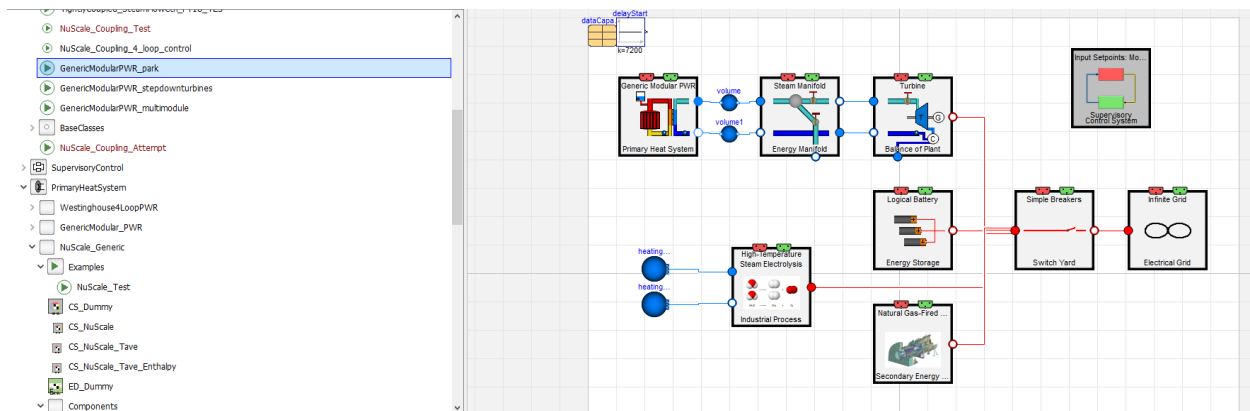**Figure 5:** Creating a duplicate class for model runs.

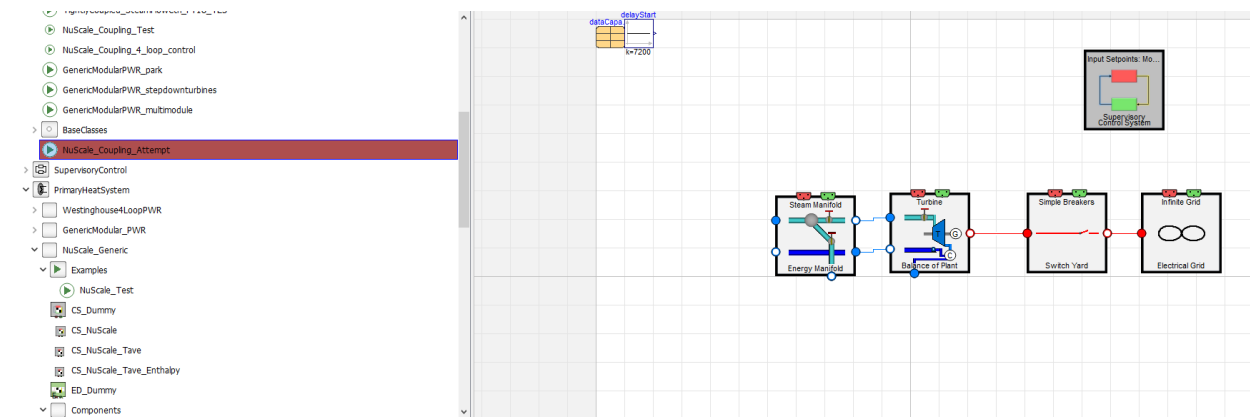**Figure 6:** Initial Integrated Energy System Starting Point

Figure 7: Rearrangement of Initial Energy System
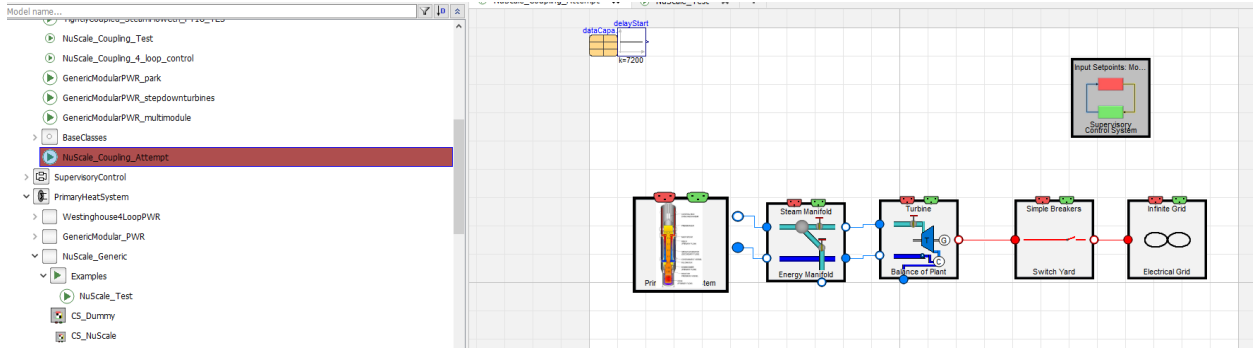
of the NuScale primary unit.

Figure 8: Creation of NuScale Energy System

Then from this point it is a matter of telling the systems what control schemes to use. For this system the reactor operates to meet a certain primary system average temperature in accordance with the turbine output. To input this the control system: PrimaryHeatSystem.NuScaleGeneric. CS_NuScale_Tave was used with input: W_turbine = BOP.powerSensor.power and W_Setpoint = SC.W_totalSetpoint_BOP, see Figure 9. And the turbine control scheme is modified to reflect a once through system type control strategy where the turbine control valve operates to meet a constant pressure in the turbine, Figure 10. While it is noted that is not the official control strategy strictly speaking for the NuScale system nor is it the one used in load following scenarios in the hybrid repository, it does provide a baseline for which to control the system and modifications can be made from this point. The power setpoints in the BalanceOfPlant.Turbine.CS_OTSG_Pressure control module are 160MW for both Reactor_Power and Nominal_Power while p_nominal parameter is set to BOP.port_a_nominal.p to ensure a single parameter value is carried throughout the system. Additionally, W_totalSetpoint is set to SC.W_totalSetpoint_BOP, Figure 10.

To complete the construction of the model the systems need to match on the boundaries. To do this the values from the primary heat system need to be transferred to the Steam Manifold under the nominal values tab, Figure 11 and 12.

3.3 Test Creation

To create a regression test once a user develops an example test in the Dymola NHES library can be accomplished through a couple of settings. In the Dymola simulation setup tab in the output tab uncheck the store at variable events box. Then click store in model button and check the output box, click ok, then click ok again, then Save the model. Example settings are shown in Figure 13.

In the simulateModel command one of the following two flags is required. Either "*numberOfIntervals*" or "*OutputInterval*". numberOfIntervals tells dymola how many output intervals to make. OutputInterval tells dymola at what timestep interval should an output be present for comparison.

Figure 9: Primary System Controller Settings

Figure 10: Turbine Control Settings

Figure 11: Port a Boundary Values of the Energy Manifold

Figure 12: Port b Nominal Values of the Energy Manifold
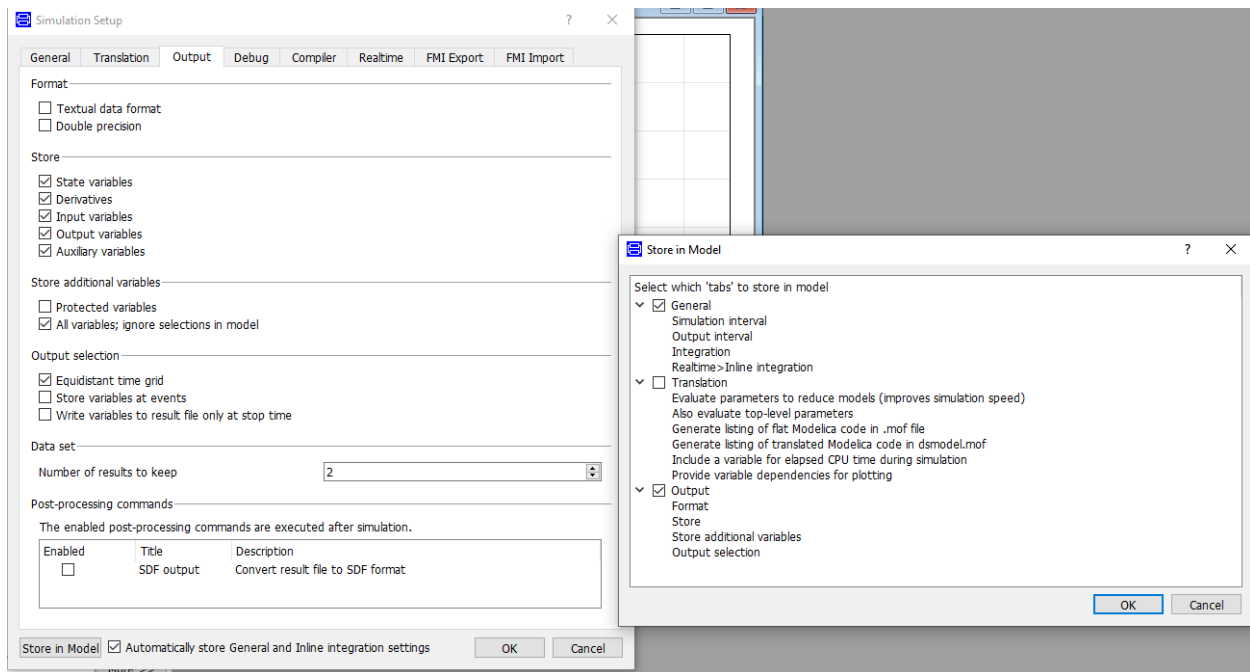
Figure 13: Settings to Create a proper mat file for a gold folder test

The .mat file in the gold folder will need to be run using the same simulateModel command that is present in the .mos file being created.

These can be selected in the Simulation Setup tab of the Dymola GUI, Figure 14, and should carry down to the command you copy and paste in the .mos file. An example is shown below of the simulation setup tab.

Then run the simulation, (ideally a test should take less than 100 seconds). On the simulation tab in the command line copy the simulation command. Example below:

```
simulateModel("NHES.Systems.EnergyManifold.SteamManifold.
Examples.SteamManifold_Test", stopTime=100, numberOfIntervals=100,
method="Esdirk45a", resultFile="SteamManifold_Test");
```

This command should then be added to a file and named something like Test_Example.mos. The command can be found in the Simulation Setup tab of the Dymola GUI once you hit simulate

Then in folder /path/to/hybrid/hybrid/tests/dymola_tests create a folder named Test_YourModel.

Create a *gold* folder in the new folder, drop the .mat file from your simulation that is named resultFile="SteamManifold_Test" from your simulateModel command into the gold folder. The .mat file is created in your working directory in Dymola. Then in the main Test_YourModel folder drop the Test_Example.mos file and create a tests file open it up and place the following in it:
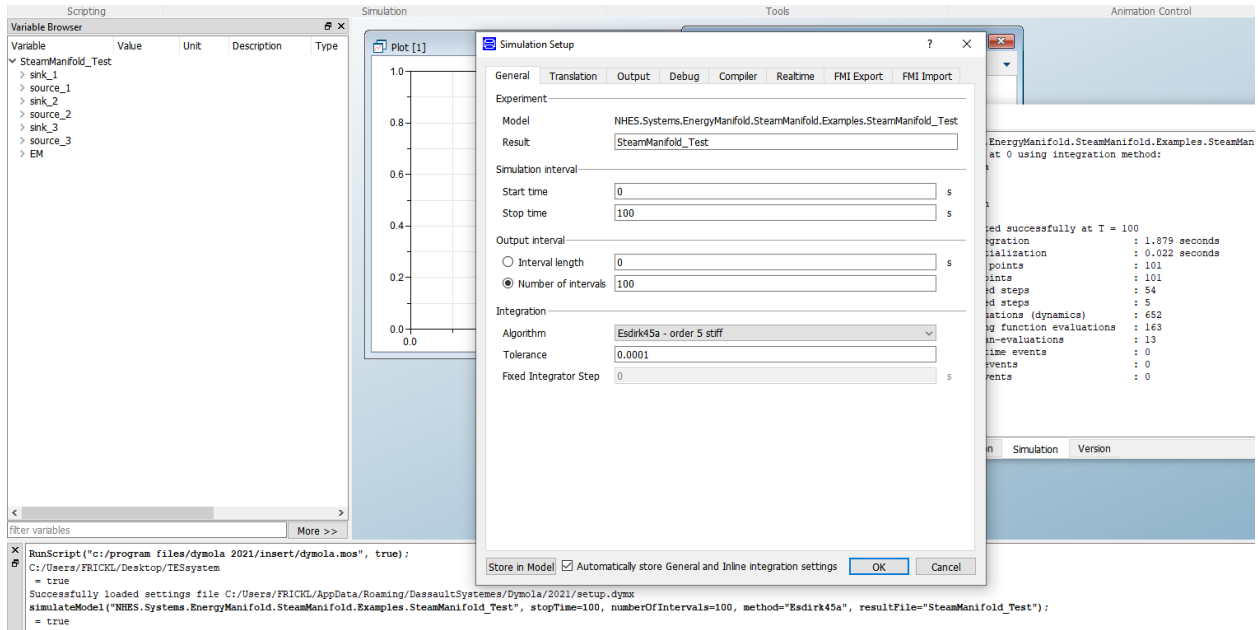
Figure 14: Simulation Setup

```
[Tests]
  [./]
   type = 'HYBRIDTester'
   input = 'Test_Example.mos'
   workingDir = '.'
   output = 'SteamManifold_Test.mat'
   dymola_mats = 'SteamManifold_Test.mat'
   rel_err = 0.001
  [../]
[]
```

where SteamManifold_Test.mat should be your result .mat file name, rel_error is the amount of error allowed between the gold file and the regression test output, and Test_Example.mos is the run script created.

3.4   Advanced Test File Options utilized for complex models

For complex models the initialization phase of a simulation can take the Modelica solvers a significant amount of time to find an initialization point. This occurs due to the highly nonlinear nature of the underlying physical equations. A way to avoid such situations is to provide a restart file to bypass the initialization phase of the simulation. A restart file is automatically created at the end of each simulation as the dsfin.txt file created in the folder where the simulation is run.

This file includes the final values of the previous simulation from which the new model can restart. Move this file to the gold folder for your new testing system. Once this file is created it can then be loaded automatically via the continue button in Modelica under the Simulation Tab. Select $Continue \rightarrow ImportInitial \rightarrow dsfin.txt$. See Figure 15.
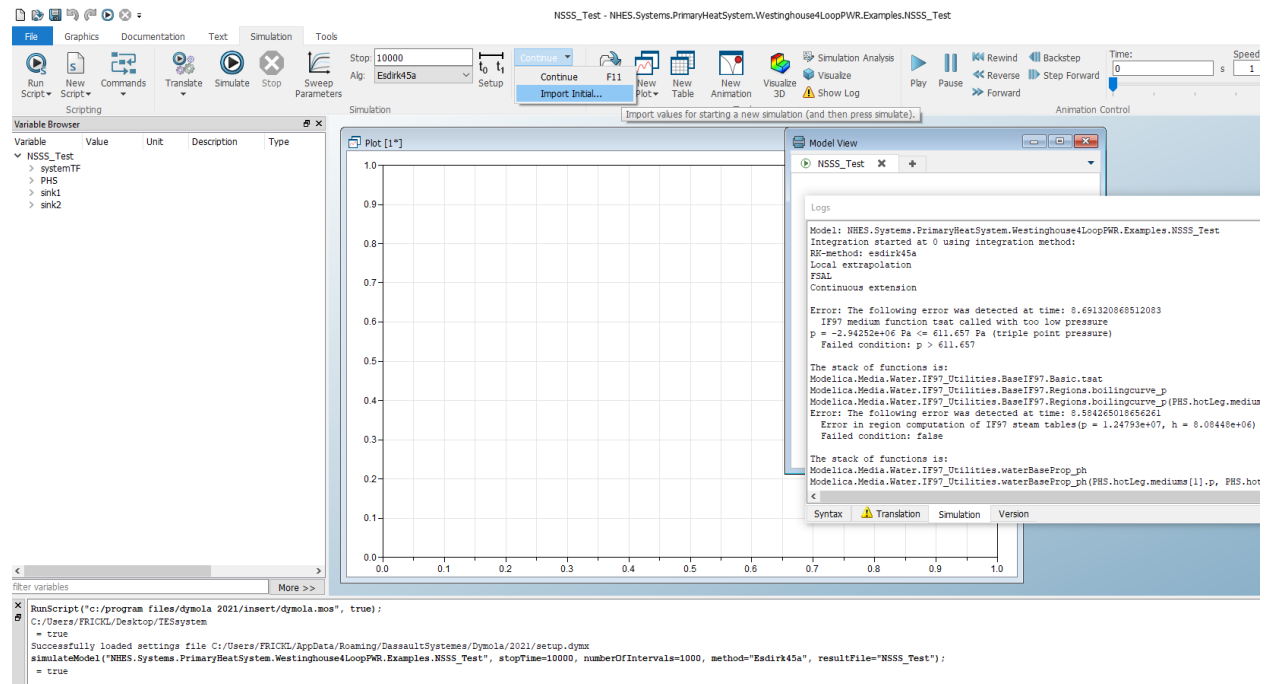
Figure 15: Import Initial conditions from a previously run simulation

Then once the dsfin.txt file is loaded go into the Setup tab and move the time back to start from zero and the end time to the desired simulation point for the test, shown in Figure 16. This is necessary since Dymola assumes the user wants to restart the simulation from where it ended in time as well. This is not the case for the test. Instead the goal is to skip the initialization phase of the simulation and provide a clean solution with which to compare.

Simulate this model and save the result file, in this example "NSSS_Test.mat" and place it into the gold folder of the testing system. Additionally, copy and paste the simulateModel command that is in the Dymola GUI as the last line of your .mos script file for the test. The first two lines should be translateModel to make sure the right model is loaded into the equation set, followed by the importInitial command that loads all the values into the translated Model. The final command should be the simulateModel command. The .mos file should look something like what is shown below.

```
translateModel("NHES.Systems.PrimaryHeatSystem.Westinghouse4LoopPWR
.Examples.NSSS_Test");

importInitial("./gold/dsfinal.txt");
```

23
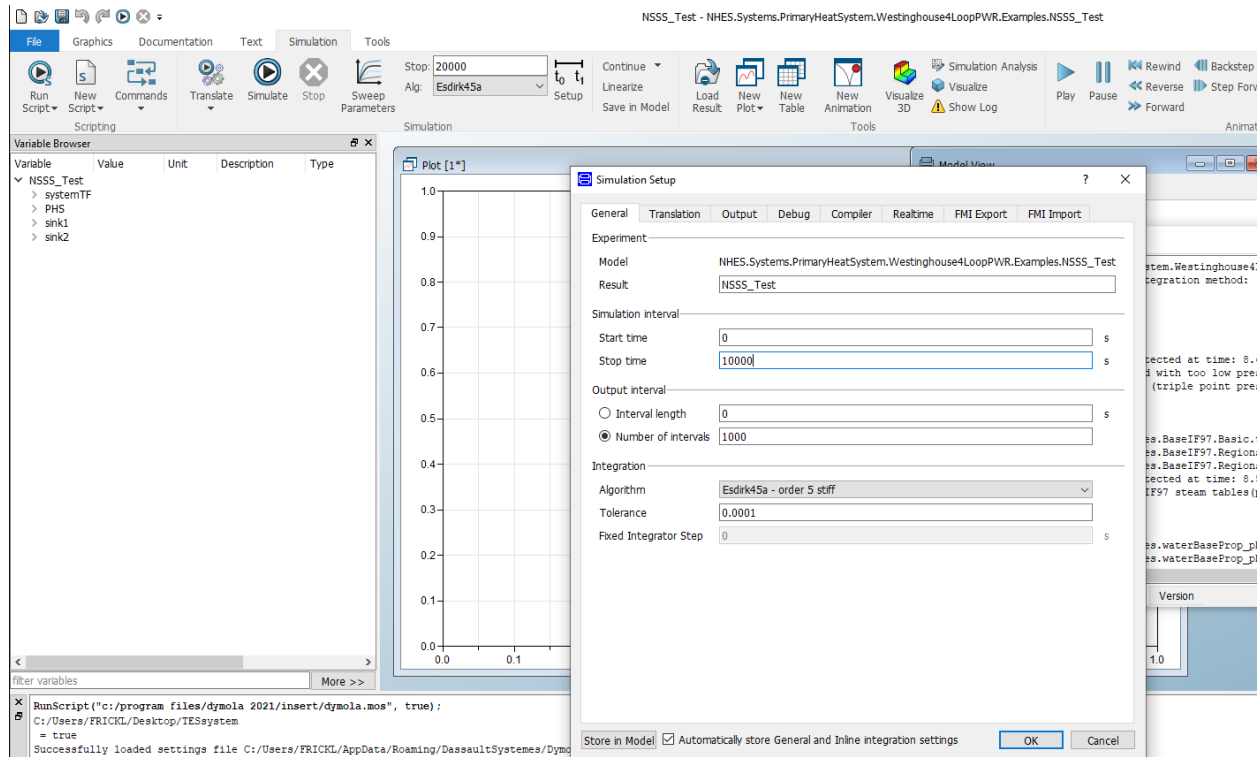
Figure 16: Realign the Simulation Time

```
simulateModel("NHES.Systems.PrimaryHeatSystem.Westinghouse4LoopPWR.
Examples.NSSS_Test", stopTime=10000, numberOfIntervals=250,
method="Esdirk45a", resultFile="NSSS_Test");
```

4 Model Description

It is the intent of this document to provide a level of understanding of each of the process models sufficient to allow users, with some background of Modelica, the ability to integrate, modify top level parameters, and run simulations of Integrated Energy Systems. Advanced users will be able to use the models as they see fit, but the descriptions provided here will not necessarily explain all facets of the models in detail.

4.1 Primary Heat System

In the Hybrid repository there are four potential primary heat sources: The Four-Loop PWR plant, the Generic Modular PWR, a natural circulation SMR power plant, and the Natural Gas Fired Turbine. Generally, we consider the Natural Gas Fired Turbine as a peaker unit and thus will save its' discussion and coverage for the secondary power source section of the Model Descriptions.

4.1.1 Four Loop Pressurized Water Reactor

The Four loop PWR system, Figure 17, is designed to be consistent with publicly available information for the Westinghouse plant design [7]. This is a Pressurized Water reactor with a nominal thermal power of 3400MWt and has control systems designed to output 1100MWe. All system parameters can be found in the SubSystem model under the "data" record. The steam generator is of U-tube design and operates at a nominal pressure 1000psia. Reactivity feedback can be found in the coreSubchannel module alongside an external source of activity that is designed to provide reactivity feedback from the control rods. Reactivity in the core is based on a point kinetics models, that includes feedback from fission products, boron, fuel temperature, and moderator temperature. System decay heat is calculated from the TRANSFORM package via an eleven-group decay heat correlation from the TRACE user manual.

4.1.2 Generic Modular PWR

The generic modular PWR unit, Figure 18, is sized to be 160 MWt with 50MWe output as is consistent with the NuScale power module. However, the generic modular PWR does not operate under natural circulation but instead operates under forced flow. Therefore, this unit provides more stability in the code since it does not rely on density differentials to drive flow. This makes the unit less useful than is the NuScale style reactor modeled below, but it does provide the user a power input consistent with NuScale style systems but without the need to tune system geometries, friction factors, etc.. to meet the proper flow dynamics. As with the Westinghouse plant the data file is included in the subsystem model and has reactivity controls within the core submodule. The
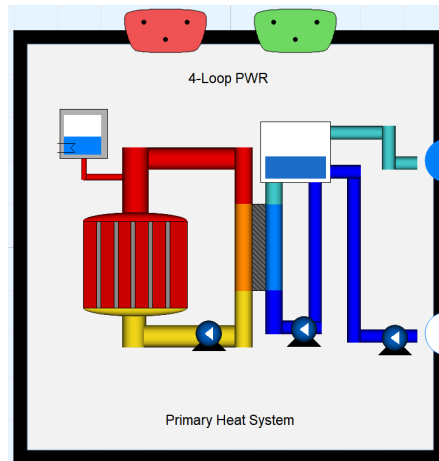
Figure 17: Top View of the Four-Loop PWR Plant

Generic Modular PWR relies heavily on the TRANSFORM library for its subcomponents. The steam generator is a once through design with geometrical orientation consistent with a helical coil steam generator.
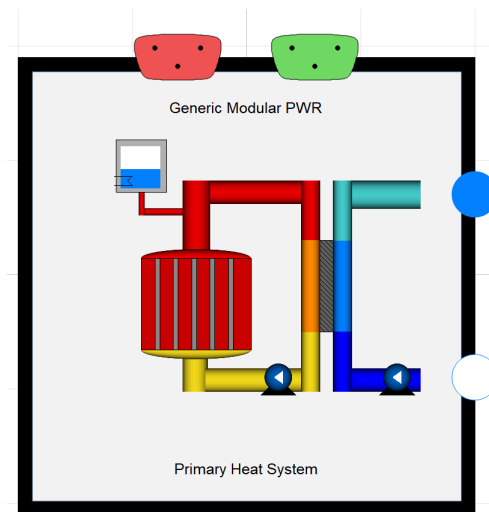
Figure 18: Top Level Depiction of the Generic Modular PWR in the NHES package.

4.1.3   Natural Circulation Small Modular Reactor

The natural circulation SMR power module,Figure 19, is an integral pressurized water reactor (IPWR) that operates with a nominal thermal power of 160 MWt capable of producing 50 MWe to the electric grid. Integral designs are fully self-contained, eliminating the need for large main steam lines that can potentially lead to large break loss of coolant accidents (LOCA). Instead the

26

primary system has only an inlet of feed water into the bottom of the helical coil steam generator and an exit point for steam at the top of the steam generator. All sizes for components is held within the data record in the sub-system. These sizes are consistent with NRC design documentation that can be publicly viewed on the NuScale NRC design certification page.

The primary system does not include any pumps but instead operates under natural circulation. Natural circulation reactors rely on the height and density differentials between hot and cold water to drive circulation of water through the core. Through elimination of primary coolant pumps an entire class of accident scenarios is eliminated. Modeling efforts in this report focused on three main efforts: matching thermal and electric output, matching system geometry, and matching natural circulation efforts in the system via flow rates and temperature differentials. The primary side of the module has heights and cross-sectional areas in accordance with NRC design certification material. The primary and secondary sides were modeled in their entirety. The helical coil steam generator was modeled as a once through steam generator where the secondary side is on the inside of the tubes and the primary side fluid run along the outside of the tubes. The full report on this module is available on OSTI [4].
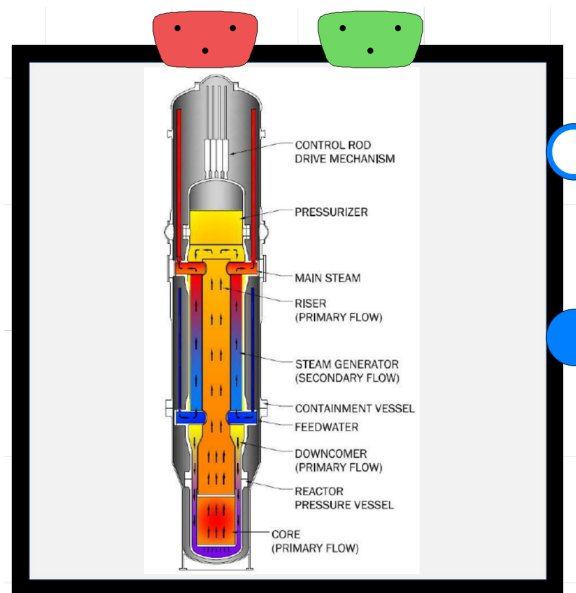
Figure 19: Top Level Depiction of the SMR System in the NHES package.

4.2 Energy Manifold

The energy manifolds intention is be a diversion module to as many different subunits as needed for fluid diversion. It consists of a series of pipes that can be extended to "n" submodules, see Figure 20. The unit has the capability of utilizing control schemes, however in many practical applications the control schemes are encapsulated within the subprocesses as opposed to within the energy

manifold. There are currently four potential energy manifolds that can be used. For practical purposes only the model SteamManifold_L1_boundaries is used in integrated energy systems as it does not include control valves and supports "n" submodules going in and out. The other versions of the energy manifold exist for advanced users in the event the balance of plant or subprocess they are connecting to does not include sufficient valving and control to properly constrain the system. For example, simplified balance of plant systems that do not contain return flow would require the energy manifold to provide makeup water from the condenser, therefore for that scenario we would need to use the SteamManifold_L1_FWH_Cond model which contains a condenser and feedwater heater.
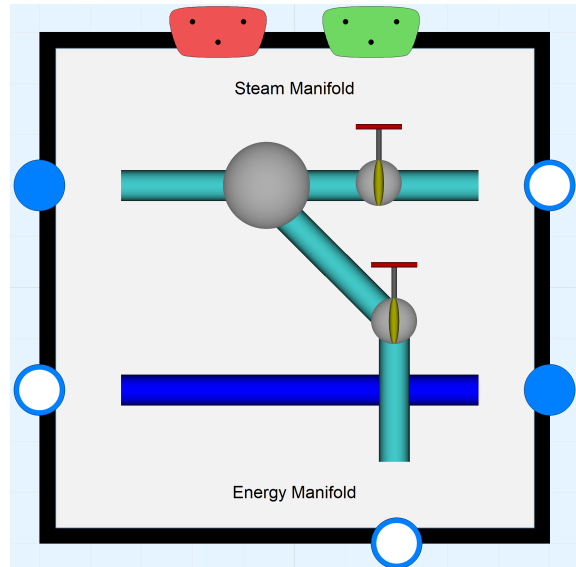
Figure 20: : Top Level Depiction of the Energy Manifold in the NHES package

4.3 Industrial Process

Integrated Energy Systems often include thermal and electrical energy users aside the electric grid. To accommodate thermal energy users and byproduct production the HYBRID repository includes a hydrogen production unit and a reverse osmosis unit.

4.3.1 Hydrogen Production

The hybrid repository includes hydrogen production via high temperature steam electrolysis (HTSE) as shown in Figure 21. HTSE utilizes a combination of thermal energy and electricity to split water into hydrogen (H2) and oxygen (O2) in Solid Oxide Electrolyzer Cells (SOECs), which can be seen in simple terms as the reverse operation of solid oxide fuel cells (SOFCs). The cathode-supported cell consists of a three-layer solid structure (composed of porous cathode, electrolyte, and porous

anode) and an interconnect (separator) plate [8]. An oxygen-ion conducting electrolyte (e.g., yttria-stabilized zirconia [YSZ] or scandia-stabilized zirconia [ScSZ]) is generally used in SOECs [9]. For electrically conducting electrodes, a nickel cermet cathode, and a perovskite anode such as strontium-doped lanthanum manganite (LSM) are typically used. The interconnect plate separates the process gas streams; it must also be electrically conducting and is usually metallic, such as a ferritic stainless steel.

For the HTSE models there are four main models developed by INL, each relying on the same underlying physics of the system but with different control schemes. The HTSE units within the Modelica framework have been specifically designed for integration with light water reactor systems and have been sized with the necessary components to allow for steam side preheating under this assumption. It should be noted that in other HTSE designs there may be varying degrees of preheating equipment based on inlet conditions from the external process. For the HTSE process system parameters are finely tuned and highly non-linear when compared with other process models. Changes in heat exchanger design and sizing can be made directly within the subsystem model however due to the nonlinearity of the system convergence following any changes, a singular component cannot be guaranteed. To modify HTSE stack characteristics the user will need to go two levels down into the HTSE stack system the HTSE stack can be clicked on to open a parameter table where stack characteristics can be modified. Due to the high level of complexity required with HTSE stacks and the customization required depending on the inlet conditions of external system usage of the existing HTSE is preferred, with more details available in two reports published. [2], [10], [11]. Base classes for the HTSE system can found in the "Electrolysis" package within the NHES framework and can be utilized if one desires to create their own HTSE unit.
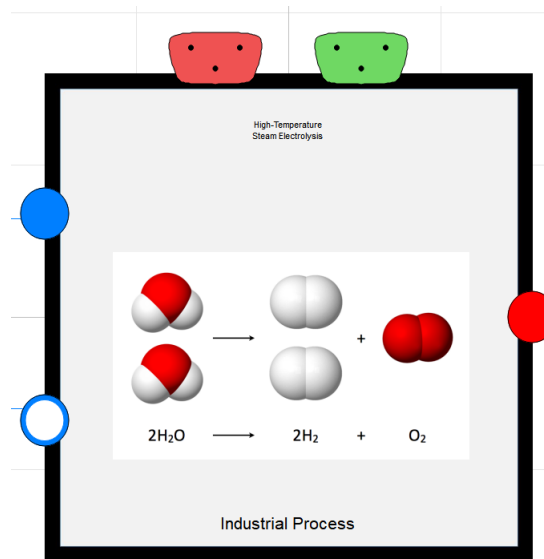

Figure 21: Top Level Depiction of the High Temperature Steam Electrolysis Unit in the NHES package

4.3.2 Desalination

The NHES repository includes a desalination industrial process based on reverse osmosis (RO), Figure 22, designed for brackish water desalination. RO desalination utilizes a semi-permeable membrane, which allows water to pass through but not salts, thus separating the fresh water from the saline feed water. A typical Brackish Water RO (BWRO) plant consists of four main components: feed water pretreatment, High-Pressure (HP) pumping, membrane separation, and permeate (fresh water) post-treatment. The concentrate water rejected by the first membrane module plays a role as the feed water for the second membrane module by the successive order, and so on. These pressure vessels are arranged in rows in each membrane stage, with two-stage membrane separation being typical in BWRO. Each stage has a recovery of 50–60 percent, achieving overall system recovery of 70–85 percent [12].

The Reverse Osmosis Subsystem unit provides the user the ability to modify the number of parallel reverse osmosis units being utilized within the plant alongside to specify how much power is being input into the RO system. Each one of these parallel systems is assumed to go through a two-step the desalination process. In addition, the unit provides the user the ability to alter the salinity of the brine coming into the plant alongside a specified pressure differential across the plant. If further alterations and control are desired from a user perspective reports detailing the full specifications of the plant designs are available in [3], [12]. Additionally, base components for the entire desalination plant can be found in the "Desalination" package within the NHES repository.
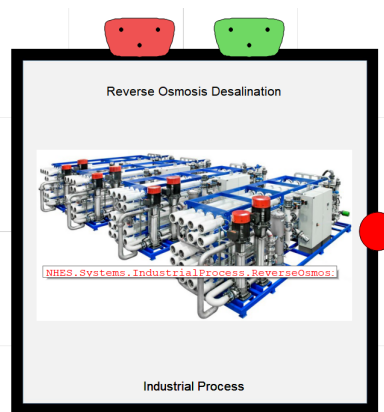

Figure 22: Top Level Depiction of the Brackish Water Desalination Process in the NHES package

4.4 Balance of Plant

There are two main balance of plant models in the hybrid repository. The standard ideal turbine (with and without a condenser and feedwater heater) and step down turbines that allow turbine tap offtake.

4.4.1 Simple Balance of Plant

The balance of plant system consists of an ideal steam turbine model, a condenser, feedwater system for reheat, and a couple of valves that allow steam flow either to the turbine or as bypass to the condenser, Figure 23. Additionally, piping exists to send condensate and rejected heat from ancillary processes directly to the condenser. The balance of plant model can handle supervisory control input for direct control of the turbine control valve and turbine bypass valve based on different sensor input. The main balance of plant system is designed to model Rankine systems.
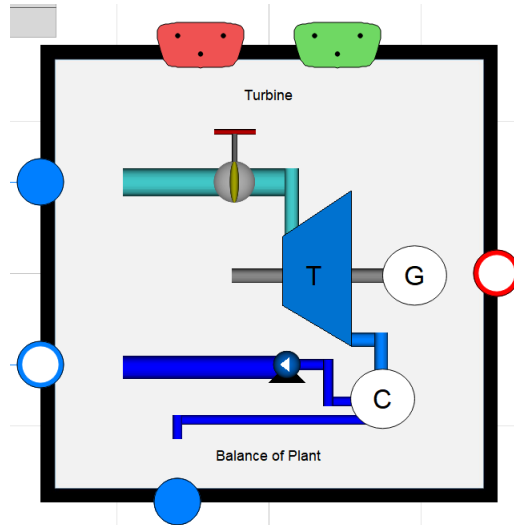
Figure 23: Top Level Depiction of the Balance of Plant in the NHES package

4.4.2 Step Down Turbines

The step-down turbines consist of a series of an ideal steam turbines connected via a singular rotational inertia shaft with bypass tap lines coming off the turbines, Figure 24. The purpose of this model is to allow turbine tap offtake in a dynamic system. The data record within the model includes a series of inputs that allows the user to specify the turbine tap offtake pressures. Additionally, each individual offtake fraction can be input from the data record. The outlet of the stepdown turbines does not include a condenser; therefore, a condenser model would need to be included in a separate system model if the fluid is to be re-introduced into an overall system model.

4.4.3 Stage by Stage Turbine

The stage by stage turbine package is designed to allow for detailed design of a Rankine cycle: including all turbine taps, moisture separators, reheaters, fluid junctions, and peaking capabilities.

Figure 24: Top Level Depiction of the StepDown Turbines in the NHES package

The model uses geometric design rather than system design (pressure ratios, setpoints, efficiencies). The primary design values are cross sectional and design flow deflection angles. Pressure, mass flow rate, and entropic efficiency are all uncontrolled and unspecified.
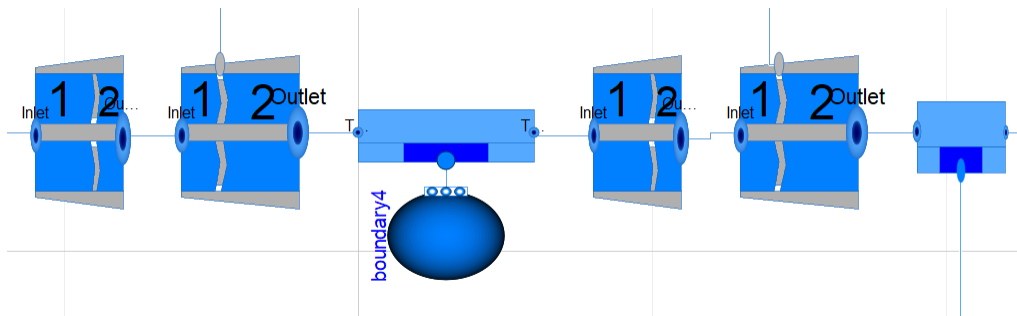
Figure 25: Top Level Stage by Stage Turbine Section With Stators, Rotors, a Tap, and a Moisture Separator
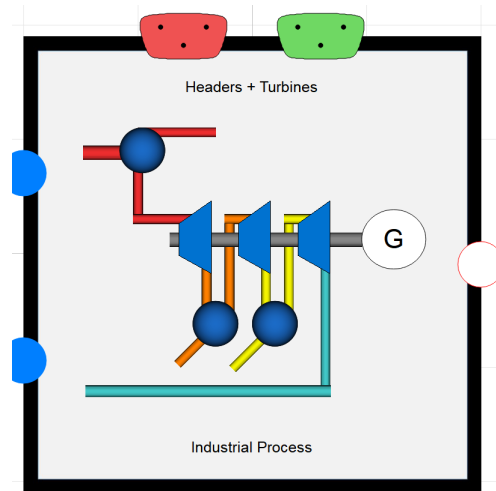
There are 8 primary components used to develop a full stage by stage turbine system. Multiple are shown in Figure 25. The first are the turbine inlet and turbine outlet components which allow for transition between 1D axial fluid components and 3D velocity cylindrical fluid flow components. The next components are the stator and rotor stages. Stator stages deflect incoming flow to better impinge and deposit energy on rotating rotor stages. Rotor stages additionally have torque connectors to connect to a physical turbine model which is responsible for linking the rotor stage torque to the electrical generator. T-junction components setup for cylindrical flow connectors allow for overpowering a rated turbine flow. The final two components deal with removing flow from the turbine. One is a turbine tap, which sets pressures equal between stages. The other is a moisture separator that removes a specific fraction of liquid flow quality.

The stage by stage turbine has been tested from 50-100 percent steam flow.

4.5   Energy Storage

Energy Storage is a large component of Integrated Energy Systems. Currently there are two models of Energy Storage in the repository. Electric Battery Storage, characterized largely as Li-ion battery technology, and two-tank sensible heat thermal energy storage that uses Therminol-66 as the working fluid.

4.5.1   Electric Battery Storage

Electric Battery Storage, shown in Figure 26, is largely characterized as fast and expensive. Due to the speed with which battery storage systems operate, on the order milliseconds, the battery within the hybrid repository has been modeled as a simple logical battery system. The battery can both charge and discharge based upon the direction of electricity flow through the port. It is assumed to be a "perfect" battery and due to the speed of the system, subcomponents have not been modeled simply because they would operate faster than would be useful for the types of analysis utilized with the system. The battery has user-based inputs that control how fast or slow the system can charge and discharge as well as how much energy can be stored within the battery before it is considered full.

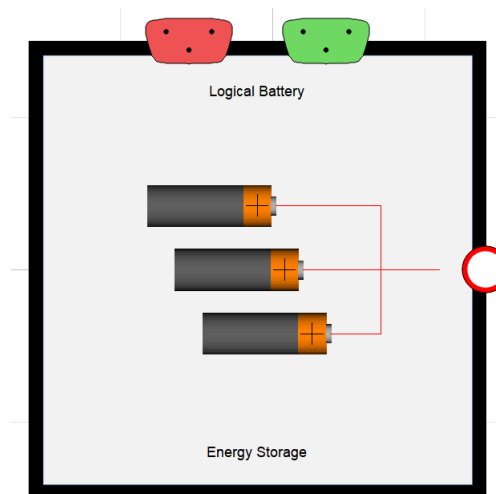**Figure 26:** Top Level Depiction of the Logical Battery in the NHES package

4.5.2   Two-Tank Thermal Energy Storage

Sensible heat storage involves the heating of a solid or liquid without phase change and can be deconstructed into two operating modes: charging and discharging. A two-tank TES system, shown in Figure 27, is a common configuration for liquid sensible heat systems. In the charging mode

cold fluid is pumped from a cold tank through an Intermediate Heat Exchanger (IHX), heated, and stored in a hot tank while the opposite occurs in the discharge mode. Such systems have been successfully demonstrated in the solar energy field as a load management strategy. The configuration of the TES system held within the repository involves an outer loop interfaces with the energy manifold. Bypass steam is directed through an IHX and ultimately discharged to the main condenser of an Integrated system. An inner loop containing a TES fluid consists of two large storage tanks along with several pumps to transport the TES fluid between the tanks, the IHX and a steam generator. Flow Bypass Valves (FBVs) are included in the discharge lines of both the "hot" and "cold" tanks to prevent deadheading the pumps when the Flow Control Valves (FCVs) are closed. Therminol-66 is chosen as the TES fluid as it is readily available, can be pumped at low temperatures, and offers thermal stability over the range (-3°C–343°C) which covers the anticipated operating range of the light water reactor systems (203°C–260°C). Molten salts (e.g. 48 percent NaNO3 – 52 percent KNO3) were not considered, as the anticipated operating temperatures fall below their 222°C freezing temperature. The TES system is designed to allow the power plant to run continuously at 100 percent power over a wide range of operating conditions. During periods of excess capacity, bypass steam is directed to the TES unit through the auxiliary bypass valves where it condenses on the shell side of the IHX. TES fluid is pumped from the cold tank to the hot tank through the tube side of the IHX at a rate sufficient to raise the temperature of the TES fluid to some set point. The TES fluid is then stored in the hot tank at constant temperature. Condensate is collected in a hot well below the IHX and drains back to the main condenser or can be used for some other low pressure application such as chilled water production, desalination or feed-water heating. The system is discharged during periods of peak demand, or when process steam is desired, by pumping the TES fluid from the hot tank through a boiler (steam generator) to the cold tank. This process steam can then be reintroduced into the power conversion cycle for electricity production or directed to some other application through the PCV. A nitrogen cover gas dictates the tank pressures during charging and discharging operation. Full details of the model and its use within integrated energy systems can be found in report [3], [13].

The model itself is coded in a non-conventional manner compared to the rest of the modelica models. It is coded in an input, output sense rather than in a fluid-port, electric-port based modeling system. This is because the model was transferred over from a FORTRAN style code rather than initially coded in Modelica. To modify the two-tank thermal storage system the user will need to look at each individual model within the charging mode and the discharge mode. Base components within the models are fully commented within the code. Like the HTSE the thermal storage unit is finely tuned and thus use outside of its current state will take a bit of work. To help with this the thermal storage unit has been sized to be compatible for varying sizes of offtake from a power unit. One is sized to take 20 percent of nominal steam from a standard 3400MWt Westinghouse plant, and one is designed for 5 percent offtake. Both designed to provide energy back as a peaking unit. The peaking unit is held within the discharge side of the model and is assumed to have its own turbine or is sent back to the low-pressure turbine. Explicit modeling of the coupling back with the low-pressure turbine has not been done. Future updating of the two-tank thermal storage unit to be consistent with other models is planned.
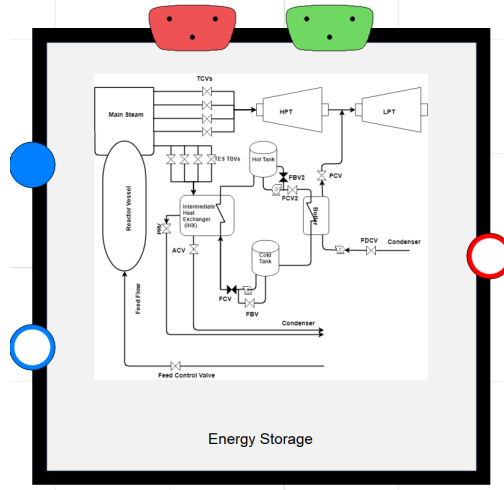
Figure 27: Top Level Depiction of the Two-Tank Sensible Heat Storage Unit in the NHES package

4.5.3 Thermocline Packed Bed Thermal Energy Storage

A thermocline storage system, shown in Figure 28, stores heat via hot and cold fluid separated by a thin thermocline region that arises due to density differential between the fluid. Assuming low mixing via internal flow characteristics and structural design, this thermocline region can be kept relatively small in comparison with the size of the tank. Additionally, large buoyancy changes and low internal thermal conductivity are also extremely useful in maintaining small relative thermocline thickness.

To increase the cost-effective nature of these designs, it is common to fill the tank with a low-cost filler material, such as concrete or quartzite. These filler materials are cheap, have high density, and high thermal conductivity. By using such material, a reduction in the amount of high cost thermal fluid can be achieved, thereby increasing the economic competitiveness of such designs.

The thermocline system was modeled from a modified set of Schumann equations that were originally introduced in 1927 [14]. The equation set governs energy conservation of fluid flow through porous media. His equation set has been widely adopted in the analysis of thermocline storage tanks. The modified equations adopted a new version of the convective heat-transfer coefficient to incorporate low and no-flow conditions from Gunn in 1978 [15]. Additionally, a conductive heat-transfer term was added for the heat conduction through the walls of the tank. Self-degradation of the thermocline in the axial direction is neglected due to low relative values when during standard operation, this is a known limit of the model during times of no flow.
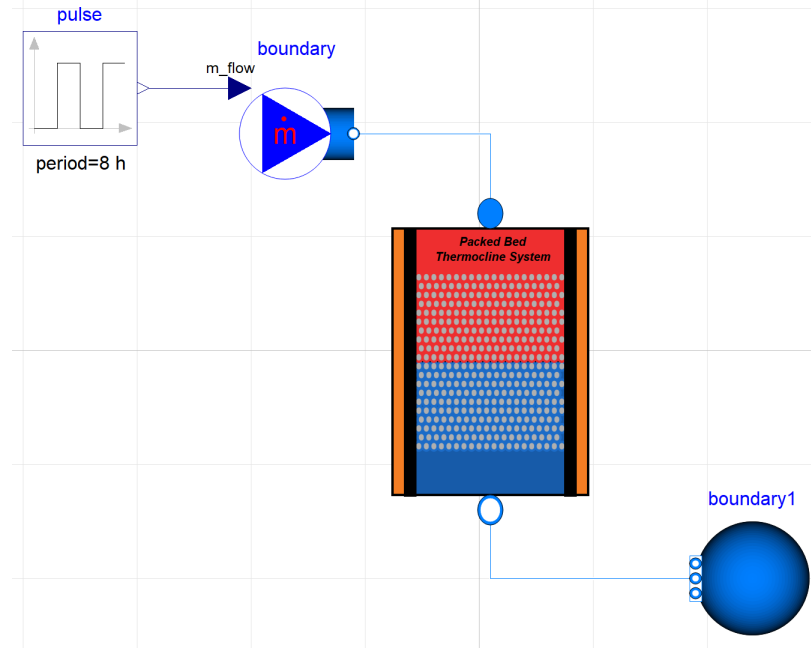
Figure 28: Top Level Depiction of the Single Tank Packed Bed Heat Storage Unit in the NHES package

4.5.4 Concrete Solid Media Storage

Three different models exist for concrete solid media storage, shown in Figure 29. Concrete solid media storage uses inexpensive materials to charge and discharge heat from a fluid system. The models within the Hybrid system use HeatCrete as the concrete material, the properties of which are found in literature. The initial development has been done with energy arbitrage for a Rankine system in mind. However, there are not limitations on the heat transfer fluid that can be used in conjunction with the concrete system. Two design modalities exist with the three developed models. One is a single pipe model and the other is a dual pipe model. All models use simplified flow models: a single pressure and mass flow rate is imposed within a pipe (which allows for significantly improved performance at low and no flow rate conditions) leaving the dynamics of the system to primarily be described by the conservation of energy equation. Within the concrete model, heat conducts both radially and axially in a 2-D nodalized vector. All models calculate values for an average pipe and the system behavior is then scaled by the number of pipes.

The single pipe model is so called because it only models one fluid pipe within a concrete system. This modeling choice imposes a restriction that heat transfer fluid only flows in one direction or the other (note, no error signals are sent by this, m_flow_internal = m_flow_ch-m_flow_dis). The behavior of this system lends itself better to batch energy applications. The power level during quenching (HTF quenching during charging or concrete quenching during discharging) is an order of magnitude higher than the steady power level. The pressure of the system is taken at the

cold end, and defaults to the charging pressure. This model can operate in charging, discharging, OR standby mode. The single pipe model operates with an established axial thermocline and a reversing radial thermal gradient depending on operation.

The dual pipe and dual pipe two HTFs model contain separate pipes for the charging and discharging flow. This model operates more steadily over long periods of time. Heat is always conducted within the CTES and between the concrete and the HTFs. The dual pipe CTES can operate in charging, discharging, standby, or thermal buffer (both charging and discharging) modes. The dual pipe system operates with mono-directional thermal gradients in both axial and radial directions in all operating modes.
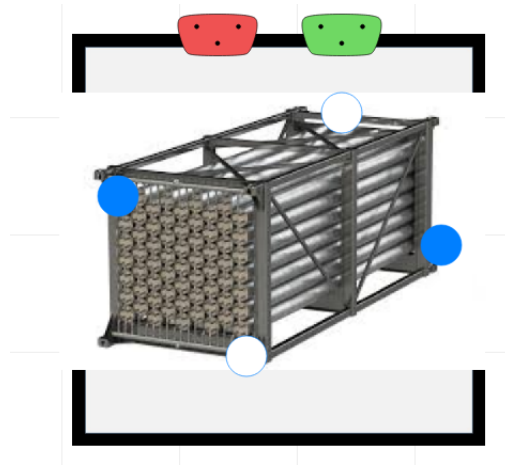

Figure 29: Top Level Depiction of Concrete Thermal Energy Storage System

4.5.5 Latent Heat Thermal Energy Storage

Latent heat energy storage stores energy in materials undergoing phase change. These phase change materials (PCMs) can undergo melting and fusion, boiling and condensation, or hydration and dehydration. PCMs have high volumetric and mass-specific energy storage densities. They can theoretically operate isothermally as the phase change occurs at a single temperature. The ideal PCM would have a large latent heat associated with its phase change, little to no density change between phases, indefinite cyclability between phases, and high thermal conductivity. Research continues in PCMs to identify enhancements that would allow lab-scale experiments to increase size to demonstrate grid-scale applications. A significant challenge is enhancing the thermal conductivity in the PCMs. As the charging or discharging HTF will drive the phase change condition at the heat transfer surface, continuing that process into the PCM mass is based on the thermal conductivity of the material. Conductivity enhancement via geometry specification, micro-encapsulation, and material impregnation has been investigated over time.

The most common PCMs identified operate between the solid and liquid phases where the density change is minimal. Because the melting or solidification front is key to moving heat between

the HTF and the PCMs, many high-fidelity models have been developed across the research field. The model available in the HYBRID repository is based on a paraffin wax experiment. The experiment used water as the HTF to melt and solidify paraffin wax. The original model was built in Star-CCM+ and then converted to Matlab. The Matlab version subsequently has been converted into a Modelica model within HYBRID. The generalized low-fidelity model has been built to accommodate new geometric designs, materials, and HTFs.

The model is a two-dimensional radial conductivity model across three materials: the HTF, a pipe wall, then through the PCM as shown in Figure 30. Within the fluid, the following equation set applies. The equations are also applied within the tube, but the velocity term disappears without internal mass movement. The term α is thermal resistance, equal to thermal conductivity divided by the density and specific heat capacity.

$$\frac{dT}{dt} = \alpha \nabla^2 T - (\overrightarrow{v} \cdot \nabla T)$$

Effectively, the equation set is applied across the material interfaces, with an effective interface thermal conductivity

$$k_{eff} = \frac{2k_1 k_2}{k_1 + k_2}$$

Because there is phase change within the PCM. The equation is written in terms of enthalpy instead of temperature, and the PCM temperature is calculated as a function of the enthalpy (h).

$$\frac{dh}{dt} = \alpha \nabla^2 h$$

The initial models built in Matlab and Star-CCM+ used constant fluid properties and an assumed inlet velocity. The Modelica model has a fluid inlet and exit to allow for replacement of the HTF in the model, and for outer conditions to dictate to the PCM model the fluid conditions at the interfaces.

4.6   Secondary Energy Source

Secondary Energy Sources, or commonly known as peaking units, are an essential part of the energy grid. These systems provide "on-demand" energy during moments when the electrical demand is larger than what the rest of the grid can accomodate. A common feature of
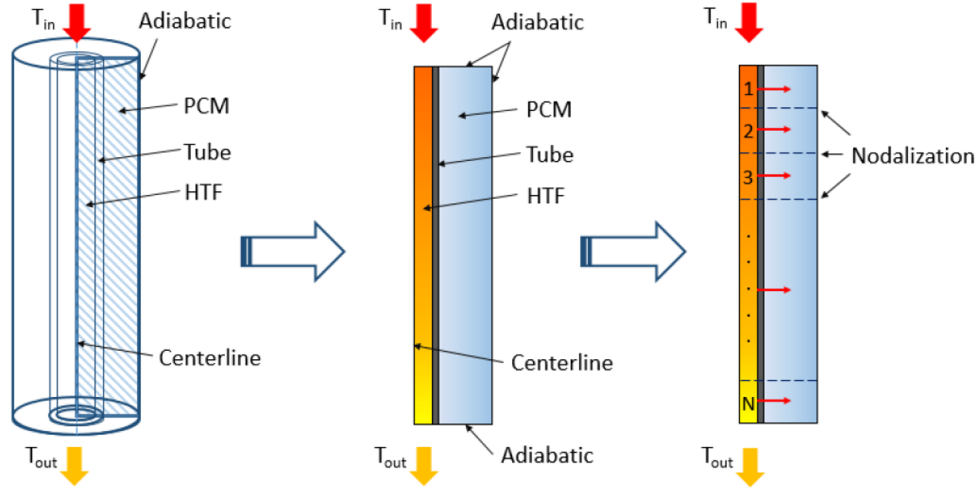
Figure 30: PCM Nodalization diagram

4.6.1 Natural Gas Fired Turbine

Recently, natural gas-fired turbines have found widespread use because of their higher efficiencies, lower capital costs, shorter installation times, abundance of natural gas supplies, lower greenhouse gas emissions compared to other energy sources; and fast start-up capability, which enables them to be used as peaking units that respond to peak demands [16], [17]. Due to their special characteristics, natural gas fired turbines are installed in numerous places around the world and have become an important source for power generation. This section is dedicated to detailed process and control designs of the GTPP, whose primary role is to cover rapid dynamics in grid demand that cannot be met by the remainder of the N-R HES. Simulation results involving several case studies are also provided. Full system details are available in OSTI [2].

The natural gas turbine, Figure 31, is designed with parameters embedded in each individual component. The top level variables can be edited directly within the GTTP_PowerCtrl system. This component is where things such as pressure ratios, flow rates, mechanical efficiencies, and shaft inertia can be modified. The natural gas turbine is designed with a nominal electrical power generation capacity of 35MWe but has a specially designed capacityScaler variable that allows the user to scale the system to between 17 and 70MWe for a singular load. If more are desired then the deployment of several natural gas fired turbines would be required.

4.6.2 Hydrogen Turbine

With the increase in hydrogen production technologies comes on the other end hydrogen burning technologies. To accommodate a hydrogen burning technology the HYBRID repository has been
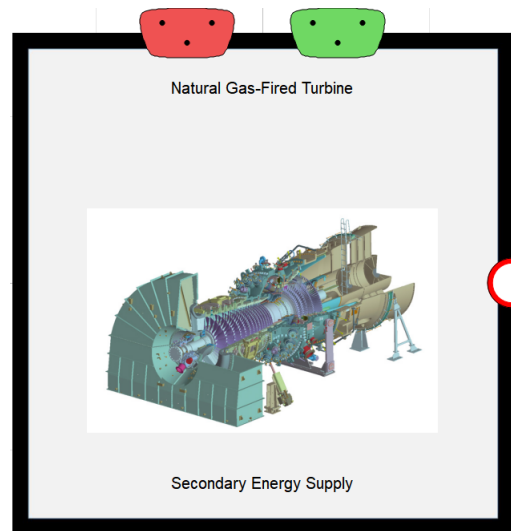
Figure 31: Top Level Depiction of the Natural Gas Fired Turbine in the NHES package

outfitted with a retrofit natural gas burner that is capable of handling pure hydrogen.

The hydrogen turbine, Figure 32, is designed with parameters embedded in each individual component. The top level variables can be edited directly within the Hydrogen_PowerCtrl system. This component is where things such as pressure ratios, flow rates, mechanical efficiencies, and shaft inertia can be modified. The hydrogen turbine is designed with a nominal electrical power generation capacity of 35MWe but has a specially designed capacityScaler variable that allows the user to scale the system to between 17 and 70MWe for a singular load. If more are desired then the deployment of several hydrogen turbines would be required.
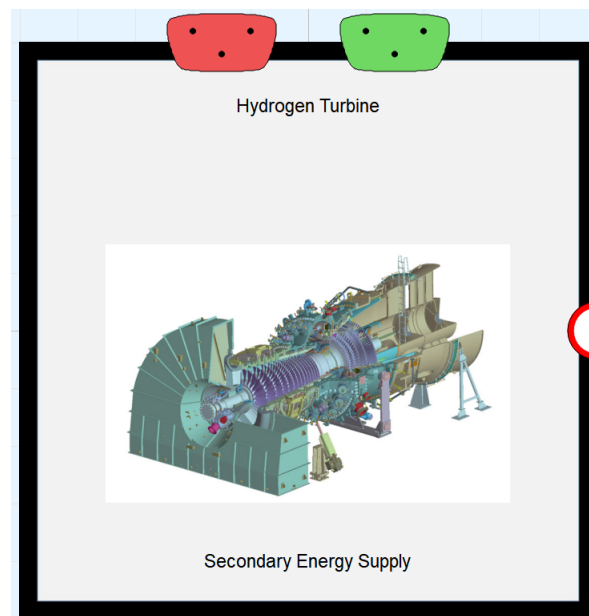
Figure 32: Top Level Depiction of the Hydrogen Turbine in the NHES package

Document Version Information

09a34b32ae7a18d79261047cb9da2e7e7d35523a alfoa Mon, 15 Mar 2021 14:09:50 -0600

References

[1] C. Rabiti, A. Epiney, P. Talbot, J. Kim, S. Bragg-Sitton, A. Yigitoglu, S. Greenwood, S. Cetiner, F. Ganda, and G. Maraonati, "Status report on modeling and simulation capabilities for nuclear-renewable hybrid energy systems," Tech. Rep. INL/EXT-17-43441, Idaho National Laboratory, 2017.

[2] J. Kim, M. Mckellar, S. Bragg-Sitton, and R. Boardman, "Status report on the component models developed in the modelica framework: High-temperature steam electrolysis and gas turbine power plant," Tech. Rep. INL/EXT-16-40305, Idaho National Laboratory, October 2016.

[3] J. Kim and K. Frick, "Status report on the component models developed in the modelica framework: Reverse osmosis desalination plant and thermal energy storage.," Tech. Rep. INL/EXT-18-45505, Idaho National Laboratory, May 2018.

[4] K. Frick, "Status report on the nuscale module development in the modelica framework," Tech. Rep. INL/EXT-19-55520, Idaho National Laboratory, August 2019.

[5] "Modelica standard library." https://github.com/modelica/Modelica.

[6] M. Greenwood, "Transform - transient simulation framework of reconfigurable models," no. doi.10.11578/dc.20171109.1, 2017.

[7] "Westinghouse pwr manual," 1984.

[8] J. Udagawa, P. Aguiar, and N. Brandon, "Improved goodness-of-fit tests," *Journal of Power Sources*, vol. 166, no. 1, pp. 127–136, 2007.

[9] J. O'Brien, "Thermodynamic considerations for thermal water splitting processes and high-temperature electrolysis," *IMECE2008*, 2008.

[10] J. Kim, S. Bragg-Sitton, and R. Boardman, "Status report on the high-temperature steam electrolysis plant model developed in the modelica framework (fy17)," Tech. Rep. doi:10.2172/1408745, Idaho National Laboratory, 2017.

[11] J. Suk Kim, R. Boardman, and S. Bragg-Sitton, "Dynamic performance analysis of a high temperature steam electrolysis plant integrated within nuclear-renewable hybrid energy systems," *Applied Energy*, vol. 228, pp. 2090–2110, 2018.

[12] J. Suk Kim, J. Chen, and H. Garcia, "Modeling, control, and dynamic performance analysis of a reverse osmosis desalination plant integrated within hybrid energy systems," *Energy*, vol. 112, pp. 52–66, 2016.

[13] K. Frick, J. Doster, and S. Bragg-Sitton, "Design and operation of a sensible heat peaking unit for small modular reactors," *Nuclear Technology*, vol. 205, pp. 415–441, 2018.

[14] J. Lew, P. Li, C. Chan, W. Karaki, and J. Stephens, "Analysis of heat storage and delivery of a thermocline tank having solid filler," *Journal of Solar Engineering*, vol. 133, 2011.

[15] T. Esence and et al, "Analysis of heat storage and delivery of a thermocline tank having solid filler," *Solar Energy*, vol. 153, pp. 628–654, 2017.

[16] S. Yee, J. Milanovic, and F. Hughes, "Overview and comparative analysis of gas turbine models for system stability studies," *IEEE Transactions of Power Systems*, vol. 23, pp. 108–118, 2008.

[17] J. Kim, K. Powell, and T. Edgar, "Nonlinear model predictive control for a heavy-duty gas turbine power plant," *American Control Conference*, vol. 23, pp. 2952–2957, 2013.

Idaho National Laboratory